Roope Reimi

Mayariggauksen Mekanismit

Mihin Maya-rigin perustoiminnot pohjautuvat

Metropolia Ammattikorkeakoulu

Tutkinto

Koulutusohjelma

Opinnäytetyö

30.4.2015

Metropolia

Tekijä Otsikko Sivumäärä Aika	Roope Reimi Mayariggauksen Mekanismit 77 sivua + 0 liitettä 30.4.2015
Tutkinto	Medianomin Tutkinto
Koulutusohjelma	Viestinnän Koulutusohjelma
Suuntautumisvaihtoehto	3D-animointi- ja Visualisointi
Ohjaaja	Jaro Lehtonen, Vastaava Opettaja

Tämä on tutkielma Mayan yleisimpien riggaustyökalujen ja mekanismien toiminnasta ja yleinen katsaus Mayan pohjarakenteeseen. Maya on Autodeskin yleis-käytännöllinen 3D-ohjelmisto, joka on saanut teollisuuden aloilla maineen luotettavana, mutta haastavana animaatioalustana. Tässä tutkielmassa aion selvittää muutamia konsepteja, jotka saattavat hämmentää Mayassa ja avata sen logiikkaa toimintojensa takana. Aion myös pureutua syvemmin muutamien Mayan perus riggaustyökalujen toimintaan ja millainen niiden pohjafunktio on.

Tutkielma on vahvan teoria-painotteinen. En käy läpi yksityiskohtaisia riggausohjeita vaan pyrin löytämään yleisiä teorioita ja sääntöjä Maya toimintamekanismien takana. Toivon, että työ saattaisi antaa lukijalle joitakin ideoita, miten soveltaa näitä ohjeita heidän omissa rigeissään. Tutkielman kenties tärkeimpiä aiheita ovat hierarkioiden ja liitosten välinen toiminta ja lokaalin ja globaalin tilan konseptien ymmärtäminen.

Tutkielmassa käytetty pohjaa pitkälti omien kokeilujen ja ajan saatossa kerättyjen havaintojen pohjalta. Apuna ovat olleet Jaro Lehtosen opetukset, Digitaltutors-sivuston videot ja Autodeskin tietokannat. Myös Mayan oma manuaali on ollut tärkeä apu työssä.

Avainsanat	Maya, riggaus, rigi, 3D

Author Title Number of Pages Date	Roope Reimi Mechanics of Basic Maya Rigging 77 pages + 0 appendices 30 April 2015
Degree	Bachelor of Arts
Degree Programme	Media
Specialisation option	3D Animation and Visualization
Supervisor	Jaro Lehtonen, Senior lecturer

This thesis is about rigging with Maya, the generalist 3D-graphics tool owned by Autodesk. The aim of this thesis is to introduce starting Maya riggers to the mechanics behind Maya's basic rigging concepts. I also briefly explain Maya's structure and how it affects the rigging process. With these insights I wish to shed light upon certain concepts of Maya's structure, which could help to improve the trouble-shoot process in rigging. The main goal is that reader understands the underlying concept of how Maya's certain rigging mechanics work and the functions behind them.

This thesis is mostly theory-based. I try to bring smaller parts together to form a bigger concept so that the reader would be able to see the logic behind Maya's system. These theories include the idea of local and global space, transform and shape nodes, hierarchies, object orientations, connections etc. Finally, I discuss a practical example in reference to theories I have introduced.

In the report, I give a detailed account of the findings I have made while creating my own rigs. Most of the material I have used is from the site of the Autodesk and the Maya manual. I have also consulted my instructor Jaro Lehtonen about certain aspects. Another great source of information has been video tutorials at Digitaltutors.com.

Keywords	3D animation, rigging, Maya

Sisällys

1 Johdanto

Tämä on tutkielma Maya-riggauksen mekanismeista ja sen pohjalla toimivasta rakenteesta. Riggaus on prosessi, jossa 3D-hahmolle rakennetaan luuranko ja ohjainjärjestelmä, jolla tätä hahmoa animoidaan. Riggaus on astetta mekaanisempi työvaihe 3D-hahmon rakennusprosessissa. Rigin täytyy pystyä toteuttamaan animaattorien vaatimat liikeradat ja kyetä mahdollisimman monipuolisiin tuloksiin.

Maya on yksi teollisuuden käytetyimpiä 3D-alustoja. Ammattipiireissä se on hyvin usein käytetty ohjelmisto etenkin riggauksen ja animoinnin tullessa kyseeseen. Maya mahdollistaa osaavissa käsissä monipuoliset ja monimutkaisetkin ohjain-järjestelmät, jotka puolestaan mahdollistavat sulavan animaation. Olen itse käyttänyt Mayaa toisesta opiskeluvuodesta lähtien. Aikoinaan ilmoittauduin ryhmämme lyhyt-animaatioprojektin riggaajaksi pahemmin olematta perillä siitä, mihin olin ryhtymässä.

Lopulta kävi niin, että ensimmäisessä rigissä oli liikaa vikoja, joiden korjaus olisi vienyt turhan kauan aikaa. Siinä meni noin kahden kuukauden työ hukkaan. Tuolloin työalustanani oli Maya ja suurin osa työskentelystäni oli pohjautunut oman työn ja opettajan työprosessia vertaillessa. Tuolloin sain kantapään kautta kokea monet virheet ja haasteet Mayalla rigatessa. Jälkikäteen ajateltuna monet virheistä johtuivat siitä, etten täysin ollut perehtynyt Mayan perustoimintaperiaatteisiin. Olin pääasiassa kopioinut opettajan tekemät sovellukset miettimättä mekanismeja niiden takana. Jos olisin perehtynyt näihin mekanismeihin etukäteen, luultavasti ongelmat eivät olisi olleet liian ylitsepääsemättömiä.

Tästä huolimatta koin Mayan käyttöjärjestelmän kiehtovaksi ja monipuoliseksi alustaksi rigien tekemiseen. Vaikka esimerkiksi 3DsMaxin selkeämmin esivalmisteltuihin riggaustyökaluihin oli helpompi päästä sisälle, koin Mayan sisältävän enemmän potentiaalia tulevaisuutta ajatellen. Tämä tutkielma on rakennettu sitä varten, että Mayan kanssa riggaustyötä aloitteleva henkilö voisi päästä helpommin ja nopeammin perille niistä perus toimintamekanismeista ja logiikasta, johon Mayan riggausjärjestelmä, ja myöskin koko työtila, perustuu.

1.1 Mistä tutkielmassa on kyse

Tutkielman peruskysymyksenä on: ”Mitkä ovat toimintaperiaatteet ja peruskäytännöt Maya-riggauksessa?” Tutkielmani lähtökohtana on ensisijainsesti esitellä mekanismit ja

mayan työkalujen takana, mutta ei antaa yksityiskohtaisia kuvauksia Mayan työkalujen asetuksista ja käytöstä. Näitä asioita kannattaa katsella Mayan oman manuaalin kautta.

Maya rakentuu monille sellaisille yleiskäsitteille kuten hierarkiat, sidokset ja noodit. Näiden periaatteiden tutkiminen on tutkielmani varsinainen aihe. Näiden ja monien muiden teesien ymmärtäminen on mielestäni välttämätöntä, jos haluaa saada Mayan työkaluista kaiken irti. Monet ongelmat pohjautuvat myös näiden ominaisuuksien virheellisiin asetuksiin. Hierarkiassa ja orientaatioissa tapahtuvat virhelaskelmat ovat syitä monien Maya-rigien ongelmien takana.

Muita tärkeitä käsitteitä ovat lokaalin ja globaalin tilan ja napojen sijainnit ja miten ne vaikuttavat objektien toimintaan. Aion tutkia näitä asioita ja selvittää logiikkaa näiden toimintojen takana.

1.2 Miten asiat esitellään

Olen asetellut esiteltävät asiat ja teoriat osin siinä järjestyksessä, missä ne ilmenisivät hahmo-rigiä rakennettaessa. Esimerkiksi hierarkiat käsittelen kolmannessa luvussa nivelien (joints) yhteydessä. Nivelet ovat rigin rakennusprosessista useimmiten ensimmäisiä, joiden kohdalla joudutaan tekemisiin hierarkioiden kanssa. Missään nimessä ei kannata olettaa tässä oppaassa esitetyn työjärjestyksen olevan ainoa mahdollinen.

Alun perin ajattelin tehdä tutkielmasta eräänlaisen opastuksen humanoidihahmon rigaukseen, mutta lopulta ajan loppuessa ja heikosti asetellun aikataulun takia päätin luopua ajatuksesta. Hahmon rigausprosessin yksityiskohtainen dokumentointi olisi muutenkin paljon yksinkertaisempaa ja selkeämpää videomuodossa, koska katsoja voi turvautua vahvasti esimerkin apuun. Monet tässä oppaassa esitellyt kuvaukset kuitenkin pohjaavat tämän hahmon rakennuksen pohjalta tehtyihin havaintoihin.

Toinen luku kattaa yleisiä huomioita Mayan toimintamekanismeista ja niiden soveltamisesta. Katsahdan jo noodien välisiä liitoksia ja muutamia työkaluja, joilla noodeja voidaan käsitellä. En kuitenkaan esittele näiden työkalujen toimintaa kovinkaan yksityiskohtaisesti.

Kolmas käsittelee hierarkioita, lokaaleja ja globaaleja tiloja ja mittasuhteita, jotka linkittyvät hahmon luuston rakennukseen. Kolmannen luvun lopulla käsittelen nivelien toimintaperiaatteita ja hivenen meshin sitomista luustoon, mutta jälkimmäisen asian jätän kuitenkin tilan vuoksi vähemmälle huomiolle.

Neljännessä luvussa katsellaan ohjain-objektien toimintaa ja yleisien rigin työkalujen toimintaperiaatteita, kuten ik-ketjuja. Viides luku kattaa rigin yksityiskohtien toimintaan limittyviä työkaluja kuten extra-ominaisuuksia, omien riippuvaisliitosten luomista ja ”set driven key”-systeemiä. Kuudes luku käsittelee deformereita, jotka teemoiltaan limittyisivät viidenteen lukuun, mutta ovat sen verran poikkeavia toiminnoiltaan, että koen tarpeelliseksi pitää ne erillään.

Sen lisäksi tutkielma sisältää muutaman yksityiskohtaisemman käytännön esimerkin muutamien toimintojen soveltamisesta hahmon riggauksessa. Käsiteltävät esimerkit saattavat sisältää sellaisienkin toimintojen esittelyä, jotka eivät enää olleet varsinaisen pääluvun aiheena. Tämä siksi, että tietyt esimerkit pystytään toteuttamaan vasta sellaisten toimintojen kanssa, jotka käsitellään tutkielmassa vasta myöhemmin. Siksi esimerkiksi ik-spline ketju esitellään jo neljännessä pääluvussa, mutta käytännön sovellus esitellään vasta kuudennessa pääluvussa, koska viimeisin ik-splinen käsittelyssä tarvittava ominaisuus on käsitelty.

1.3 Kenelle tutkielma on suunnattu

Tutkielma on ennen kaikkea suunnattu heille, jotka ovat jo osin perehtyneet Mayan käyttöliittymään ja kenties jopa käyttäneet muutamia tässä oppaassa tutkittavia työkaluja. Toivottavaa myös olisi, että lukija tuntee jo Mayan muutamia olennaisia termejä ja käsitteitä. Henkilön ei välttämättä tarvitse olla täysin suuntautunut riggaukseen, vaan voi esimerkiksi olla animaattori, joka tarvitsee hieman käsitystä siitä, miten Mayan työkalut ja siten rigit rakentuvat. Tämä voi edesauttaa paljonkin animaattorin ja riggaajan keskinäistä työskentelyä, koska Mayaa pidetään hyvin vahvasti animaatio- ja riggaus-orientoituneena.

Lukijan pohjaosaaminen on puolestani siksi toivottavaa, sillä todennäköisesti kaikki tutkielman käsitteet eivät ole niin selkeästi pohjustettuja kuin toivottavaa olisi. Toivon silti, että kyseisiä käsitteitä ja käytäntöjä jo hivenen ymmärtävä lukija saa tästä tutkielmasta asteen syvempää ymmärrystä Mayan käyttöön.

1.4 Käsitteitä

Tässä tutkielmassa käytän useimmiten erilaisten työkalujen suomennettuja nimiä ja esitän alkuperäiset nimikkeet sulkeissa. Joidenkin asioiden kohdalla saatan taas käyttää suomalaista väännöstä englannin kielen sanasta, joka kenties ei ole kaikkein oikeaoppisin, mutta jonka Mayaa käyttäneet ymmärtävät hetkessä.

Meshi (Mesh): Polygoneista koostuva 3D-malli.

Vertex (control vertex): Komponentti, joka yhdistettynä muihin vertexeihin muodostaa 3D-pintoja. Referoin vertexeihin joskus sanalla kontrolli-vertex.

Polygon: Vähintään kolmen vertexin muodostama pinta. Teollisuudessa suositaan vahvasti nelikulmaisten polygonien käyttöä.

Rigi (rig): Toinen käsite on “character setup”. 3D-objektiin liitetty ohjainjärjestelmä, johon meshi on sidottu, ja jolla hahmoa liikutellaan. Ei ole yhtä ainoaa oikeaa tapaa rakentaa rigiä.

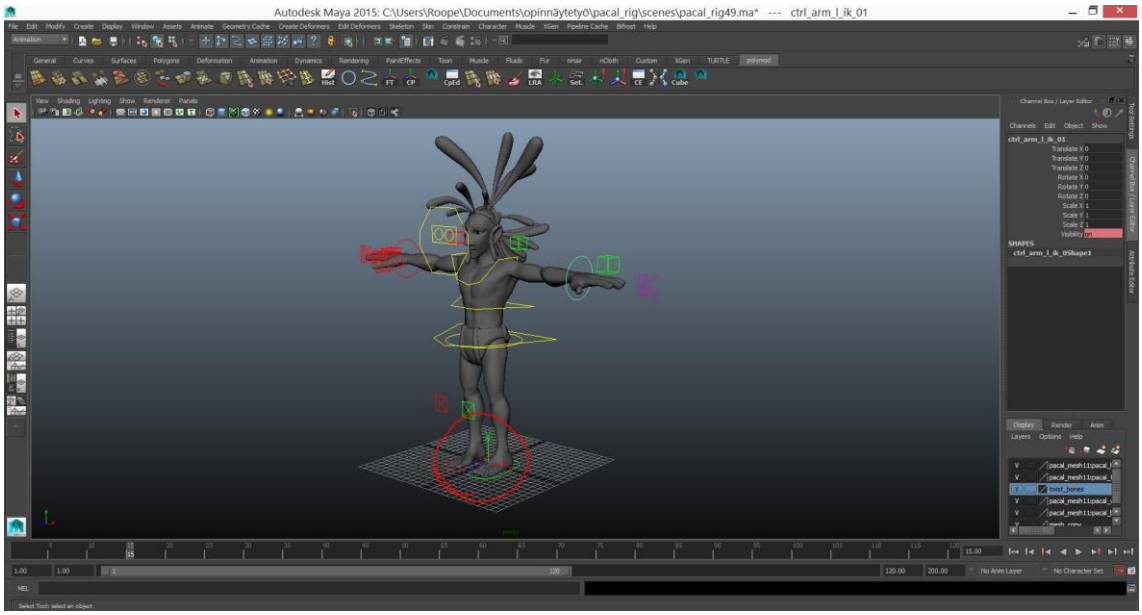
Riggaus: Rigin rakennusprosessi tai rigin sovittaminen 3D-objektiin.

Skinnaus (skinning): Prosessi, jossa mesh liitetään valistettuun rigiin.

Työtila (scene): Työtilalla tarkoitan Mayan perus työnäkymää. Scene rakentuu erilaisten noodien ja skriptien yhteensulaumasta. Englannissa työtilasta käytetään käsitettä ”world” ja työtilasta käsitettä ”world space”. Kun puhun globaalista nollakohdasta, tarkoitan sillä Mayan työtilan äärimmäistä nollakohtaa (world origin). Hierarkioiden tapauksessa objekti, joka ei ole hierarkioitu mihinkään muuhun objektiin, on hierarkioituna työtilaan (englannissa käsite on ilmaistu: ”parented to the world”).

Objekti: Tässä oppaassa tarkoitan jokaista Mayan työtilassa olevaa objektia, jonka pystyy valitsemaan outlinerin tai object-moden avulla. Objektityypit voidaan jakaa vielä DAG- sekä DG-objekteihin.

Komponentti (component): Objektin rakennusosa. Mesh-objektissa voi tarkoittaa esimerkiksi yksittäistä vertexiä. Tässä oppaassa kutsun komponentiksi jokaista sellaista asiaa, joka on skenessä, mutta ei ole varsinainen objekti.



Kuva 1. Scene on nimike kokonaiselle työtilalle. Se on alue, jossa hahmoa, sen meshiä ja rigiä käsitellään. Englannin kielessä scenen tilasta saatetaan käyttää käsitettä ”world”. Ympyröity piste kuvaa tilan globaalia nollapistettä, joka määrittää itse työtilan akseliston.

Napa (pivot): Napa tarkoittaa objektin nollakohtaa ja rotaatioakselia, jonka ympäri rotaatio tapahtuu. Kun objektia käännetään esimerkiksi x-akselilla, objekti kääntyy oman nollakohtansa ympäri. Esimerkiksi kuun rotaation napa on maa ja maan rotaation nollakohta on aurinko.

Transformaatio: Transformaatiolla tarkoitan niitä muutoksia, joita tapahtuu objektin translaatiossa (translate), rotaatiossa (rotate) ja skaalauksessa (scale). Transformaatiot tapahtuvat xyz-koordinaatistossa. Yhtä koordinaatin suuntaa kutsutaan akseliksi (esimerkiksi x-akseli). Objektin sijainti, kallistuskulma tai skaala voi muuttua yhdellä tai useammalla akselilla kerrallaan.

Translaatio (translate): Yksi perus muutosominaisuus Mayassa on objektin sijainnin muutos työtilassa ja hierarkiassa. Translaatiossa tarkemminkin liikutetaan objektin rotaationapaa. Channel editorissa nähtävä translate-arvo kuvaa tätä.

Rotaatio (rotate): Toinen tärkeä muutosominaisuus, jolla objektin kallistuskulmaa muutetaan sen omaan napaan nähden.

Skaalaus (scale): Kolmas muutosominaisuus. Skaalaus voi muuttaa objektin suuruutta yhdellä tai useammalla akselilla.

Deformaatio: Control vertexien sijaintien muutos. Tällä tarkoitetaan tässä tutkielmassa objektin meshin muokkaantumista eikä niinkään sijainnin tai kallistuskulman muutosta. Deformaatiot kuitenkin tapahtuvat myös xyz-koordinaatistossa, mutta niiden sijainti mitataan eri tavalla. Deformaatiota ei pidä tässä tutkielmassa sekoittaa transformaatioon, joka liittyy objektien liikutukseen. Käsittelen deformaatioita tarkemmin tutkielman lopuilla.

2 Yleistietoa Maya-riggauksesta

2.1 Mayan rakenne

Maya on useiden yritysten, kuten Aliaksen ja Wavefrontin vuosien yhteistyön tulos. Tämä näkyy Mayan hyvin laaja-alaisessa ja löyhässä rakenteessa, jossa on piirteitä monista erilaisista ohjelmistotyypeistä.

Mayan työtilan yleisrakennetta voi kutsua noodi rakenteiseksi arkkitehtuuriksi, jollaisia löytyy monista muistakin ohjelmistoista kuten Nuke-ohjelmistosta (wikipedia 2015a). Noodipohjaisessa arkkitehtuurissa pienet noodikomponentit yhdistyvät toisiinsa liitoksilla ja hierarkioilla, muodostaen ohjelman työympäristön. Useimmiten tämä noodien luoma verkosto on tarkasteltavissa kaksiulotteisessa kartastosta. Mayassa näitä noodeja ja niiden muodostamia liitoksia pystyy katselemaan esimerkiksi node-editorin ja hypergraphin kautta (wikipedia). Tarkempia tietoja näiden apuvälineiden käytöstä kannattaa tarkastella Mayan manuaalista.

Mayan noodirakenteen huomioiminen on tärkeä askel Mayan riggauksen ymmärtämisessä, sillä rigit ovat pohjimmiltaan lukuisten noodien luomia verkostoja. Täysin noodien tasolle ei kuitenkaan tarvitse mennä, jotta riggaus Mayalla onnistuisi.
Useimmiten rigeissä käytettävät välineet ovat objekteja, jotka ovat rakentuneet erilaisten noodien yhdistelmistä ja liitoksista.

Maya-riggaus on monille vasta-alkajille melkoinen haaste juuri Mayan rakenteen monimutkaisuuden vuoksi. Vapaus tehdä täysin omia muokkauksia käyttöjärjestelmään ei välttämättä tee palvelusta käyttäjälle. Aloittelevat Maya-riggaajat saattavat jättää asiat

kesken kun tarvittavaa kosketuspintaa riggauksen aloittamiseen ei löydy. Monet Mayan tarjoamista työkaluista eivät välttämättä ole edes kaikkein suoraviivaisimpia toiminnassaan. Mayan yhteydessä on monesti puhuttu siitä, miten monipuolisten mahdollisuuksien vastapainona Maya edellyttää turhankin pitkällistä perehdytystä ja mekanismien opettelua, jotta sen potentiaalista saa kaiken irti.

2.1.1 MEL-skripti

MEL (maya embedded language) on Mayan oma skriptauskieli. Ohjelman kiinteintä ydintä lukuun ottamatta kaikki Mayan toiminnot pohjaavat MELiin. Kaikki työtilan ikkunat, järjestelmät, kamerat, työkalujen toiminnot ja ynnä muut rakentuvat sen varaan. Käyttäjälle itsellään on mahdollisuus kirjoittaa MEL-komentoja tehtävien suorittamiseksi tai sitten kirjoittaa kokonainen koodausrivistö ja varastoida se yksittäiseen näppäimeen työkalurivistössä (kuva). MELin suorittamia komentoja voi tarkkailla script editorin kautta.

Riittävällä määrällä tietoa riittävällä MELin tuntemuksella on mahdollista muokata sen käyttöjärjestelmää, tehdä omia työkalusovelluksia ja luoda omia työkaluja (Harovas, Kundert-Gibbs, Lee, 2000, s. 498.). Monet ohjelman mukana tulleista työkaluista ovat ensin saattaneet olla yksittäisen tekijän valmistamia omia skriptejä, mutta ovat myöhemmin päässeet osaksi virallista pakettia.

En käsittele MEL:in käyttöä tässä oppaassa. Esimerkkioppaassa käytetyt työkalut ovat Maya 2015 mukana tulleita. Koin kuitenkin tarpeelliseksi aloittaa sanasella MEL:illä, koska kyseessä on lopulta Mayan toiminnan kivijalka.

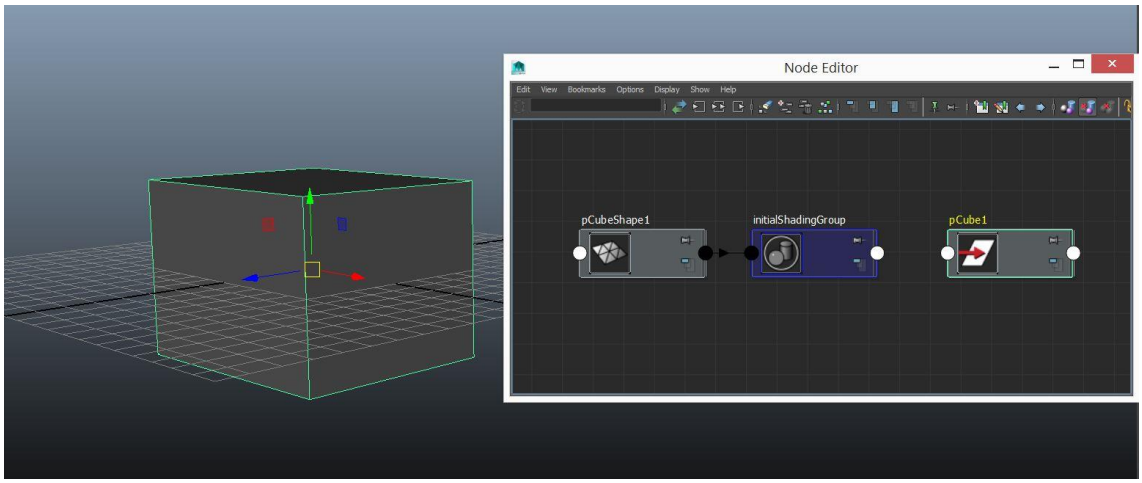
2.2 Noodit

Noodit ovat Mayan rakenteen kivijalka ja syvin osa Mayan rakenteessa, jota tulen käsittelemään tässä oppaassa. Noodit ovat näkyvän Maya skenen pääraaka-aine. Kaikki objektit, kuten kamerat, valot ja meshit rakentuvat noodien kautta.

Jokaisella noodilla puolestaan on omat ominaisuutensa (attributes), jotka määrittelevät noodin käyttötarkoituksen ja tehtävän. Noodit ovat ikään kuin datasammioita, jotka toimivat kuin komponentit piirilevyssä. Komponentit eivät tee paljoa mitään yksinään vaan ne on ensin yhdisteltävä ominaisuuksiltaan tai hierarkialtaan toisiinsa (Autodesk, 2015b.)

Jokainen objekti ja toiminto skenessä (scene) koostuvat, yhdestä tai useammasta noodista. Jotta jokainen objekti toimisi tehtävänsä mukaisesti, sen erinäisillä toiminnoilla ovat omat noodinsa omilla ominaisuuksillaan.

Esimerkiksi cube-primitiivissä cuben sijainnista vastaa transform noodi, joka sisältää ominaisuudet kuten translate, rotate sekä scale. Itse cuben muodon (shape) määrittää erillinen shape-noodi, joka on hierarkiassa transform noden alla (kuva 2.).
Noodeja pystyy tarkastelemaan hypergraphin sekä node editorin avulla, joka on uusimpia lisäyksiä mayassa.

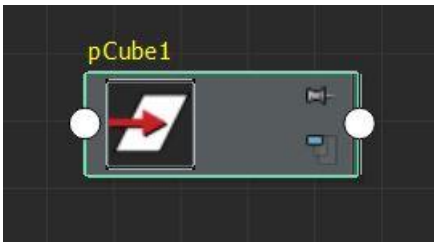


Kuva 2. Kuvassa näkyvät oikealla kuution muodostavat noodit. Noodin vasemmassa reunassa sijaitsevat informaatiota vastaanottavat input-portit ja oikealla sivulla informaatiota lähettävät output-portit. Tässä kuvassa pCubeShape1 lähettää informaatiota output-porttinsa kautta initialShadinGroupin input porttiin. pCuben ja ja pCubeshapen välillä ei näy nuolta, koska niiden keskinen liitos on hierarkkinen.

Noodien perustyyppejä ovat esimerkiksi (Autodesk, 2015b.):

Transform-noodi:

Transform node on yksi yleisimpiä noodityyppejä. Transform noodi sisältää objektin translation, rotate sekä scale ominaisuudet, mutta kätkee sisäänsä, joskin piilotettuina, myös muita ominaisuuksia. Transform noodia voisi kuvailla monikäyttöiseksi vaasiksi, johon voi linkittää erilaisia objekteja, joita olisi tarkoitus liikutella ja animoida. Node-editorissa, hypergraphissa ja outlinerissa transform noodeilla on useimmiten seuraavanlainen symboli:



Shape –noodi:

Mallintajat ovat varmaankin jo tietoisia siitä, miten polygonimallin tai surfacen pintoja käsitellään. Jos itse mallin vertexejä siirrellään, se ei vaikuta objektin sijaintiin object-modessa. Tämä siksi, että shape noodi on aina jonkin toisen noodin child, eikä voi toimia yksinään. Skenessä oleva cube-malli koostuu siis hierarkiasysteemistä, jossa transform noodi on parent cuben shapenoodille. Transform noodi määrittelee shapen sijainnin ja toimii osana, johon voidaan linkittää muita ominaisuuksia tai josta voidaan linkitää muihin noodeihin. Transform noodi on siis ikään kuin kuin vaasi, ja shape on vaasin sisällä oleva vesi. Veden paikkaa ei itsessään voi muuttaa, ellei sitä aseteta johonkin kuljetusastiaan. Ja tässäkin tapauksessa veden sijjainnille ei voi antaa yhtä selkeää määrettä, koska vesi on useiden partikkeleiden summa. Shape noodi sisältää siis objektin control vertexien ja siten myös edge-, face- ja käyrä-ominaisuuksien sijainnit. Control vertexit ovat näkyvän meshin perusraaka-aine, mutta niillä itsellään ei ole liitettäviä arvoja, ne ovat siis verrattavissa veden vastaaviin partikkeleihin. Control vertexiä ei voi siis sellaisenaan linkittää esimerkiksi toiseen pallo-objektiin ellei control vertexiin liitetä esimerkiksi clusteria.

Utility- noodit:

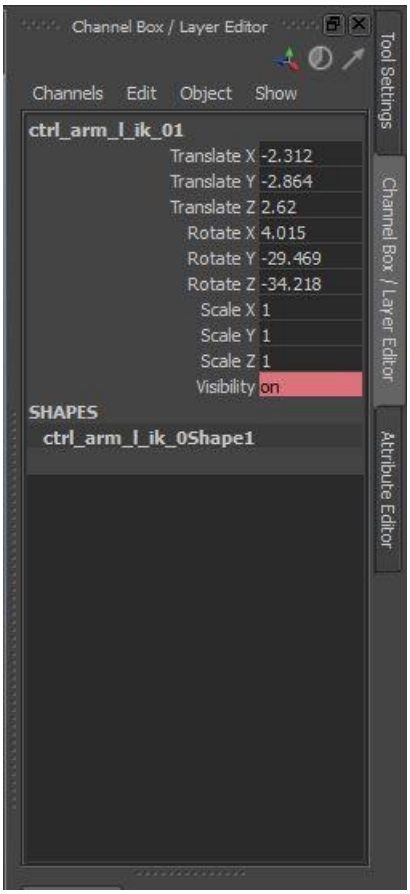
Utility noodit ovat erityinen noodien ryhmänsä, joka ennen kaikkea pyrkii suorittamaan erilaisia aputehtäviä, jotta erilaiset noodiverkostot toimivat. Utility noodit voivat esimerkiksi mitata etäisyyden kahden objektin välillä ja lähettää tämän tiedon eteenpäin toisille noodeille. Yksi yleisimpiä käytettyjä noodeja on multiply/divide –noodi, joka voi kertoa, jakaa tai suorittaa potenssilaskun numeraaliarvolle ja lähettää tämän arvon toiselle noodille. Tulen käyttämään muutamia näistä esimerkkirigissä, jos mahdollista.

Noodityypit eivät rajoitu ainoastaan näihin. Mayasta löytyy myös sellaisia noodityyppejä kuin rendausnoodi ja tekstuurinoodi. Myös näiden noodien ominaisuuksia on mahdollista animoida, kuten muitakin. Tämä opas ei kuitenkaan kata näihin noodeihin limittyviä kysymyksiä.

2.2.1 Ominaisuudet (Attributes)

Ominaisuudet (esimerkiksi translateX tai rotateY) ovat noodeissa esiintyviä toimintoja, joilla on säädeltävä ja yhdisteltävä arvo (kuva 3.). Tämä arvo voi olla numeraalinen, tautologinen (true/false, 1/0 tai on/off) tai listaus.

Attribuutit ovat noodien oma rakennusaine, joka määrittää noodin roolin ja tehtävän. Käytettävät ominaisuudet eivät kuitenkaan rajoitu pelkästään näihin. Attribuutteja voi tarvittaessa rakentaa lisää esimerkiksi channel editorin kautta. Tämä pätee niin noodeihin, jotka ovat näkyvissä kuin myös näkymättömissä. Nämä itse rakennetut attribuutit eivät itsessään kuitenkaan tee mitään ennen kuin ne on yhdistetty toisiin attribuutteihin tai niiden tehtävä on säädetty koodilla (Harovas, Kundert-Gibbs, Lee, 2000, s. 511).



Kuva 3. Translate, rotate ja scale-ominaisuudet kuten ne on esitetty channel editorissa. Visibility näyttäytyy punaisena, koska tähän ominaisuuteen on kytketty animaatio.

Toinen mainittava asia on, että jokainen ominaisuus missä tahansa noodissa on animoitavissa, eli sen arvoa on mahdollista muuttaa ajan kuluessa. Karkeasti ottaen voisi siis sanoa, että niin kauan kuin jokin Mayassa on animoitavissa, sen on myös rigattavissa. Joissain tapauksissa, jotkin ominaisuudet on ensin erikseen liitettävä objektin channel editoriin, jotta animaatio on mahdollista. Käsittelen tätä viidennen luvun ”driven key”-osassa.

2.2.2 DAG-hierarkia

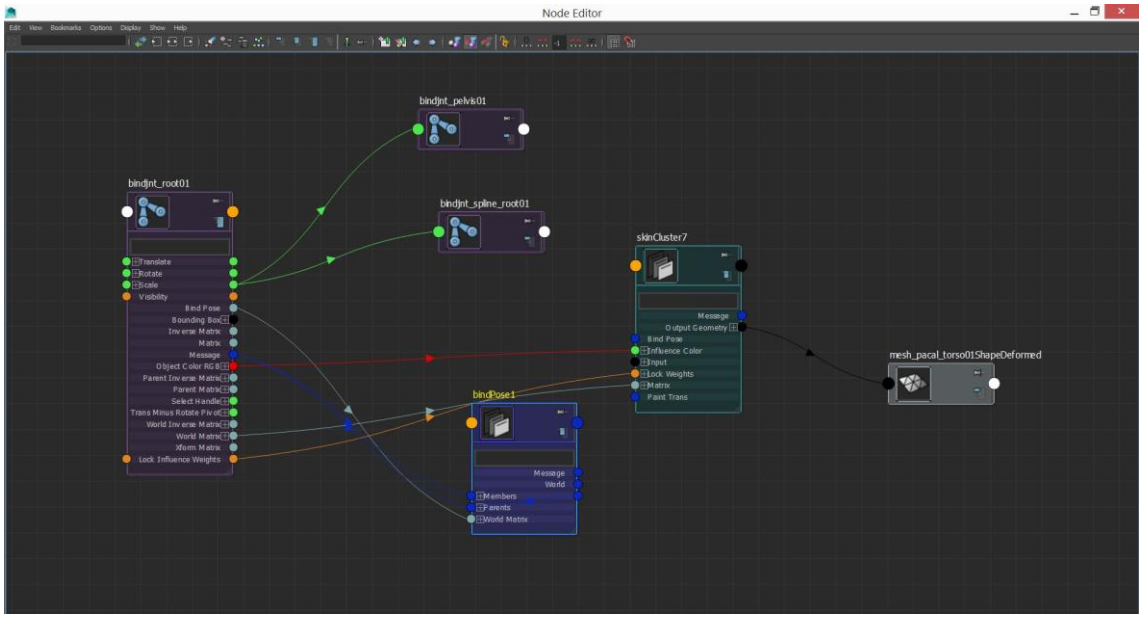
DAG hierarkiaa ei pidä sekoittaa objektien keskiseen hierarkiaan, jota käsittelen seuraavassa luvussa. DAG-hierarkia (directed acyclic graph) tunnetaan myös objektihierarkiana. DAG hierarkia tarkoittaa täsmällisemmin noodien keskeistä hierarkiaa, joka muodostaa objektin. DAG hierarkia tarkoittaa siis objektin sisäistä hierarkiaa. Jokainen objekti koostuu kahdesta noodityypistä, transform ja shape (Autodesk 2015c.) Aiemmin kuvassa 2. oli esitettynä polygonikuution sisäinen hierarkia, jossa shape- ja transform-noodi muodostivat näkyvän kuution.

DAG noodien lisäksi Mayaan kuuluvat DG-noodit (dependency graph nodes). Nämä noodit liittyvät osaltaan DAG-noodeihin, mutta DG-noodit pystyvät linkittymään keskenään syklisesti, kun taas DAG-noodit voivat linkittyä vain lineaarisesti. Tässä ilmenee

myös hierarkkisten ja riippuvaisliitosten suurin ero. Riippuvaisliitoksia on mahdollista rakentaa siten, että informaatio kulkee ympyrää, syklisesti. Hierarkkisessa liitoksessa informaatio taas kulkee lineaarisesti.

2.2.3 Riippuvaisliitokset (Dependency connections)

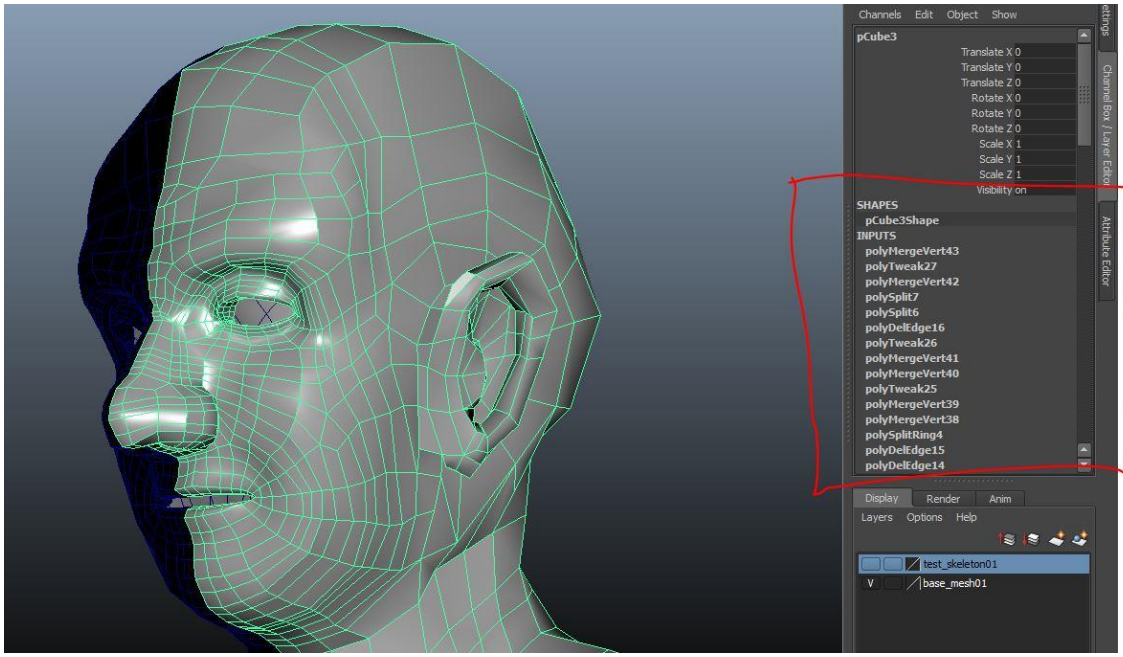
Riippuvaisliitokset tarkoittavat noodien ominaisuuksien välisiä liitoksia.
Noodeilla on ominaisuuksia, jotka vastaanottavat informaatiota muista noodeista. Vastaanotettua liitosta kutsutaan input-liitännäksi ja lähtevää output-liitännäksi. Vaatimuksena on, että yhdisteltävät liitokset ovat datatyypiltään samanlaiset. Desimaalinumerot liittyvät desimaaliin ja niin edelleen (kuva 4.).



Kuva 4. Noodien välillä kulkevat nuolet kuvaavat riippuvaisliitoksia ja osoittavat komentojen ja informaatiovirran suunnan.

Nimensä mukaisesti riippuvaisliitoksissa noodien ominaisuudet tulevat riippuvaisiksi toisen noodin ominaisuuden arvoista. Yhden noodin tehdessä yhden toimituksen (laskun tai datan keruun), se lähettää informaation sopivan output portin kautta toisen noodin input-porttiin. Toinen noodi pystyy tällöin hyödyntämään saatua informaatiota ja viemään sitä eteenpäin.

Meshien mallinnuksessa tätä noodien välistä ketjua hyödynnetään paljonkin. Mallinnustyökalujen käyttö luo uuden mallinnusnoodin, joka vastaanottaa nykyisen meshin tiedot ja käsittelee ne, muodostaen uudenlaisen meshin. Tätä kutsutaan rekennushistoriaksi (construction history), jonka voi nähdä channel editorissa (kuva 5.).



Kuva 5. Kuvassa on merkattuna meshin rakennushistoria. Kuvan meshi koostuu useiden mallinnusnoodien input-liitännäisistä, jotka koostavat meshin. Objektin muut input ja output liitännät näkyvät channel editorissa samassa kohtaa ja niitä voidaan säädellä kyseistä liitäntää klikkamalla esiintyviä ominaisuuksia säätelemällä.

Rakennushistoria ei ole niin olennaisin seikka rigauksessa, mutta noodien välinen tiedonkulun ymmärtämisellä sitäkin enemmän. Useimmiten Maya luo liitoksia automaattisesti ilman, että käyttäjä edes huomaa sitä tai tarvitsee erikseen käsitellä liitoksia. On kuitenkin tilanteita, joissa omien riipuvaisliitosten tekeminen on hyödyllistä, jopa välttämätöntä.

Yksinkertaisin esimerkki on kahden objektin ominaisuuksien välinen liitos, jossa yhden objektin rotaatio ohjaa toisen objektin rotaatiota. Tämän liitoksen voi tehdä useilla tavoin. Käsittelen niitä noodien käsittelyn työkaluissa.

Riippuvaisliitoksessa ominaisuus, joka ohjaa toista ominaisuutta, kutsutaan ***ajuriksi*** (driver). Ominaisuus, jota taas ohjataan, kutsutaan ***ajettavaksi*** (driven).

Useat työkalut kuten constraint –liitokset toimivat dependency ominaisuuksien ja hierarkian yhteistyöllä. Kriittistä olisi pysyä perillä siitä, että objektien välinen informaatiokulku ei luo ristiriitoja. Ohjainsysteemeissä, joissa riippuvaisliitokset ja hierarkiat toimivat yhtenäisenä ryhmänä, on otettava huomioon hierarkian lineaarinen tiedonkulku ja riippuvaisliitosten syklinen kulku. Tärkeää on pitää huolta siitä, että riippuvaisliitokset pystyvät jakamaan informaatiota häiritsemättä hierarkian tiedonkulkua.

2.2.4 Transformaatio ja deformaatio

Tässä vaiheessa koen tarpeelliseksi korostaa vielä kahden termin, transformaation ja deformaation eron tässä tutkielmassa. Transformaatiolla tarkoitan kaikkia niitä ominaisuuksia, jotka vaikuttavat objektin transform noodiin, kuten translate ja rotate. Nämä ominaisuudet useimmiten limittyvät objektin navan (pivot) sijaintiin ja kallistuskulmaan (Autodesk 2013b.)

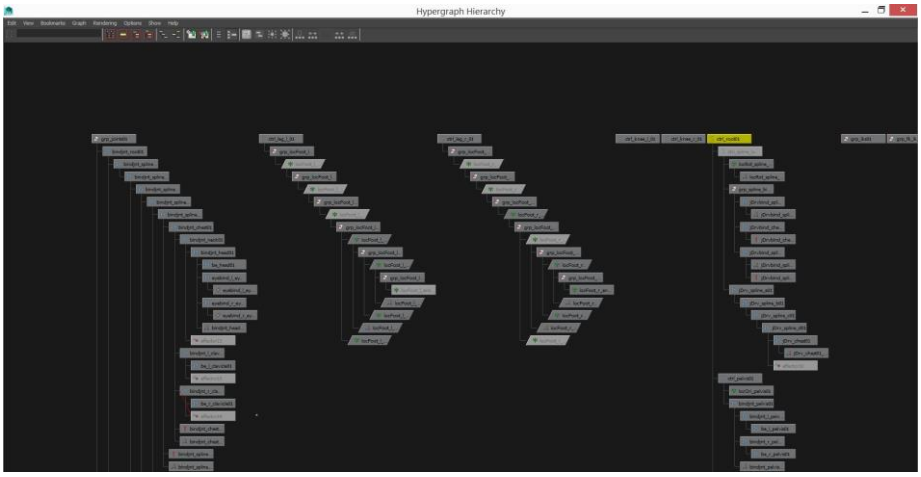
Deformaatio puolestaan tarkoittaa shapen muokkausta. Control vertexit ovat stabiileja komponentteja, joita ei itsessään voi animoida tai linkittää ilman tarvittavia deformaattoreita (käsittelen deformaattoreita luvussa 6.). Deformaatiossa näiden komponenttien sijainteja muutetaan ja siten meshin rakennetta muokataan toisenlaiseksi. Rigin pääasiallinen tehtävä onkin deformoida meshiä hallitusti ja siten, että sen muutoksille voi antaa selkeät arvot ja määreet.

2.3 Työkaluja noodien ja objektien hallintaan

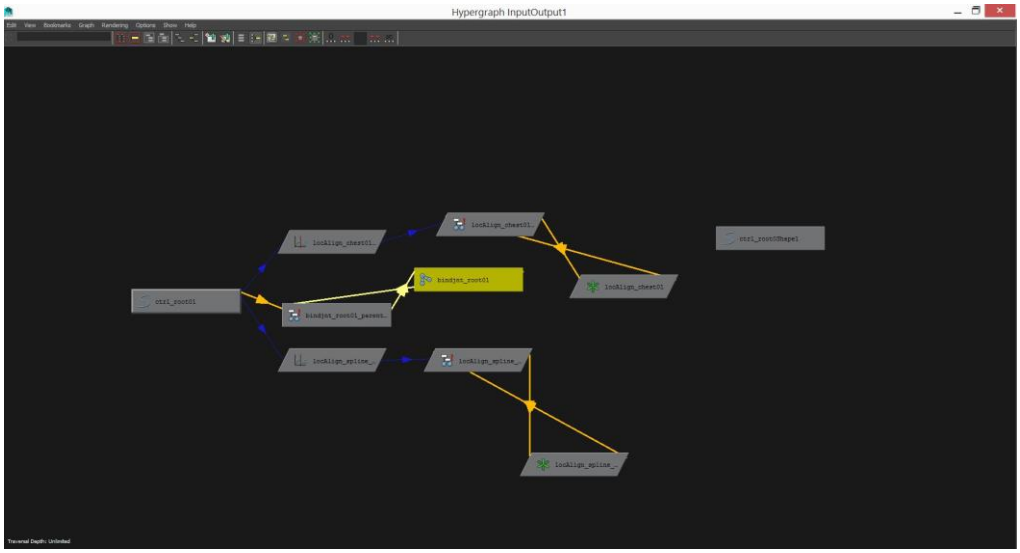
Mayan työkaluvalikoima on laaja. Yksi työtila voi kattaa suuren määrän objekteja ja noodeja. Näiden hallintaan on kuitenkin annettu runsaasti erilaisia työkaluja. En lähde tarkasti kuvailemaan sitä, miten nämä toimivat, mutta esitän yleiskatsauksen sille, mitä kyseiset työkalut tekevät.

2.3.1 Hypergraph

Hypergraph taulukkoa voi pitää node editorin edeltäjänä ja sillä on yhä tehtävänsä. Hypergraphin avulla on mahdollista tarkkailla valittuja objekteja, noodeja ja näiden valisiä liitoksia ja hallita näitä liitoksia node-editorin tapaan. Hypergraphilla on kaksi moodia, hierarkia ja dependency. Hierarkia (hierarchy) näyttää nimensä mukaisesti objektien ja noodien väliset hierarkialiitokset (kuva 6.). Dependency taas ilmaisee objektien ja noodien väliset riippuvaisliitokset (kuva 7.).



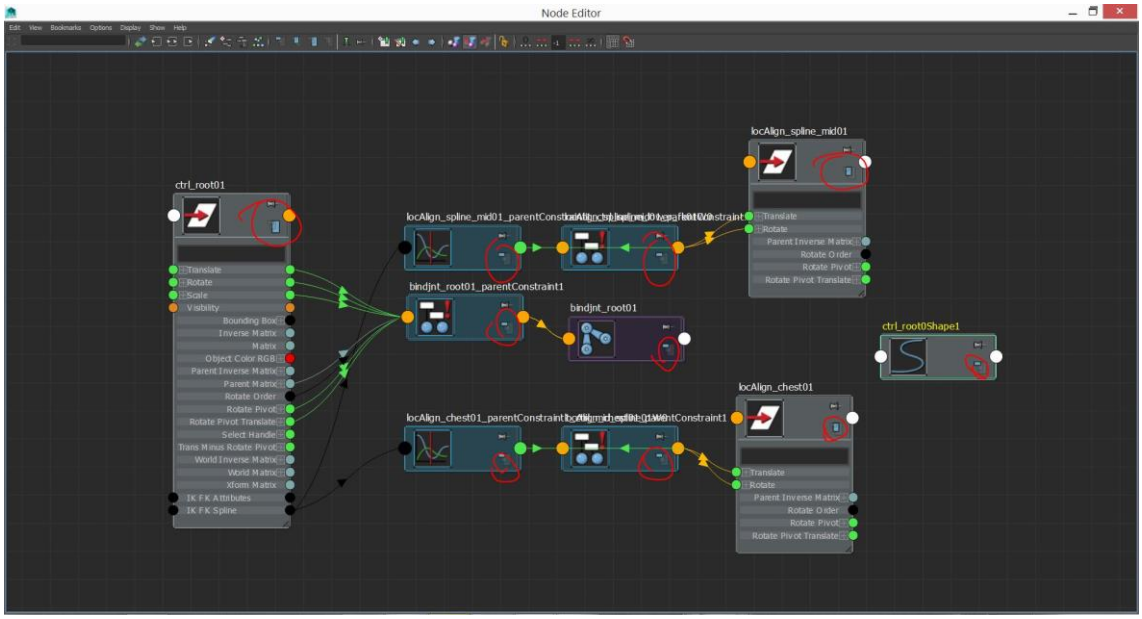
Kuva 6. Hypergraphin hierarkiamoodi esittää jokaisen scenessä esiintyvän objektin ja sen hierarkkisen liitoksen.



Kuva 7. Hypergraphin dependency-moodi. Huomaa, että osa liitoksista kulkee noodien välillä syklisesti kun hierarkkisessä näkymässä liitokset kulkivat vain lineaarisesti yhteen suuntaan.

2.3.2 Node editor

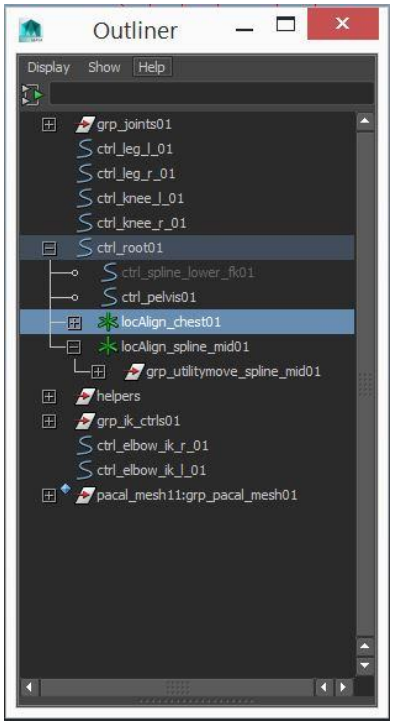
Node editor on tuorein työkalu-lisäys noodien käsittelyyn. Node editor poikkeaa hypergraphista sen vahvemman visuaalisen esityksen vuoksi. Node editorissa noodit ovat esillä visuaalisina artikkeleina. Jokaisella artikkelilla on kolme muotoa. Ensimmäinen sisältää vain noodin nimen ja summatut input-output liitokset. Toisessa moodissa osa noodin ominaisuuksista on näkyvissä. Kolmannessa moodissa jokainen ominaisuus ja niiden portit ovat näkyvissä (kuva 8.).



Kuva 8. Vasemmalla oleva noodi on kokonaan avatussa muodossa esittäen graafisesti jokaisen portin erikseen. Keskellä olevat noodit ovat suljettuja noodeja ja oikeanpuoleisimmat puoliksi avattuja. Ympyröidyt pisteet kuvaavat kytkintä, joka määrittää noodin avoimuuden.

2.3.3 Outliner

Outlinerin avulla on mahdollista tarkastella kaikkia skenessä esiintyviä objekteja ja noodeja sekä niiden keskeisiä hierarkioita. Kun rigin osasten määrä käy hyvin suureksi, outlinerista tulee lähestulkoon välttämätön työkalu. Hierarkioissa child on listattuna parentin alle ja listauksen voi tarvittaessa minimoida (kuva 9.).

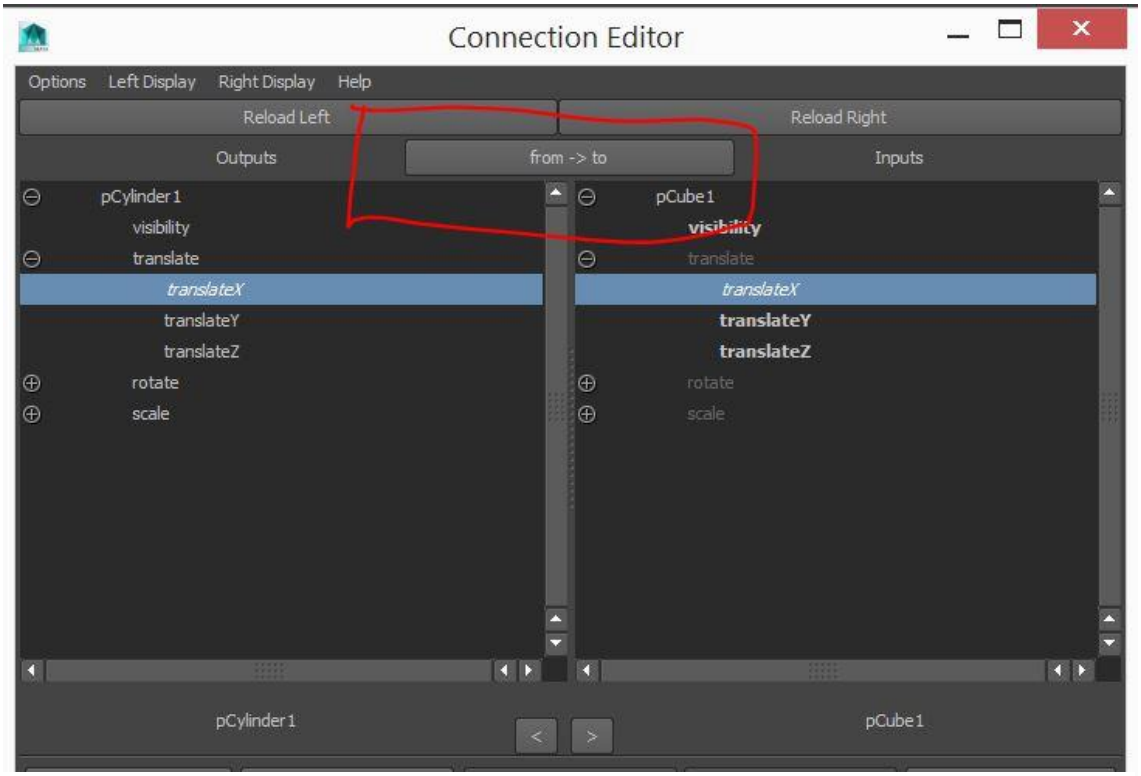


Kuva 9. Outlinerin perusnäkymä. Objektit, joiden vasemmalla puolella näkyy "+"-merkki, omaavat hierarkkisesti alaisuudessaan olevia objekteja. Objekteja kuvaavat ikonit ovat samoja kuin node editorissa ja hypergraphissa.

Outlinerissa objektien listauksessa on vaihtoehto DAG-objects only. Tämä tarkoittaa, että outliner listaa ainoastaan ne objektit, jotka muodostuvat DAG-hierarkiatyypillä. Tämän lisäksi on mahdollista saada shape-noodit näkymään erikseen outlinerin näkymässä. Tällöin shape-noodit ovat aina hierarkiassa transform noodin alla.

2.3.4 Connection editor

Connection editor on yksi keino luoda dependency liitoksia kahden noodin välille. Connection editorin löytää windows>general editors linkin kautta (kuva 10.).

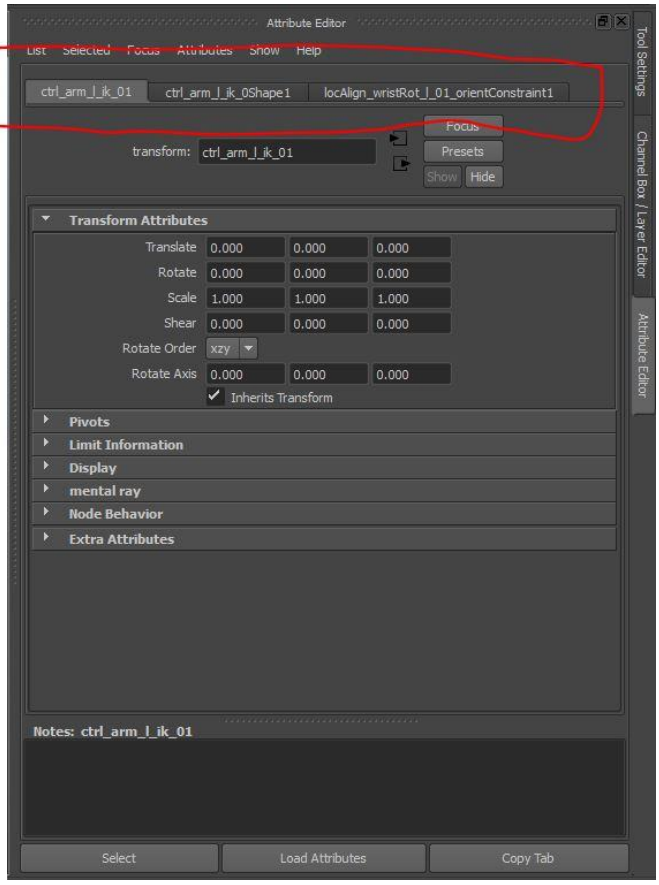


Kuva 8. Ympyröity näppäin osoittaa kumpaan suuntaan komento tapahtuu. Valitut ominaisuudet ovat keskenään liitoksessa. Kun ominaisuus on liitettynä, ominaisuus näkyy kursivoituna.

Connection editorissa on mahdollista asettaa ohjaava objekti (ajuri) ja ohjattava objekti (ajettava) jompaankumpaan sarakkeista ja päättää kumman objektin ominaisuus toimii ajurina ja kumman ajettavana. Sarakkeissa on listattuna objektien ominaisuudet. ”Left- ja right-display” –ohjainten avulla on mahdollista säätää esitettävien ominaisuuksien määrää. Normaalisti näkyvissä on vain ominaisuudet, jotka ovat käytössä channel editorissa, mutta on myös mahdollista tuoda esiin ominaisuudet, jotka ovat kätkettyinä tai, jotka eivät normaalisti ole animoitavissa (keyable).
Ensin valitaan ominaisuus, jonka käyttäjä haluaa toimivan ajurina, ja sitten toisesta sarakkeesta ominaisuus, johon ominaisuus vaikuttaa. Liitetyt ominaisuudet esiintyvät kursivoituina molemmissa sarakkeissa. Tämä liitos näkyy node- ja hypergraph ikkunoissa nuolena kahden noodin välillä.

2.3.5 Attribute editor

Attribute editorin kautta on mahdollista säätää valitun objektin kaikkia erilaisia ominaisuuksia. Valittaessa objektia, attribute editorin ikkunan ylälaidassa näkyvät erikseen noodit, jotka koostavat objektin, kuten transform- ja shapenoodin. Myös objektit, jotka osaltaan ovat hierarkioitu objektin alaisuuteen tai ovat jollain lailla liitoksissa objektiin, näkyvät editorissa.



Kuva 9. Attribute editor mahdollistaa objektin sisältämien ominaisuuksien hallinnan. Ylälaidassa ovat ympyröitynä sarakkeet objekteille, jotka ovat hierarkkisessa liitoksessa objektin kanssa.

Näillä objekteilla itsellääänkin ovat omat ominaisuutensa.

Attribute editorissa esiintyvät ominaisuudet ovat kaikki vapaasti animoitavissa. Riittää, että ominaisuutta klikataan hiiren oikealla ja valitaan “set key”.

2.3.6 Channel editor

Useimmat mayaa käyttäneet ovat jo varmasti olleet tekemisissä channel editorin kautta. Channel editor voi sisältää käytännössä kaikki samat ominaisuudet kuin attribute editor,

mutta toisin kuin attribute editorissa, channel editorissa nämä ominaisuudet ovat animoitavissa.

Channel editorin kautta on myös mahdollista nähdä objektiin liittyvät input ja output liitokset. Objektin output liitoksiin liitettyjen noodien ominaisuuksia on mahdollista säädellä channel editorin kautta. Objektiin kertyvä mallinnushistoria tallentuu myös channel editoriin.

2.3.7 Component editor

Component editor mahdollistaa tarkan hallinnan niiden komponenttien hallinnalle, jotka eivät ole transform noodeja. Useimmiten tämä tarkoittaa vertexejä. Component editorin avulla on mahdollista määrittää esimerkiksi valittujen vertexien tarkka sijainti. Sen lisäksi on mahdollista selvittää ja muuttaa erilaisten deformereiden painoarvoja. Tämä on kätevä työkalu kun on tarvetta säädellä esimerkiksi sitä, kuinka paljon yksi cluster vaikuttaa control vertexiin.

3 Hierarkkiset liitokset – Nivelet (joints)

Vaikka tarkoituksenani on käsitellä tässä luvussa nivelien (joint) käsittelyyn liittyviä asioita, pääpointtina ovat erityisesti hierarkioiden vaikutukset objektien ominaisuuksien arvoihin. Hierarkiat ovat ensimmäisiä liitoksia, joiden kanssa Mayan käyttäjä joutuu todenteolla tekemisiin.

Se miten hierarkiat vaikuttavat objektien ominaisuuksiin voi kuitenkin aiheuttaa monimutkaisemmissa rigeissä jo päänvaivaa. Hierarkkisen liitoksen osapuolten toisiinsa vaikuttavat ominaisuudet ja sijaintien määreet ovat konsepteja, jotka ovat kriittisiä Mayan toimintamekanismien ymmärtämisen kannalta.

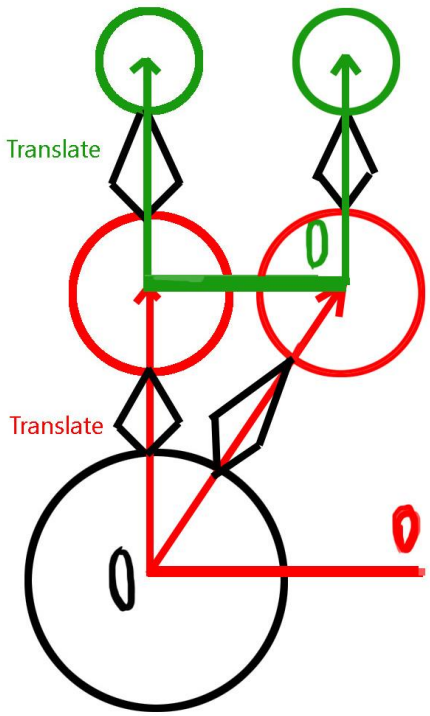
3.1 Hierarkiat

Kun puhutaan hierarkioista, Mayassa se tarkoittaa objekteja, jotka on asetettu toisen objektin alaisuuteen. Tästä käytetään useimmiten nimikettä parent-child liitos. Parent-objekti on hierarkian ylempi osa, ja child parentin alainen. Myös childilla voi olla oma child ja niin edelleen. Useat peräkkäiset hierarkkiset-liitokset muodostavat hierarkiaket-

jun. Tällaisen ketjun ylintä jäsentä voidaan kutsua myös rootiksi. Englanniksi hierarkkisen-liitoksen tekemistä kutsutaan parentoinniksi (parenting), mutta tässä tutkielmassa käytän enimmäkseen termiä hierarkiointi.

Tutkinto-ohjaajani Jaro Lehtonen ilmaisi hierarkiaketjujen toiminnan varsin selkeällä tavalla. Kaikki transformaatiot hierarkiassa etenevät ylhäältä alaspäin. Hierarkian alimman objektin suunta, paikka ja skaalaus mitataan sen mukaan, millaiset sen ylemmät transformaatioarvot ovat. Jokaisen hierarkiaketjun translaatio, rotaatio ja skaalaus mitataan hierarkian yläpäästä alapäähän. Hierarkiassa objektin parent toimii objektin nollakohtana niin rotaatio- kuin translaatioarvoissa.
Hierarkiat ovat yksi mayan rakenteen kivijaloista. Koko Mayan käyttöjärjestelmä pohjautuu sille, että sen objektit ja noodit toimivat hierarkioissa ja liitoksissa keskenään. Mielenkiintoisena yksityiskohtana mainittakoon, että objekti, jota ei ole hierarkioitu minkään toisen objektin alaisuuteen on yhä niin sanotusti hierarkkisessa liitoksessa työtilaan.

Hierarkkisissa liitoksissa parentista tulee useimmiten objektin nollakohta, johon nähden objektin ominaisuuksissa tapahtuva muutokset mitataan. Esimerkiksi objektin translaatio mitataan ennen kaikkea parentin sijainnista (kuva 11.).



Kuva 11. Vihreä ympyrä on hierarkkisesti punaisen ympyrän alainen, punainen taas mustan ympyrän alainen.

Punainen viiva kuvaa punaisen ympyrän nollapositiota ja nuoli sen translaation suuntaa. Punaisen ympyrän translate-arvo saadaan punaisen ympyrän navan nykyisen sijainnin ja nollakohdan erotuksesta.

Vihreä viiva on vihreän ympyrän nollakohta ja nuoli osoittaa vihreän ympyrän translaation. Vihreän ympyrän translate-arvo määrittyy vihreän ympyrän nollakohdasta.

Huomioitava piirre on, että punaisen ympyrän nollakohta on samassa pisteessä kuin missä musta ympyrä sijaitsee. Koska punainen ympyrä on hierarkioitu mustan ympyrän alaisuuteen, punaisen ympyrän nollakohta seuraa mustan ympyrän sijaintia.

Vihreän ympyrän nollakohta puolestaan on samassa pisteessä kuin punaisen ympyrän sijainti. Samat periaatteet pätevät kuin punaisen ympyrän ja mustan ympyrän tapauksessa.

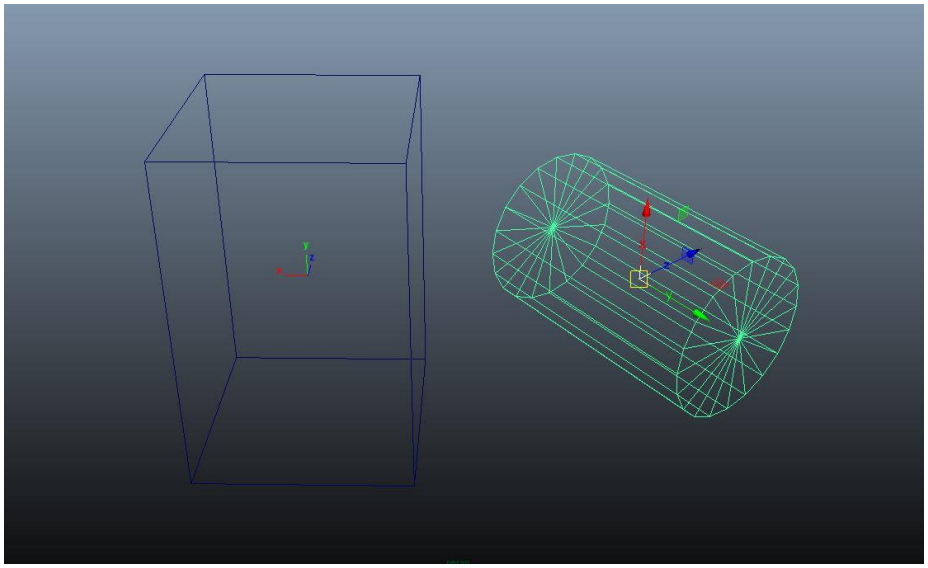
Jos punaista ympyrää siirretään sivummalle, se vaikuttaa vain punaisen translaation määrään, mutta ei muuta vihreän ympyrän translaatiota.

Galaksi-järjestelmä on yksi esimerkki reaalimaailman hierarkkisesta liittymästä. Aurinkokuntamme niin sanottu root-objekti on itse aurinko. Ympäröivät planeetat taas ovat hierarkkisesti liitettynä sen alaisuuteen. Tellus on yksi aurinkoon hierarkioitu objekti. Kuu puolestaan on hierarkkisessa liitoksessa Telluksen kanssa. Kuu seuraa täten Telluksen liikettä, mutta seuraa siten myös aurinkoa, mutta Telluksen alaisuudessa.

3.1.1 Hierarkia ja objektin napa (pivot) ja tila

Napa tarkoittaa tässä tapauksessa objektin rotaation (tai skaalauksen) keskikohtaa. Napa on käytännössä myös transform-noodissa se piste, johon nähden etäisyydet objektien välillä määritetään. Channel editorissa ja attribute editorissa nähtävä translate-ominaisuus tarkoittaa navan sijainti navan nollakohtaan nähden. Transform-noodien välisissä hierarkioissa suhteet ovat siis rakentuneet objektien napojen välille.

Objektin tilalla (object space) tarkoitetaan objektin henkilökohtaista navan sijaintia ja sitä, miten objektin rotaatioakselit suuntautuvat (kuva 12.). Transformaatiot, joita objektille tapahtuu, tapahtuvat objektin henkilökohtaisessa tilassa. Objektin henkilökohtaisen transformaation määreet mitataan sen lokaaliin navan sijaintiin nähden. Lokaalin navan sijainti mitataan hierarkiaketjuissa objektin parentin navasta



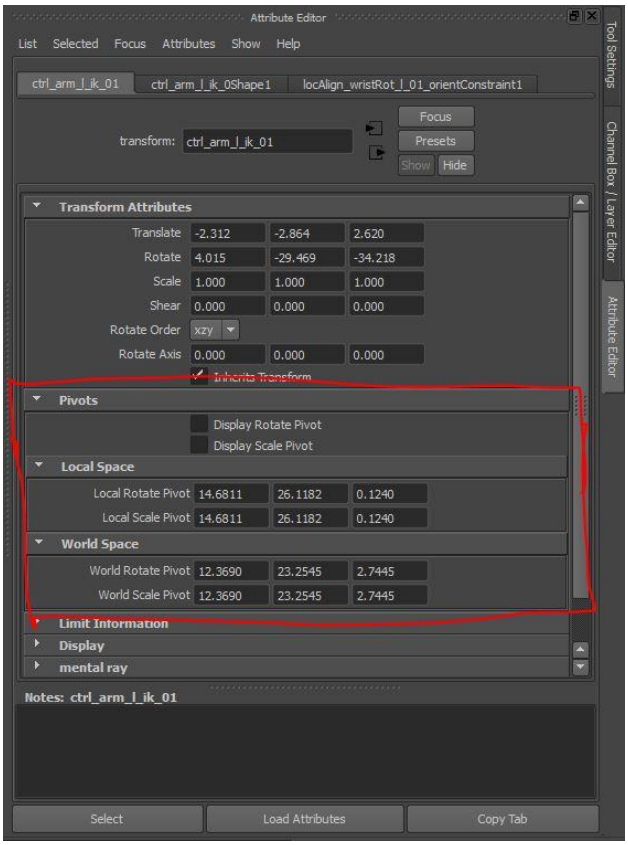
Kuva 12. Kuution keskellä näkyy kuution akselisto, joka on asettautunut globaalin akseliston mukaan. Sylinteriä puolestaan on käännetty niin, että sen henkilökohtainen akselisto osoittaa eri suuntaan.

Globaalin akseliston suunta ja nollapiste on aina sama.

Objektia liikuttava kursori voi liikuttaa objektia suhdanteessa eri tiloihin. Kursori voi esimerkiksi liikuttaa objektia lokaalin tilan mukaan (local space). Tässä tilassa kursorin suunta määräytyy objektin navan mukaan. Maailman mitassa (world space) objektin rotaatio ja translaatio määräytyvät työtilan absoluuttisen akseliston mukaan. Yllä olevassa kuvassa objektin translaatio-työkalu määräytyy objektin navan mukaisesti. Jos translaatiotyökalu määräytyisi työtilan akseliston mukaan, sen akseliston suuntimat olisivat samat kuin kuution keskellä näkyvän globaalin akseliston. Objektin liikuttavan kursorin suuntautumista voi muuttaa painamalla shift- ja ctrl-näppäimiä ja hiiren oikeaa samaan aikaan kursorin yllä.

3.1.2 Local vs. world-space

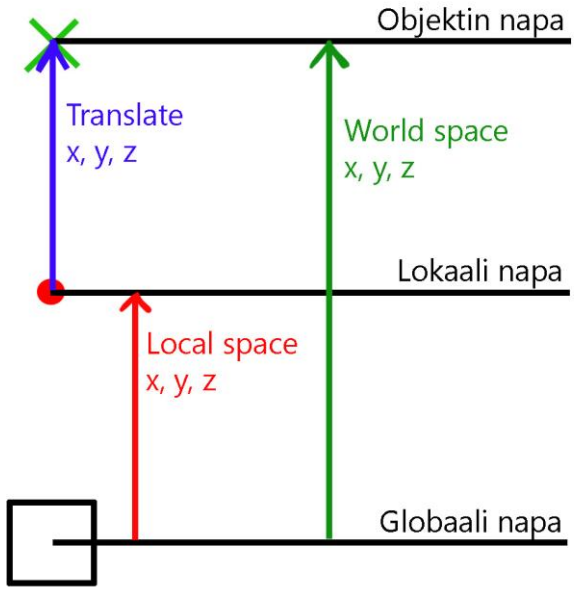
Objekteilla on kaksi ominaisuutta, jotka voivat määrittää objektin sijainnin työtilassa, lokaali ja globaali napa (local- and world-pivot). Nämä ominaisuudet eivät ole nähtävissä channel editorissa, mutta niitä pystyy säätelemään attribute editorin kautta (kuva 13.). Lokaali ja globaali napa eivät niinkään ole erillisiä napoja, vaan kaksi eri pistettä sille, mihin nähden objektin translate-ominaisuus mitataan. Lokaali navan sijainti mittaa objektin navan sijainnin tämän parentiin nähden. Globaali navan sijainti taas mittaa navan sijainnin työtilan nollakohtaan nähden. Channel editorin kautta käsiteltävä translate-ominaisuus mitataan henkilökohtaisen navan sijaintiin nähden.



Kuva 13. Ympyröidyt ominaisuudet kuvaavat lokaalin ja globaalin navan sijaintia xyz-akseleilla. World space mittaa objektin sijainnin globaalissa mittakaavassa. Lokaali puolestaan mittaa objektin lokaalin navan sijainnin globaalissa tilassa. Huomaa, että objektin globaali sijainti on yhtä kuin objektin translaatio + lokaalin navan sijainti.

Navan lokaalia sijaintia pystyy käsittelemään erikseen. Tällöin objektin shape-noodi itsessään ei liiku, mutta navan paikka vaihtuu. Rigin ohjainbjektien on oltava useimmiten transformaatioiltaan nollattuja. Tämä ei kuitenkaan tarkoita, että objektit olisivat arvoiltaan nollattuja globaalissa mittakaavassa, mutta ne ovat nollattuja lokaaliin navan sijaintiin nähden. Normaalisti lokaalin navan sijainti asettautuu sen mukaan, missä objektin parent sijaitsee.

Konsepti on hivenen hankala käsittää teoriassa. Suosittelen, että lukija kokeilee tätä käytännössä kahdella polygoniobjektilla. Parentoi toinen objekti toiseen ja tarkkaile attribute editorissa, miten objektin local space ja world space ominaisuudet päivittyvät objekteja liikuteltaessa. Kuva nro. 14 esittää vielä tiivistettynä määreet sille, miten objektin sijainnit määritetään.



Kuva 14. Tämä kaavio kuvastaa sitä, missä suhdanteessa ominaisuudet liikkuvat.

Neliö on globaali nollapiste.

Punainen piste on objektin lokaali napa.

Vihreä rasti kuvastaa objektia ja sen henkilökohtaista napaa. Tämä napa on lopulta se, jota työtilassa käsitellään.

Nuolet merkitsevät etäisyyksiä suhteessa eri pisteisiin.

Vihreä nuoli kuvaa objektin etäisyyden globaalista nollapisteestä.

Punainen nuoli kuvaa lokaalin navan etäisyyttä parentista (ellei objektia ole hierarkioitu toiseen objektiin, objekti on hierarkkisessa liitoksessa työtilaan).

Sininen nuoli kuvaa objektin translaatiota. Tämä on se etäisyys, jota käsitellään channel editorin kautta. Translate mitataan lokaalista navasta.

3.1.3 Orientaatio:

Lokaalin ja globaalin navan sijainti ovat molemmat kriittisiä konsepteja translate-ominaisuuden käsittelyyn. Navan toinen tehtävä on kuitenkin toimia objektin rotaation keskipisteenä. Aivan kuten objektin translaation nollakohtaa on mahdollista säädellä erikseen, niin myös navan rotaation nollapistettä on mahdollista säädellä erikseen.

Normaaleissa transform-noodeissa on erillinen ominaisuus ”rotate axis”, joka mahdollistaa objektin rotaatioakseleiden suunnan muutoksen. Tällöin rotaatioakseleiden suunta muuttuu, mutta rotaatio pysyy nollassa.

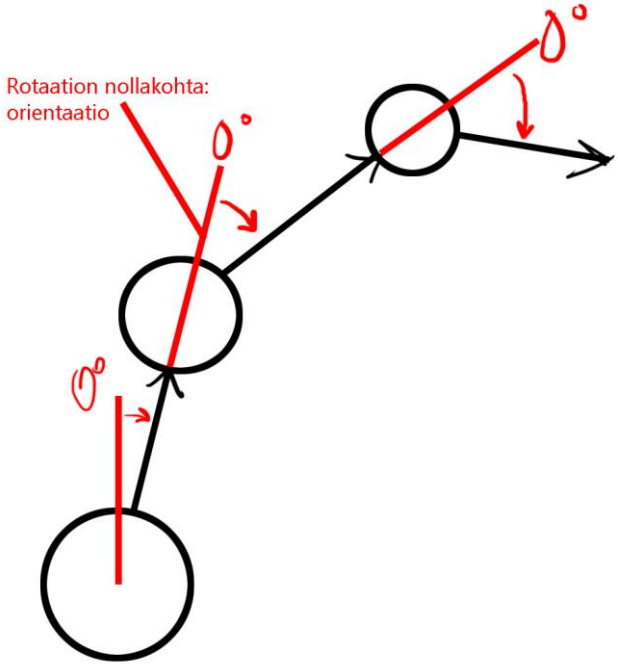
3.1.4 Rotaatio vs. orientaatio

Rotaatio on yksi transform noodin perusominaisuus, joka mahdollistaa objektin kääntämisen eri akseleilla. Rotaatioarvo riippuu siitä, kuinka paljon objektia on käännetty nähden sen rotaation nollakohdasta.

Objektin rotaation nollakohdan suuntaa on kuitenkin mahdollista muuttaa. Jos rotaatio on 90 astetta, se tarkoittaa sitä, että objekti on kääntynyt 90 astetta nollakohtaansa nähden. Jos nyt käännämme itse objektin rotaation nollakohtaa 90 astetta samaan suuntaan, objektin rotaatio on jälleen nolla, koska rotaatio osoittaa samaan suuntaan kuin nollakohta.

Nollakohdan suuntaa kutsutaan Mayassa orientaatioksi. Rotaatiolla taas tarkoitetaan objektin kääntymistä nollakohtaan nähden.

Hierarkiassa orientaatioon pätee samat säännöt kuin translaatiossa. Parent määrittää objektin nollatun orientaation suunnan (kuva).



Kuva 15. Kuva esittää objektien orientaation ja rotaation määrittymiset hierarkiassa.

Ylin ympyrä on hierarkioitu keskimmäisen alle. Keskimmäisin ympyrä on hierarkioitu alimman ympyrän alle.

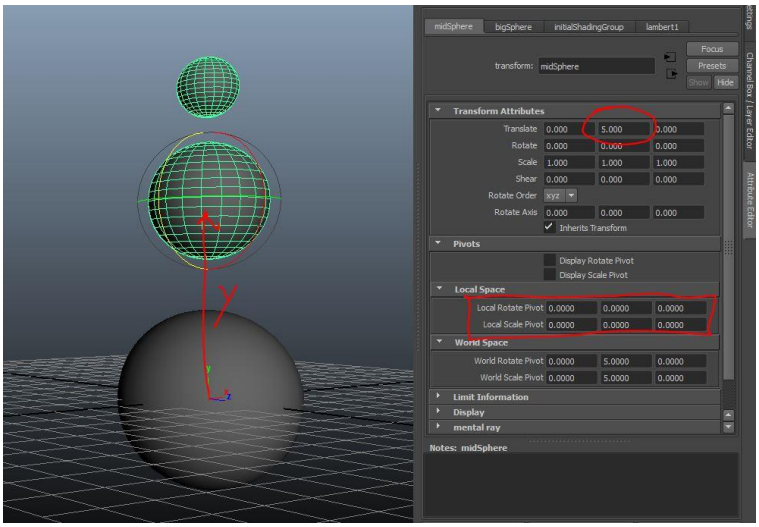
Punaiset viivat kuvaavat yksinkertaistetusti rotaation nollakohdan, orientaation, suuntaa. Oletuksena on, että objektien orientaatioarvot ovat nollattuja.

Musta nuoli kuvaa objektin rotaation suuntaa nollakohdasta katsottuna.

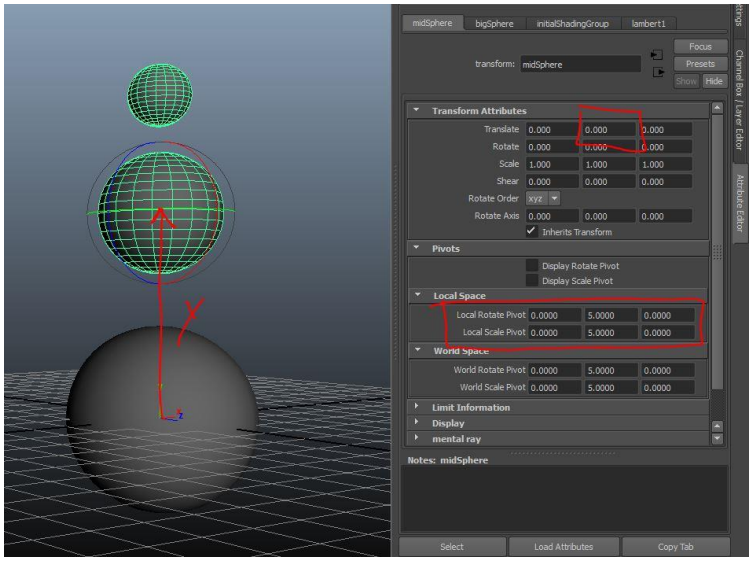
Kuvasta näkyy, että hierarkiassa child omaksuu orientaation nollakohdaksi parentin rotaation suunnan ja edelleen child-objektin child omaksuu oman parentinsa rotaation orientaationsa nollakohdaksi.

Jokaisen child-objektin rotaatio määrittyy tämän oman orientaationsa suunnasta.

3.1.5 Transformaation nollaus (freeze)



Puhuin jo aiemmin objektin ominaisuuksien nollauksesta. Mayassa tästä käytetään nimitystä jäädytys (freeze). Tämä prosessi nollaa kaikki objektin arvot, mutta ei muuta itse objektin sijaintia työtilassa. Jäädytysprosessissa objektin lokaalin navan sijainti siirtyy siihen pisteeseen, jossa objektin napa sijaitsee globaalin navan mukaan.



Kuva 16. Yllä olevassa kuvassa on ympyröity ei-nollattu translaatioarvo. Yläpuolisen kuvan lokaalin navan arvot ovat myös nollassa. Lokaali napa sijaitsee siis samassa positiossa kuin globaali napa.

Alapuolisessa kuvassa saman objektin ominaisuudet on jäädytetty. Translate arvo on nollassa, mutta vastaava lokaali arvo on omaksunut translaation arvon.

Esimerkiksi globaalin navan mukaan objekti sijaitsee xyz-akselilla pisteessä: 2. 3,1. 5. Objektin lokaali napa taas sijaitsee pisteessä: 1. 1,55. 2,5. Objektin translaatioarvo on siis globaalin navan ja lokaalin navan erotus. Jäädytysprosessissa lokaalin navan sijainti muuttuu siten, että lokaalin navan sijainnista tulee sama kuin globaalin navan sijainnista (kuva 16.).

Jäädytysprosessissa on tärkeää huomioida näiden piilevien arvojen muutos, sillä objektin transformaation muutos saattaa heijastua objektiin hierarkioitujen ja liitettyjen objektien arvoihin.

3.1.6 Napojen siirtely ja objektien täsmällinen järjestely (snapping)

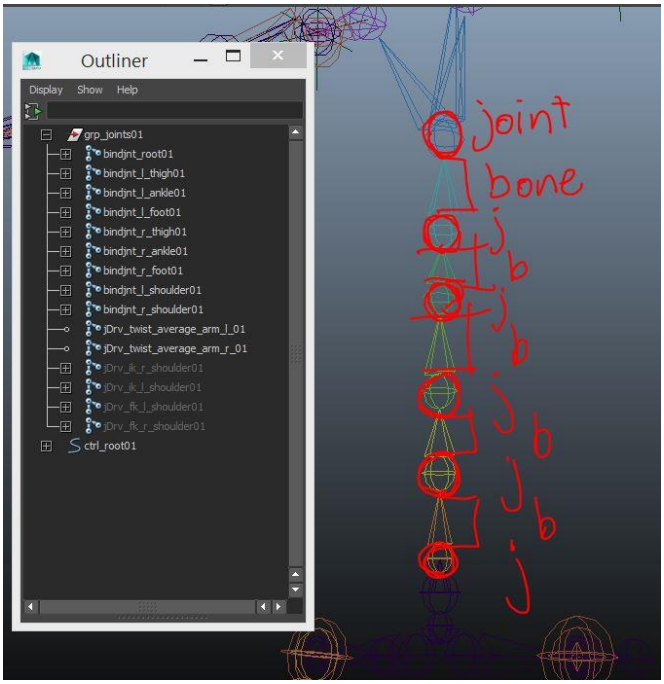
Mayassa on kätevänä työkaluna täsmällisen siirron työkalut. Erilaisina mahdollisuuksina ovat esimerkiksi objektien liikuttelu siten, että napa kulkee täsmälleen työtilan ristikolla (x näppäin pohjassa) tai asettuu täsmällisesti toisen pisteen kohdille (v-näppäin pohjassa). Nämä ovat käteviä niin rigauksessa kuin mallinnuksessakin.

Tämän lisäksi on mahdollista liikuttaa itse objektin napaa, muuttamatta objektin sijaintia tai transformaatiota (d-näppäin painettuna). Käytännössä tässä prosessissa muutetaan transform-noodin lokaalin ja globaalin navan arvoja, translate ominaisuuksien pysyessä samoina.

3.2 Nivelet (joints)

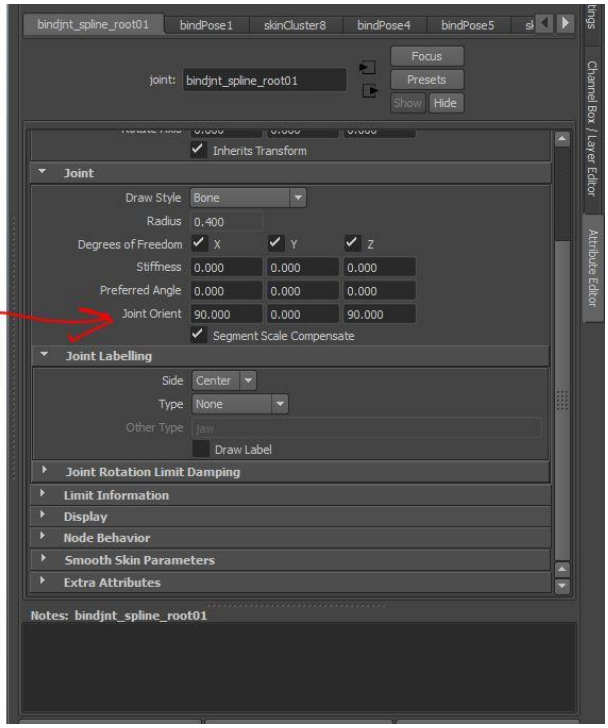
Nivelet ovat rigien perusraaka-aineita, joiden muodostamaan järjestelmään mesh lopulta sidotaan (skinning). Niillä on samat perusominaisuudet kuin transform-noodilla, mutta poikkeuksia piilee sellaisten ominaisuuksien kuin lokaalin navan ja orientaation suhteen. Nivel on oma nooditYyppinsä, jolla ei ole omaa shape noodia. Tällöin se ei näy rendatussa kuvassa.

Kaikki edellä mainitut asiat lokaaleista ja globaaleista navoista liittyvät vahvasti niveliin (jointteihin). Tavallisten luiden tapaan nivelet voidaan asetella hierarkkiseen järjestykseen muodostaen tällöin nivel-ketjun (joint-chain). Kun kaksi niveltä asetellaan hierarkiaan, parent-nivelen ja child-nivelen väliin muodostuu nuolen kaltainen kursori, joka osoittaa parentista childiin. Tämä kursori on itse luu (bone), mutta luulla ei ole omaa noodia ja niillä ei täten itsessään ole mitään muuta tehtävää kuin osoittaa nivelhierarkian suunta ja visualisoida luun pituus (kuva 17.).



Kuva 17. Ympyröidyt objektit ovat varsinaisia niveliä, jotka sisältävät nivelen rotaation, orientaation ja sijainnin. Kahden nivelen välissä oleva nuoli on ”luu”, joka itsessään ei tee muuta kuin osoittaa nivelen suunnan.

Nivelillä on sisäänrakennettu orientaatio ja hierarkiajärjestelmä (kuva 18.). Nivelten translate-arvot mitataan aina suoraan globaalista keskipisteestä tai parentista katsottuna, eikä niitä siten voi nollata. Hierarkioidun nivelen translate-arvo ilmaisee luun pituuden. Nivelillä on käytössään rotaatio-ominaisuudet, mutta toisin kuin objekteissa, jäädytys ei orientoi objekti akselistoa globaalin tai parentin navan mukaiseksi. Sen sijaan rotaatioiden arvot siirtyvät sellaisinaan nivelen orientaatioarvoihin. Nivelen orientaatioita ei voi nollata vaan ne mitataan aina globaalin akseliston tai parentin mukaan. Kun child-nivelen orientaatio nollataan, se osoittaa täsmälleen samaan suuntaan kuin nivel, johon se on orientoitu.



Kuva 18. Nivelillä on omat erilliset ominaisuutensa käytössään. Nivel-orientaation tulisi olla ensisijainen säädin nivelen asennolle, jotta nivelen rotaatiot säilyisivät nollana.

Nivelen voi säätää kääntymään pallonivelen tavoin jokaisella akselilla tai vain kahdella tai kolmella.

3.3 Huomioita luurankomallin rakennukseen

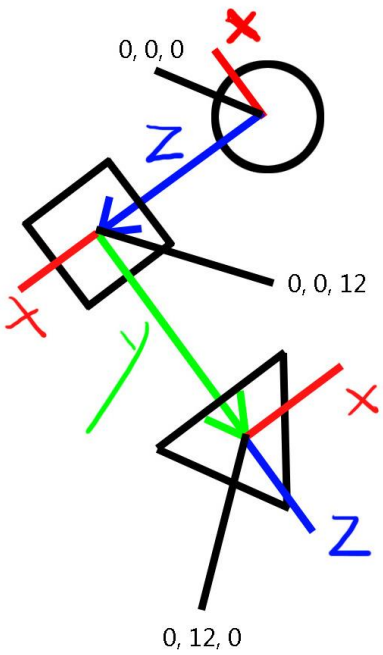
Tässä tutkielmassa pääasiallinen huomioni keskittyy teknisiin yksityiskohtiin Maya-rigin eri osien toiminnassa. Tutkielmani tarkoituksena ei siis ole niinkään esittää listauksia erilaisista tekniikoista, joilla näitä voidaan hallita. Koen kuitenkin olennaiseksi tehdä poikkeuksen nivelien kohdilla.

Meshin sidontaan tarkoitettu luusto valmistaa pohjan koko rigille, joten nivelien oikeanlainen asettelu on tärkeää ja heijastuu ohjainten toimintaan ja asetteluun. Luustotasolla olevat virheet voivat olla hyvin vaikeita korjata ja vaatia pitkällistä rigin purkua ja uudelleen rakennusta. Pienetkin ylimääräiset transformaatiot saattavat aiheuttaa kasaantumia, jotka ennen pitkää aiheuttavat vääränlaisia transformaatioita (Rigging Dojo, 2014).

Seuraavat nyrkkisäännöt voivat kuitenkin pelastaa paljon, ja vähentävät riskejä rigin virheellisille rakenteille (Rigging Dojo, 2014.)

Orientaatio: Kaikkien luiden rotaation on oltava nolla. Jos luita täytyy kääntää, niin tulee ennen kaikkea säätää niiden orientaatiota. Jokaisella nivelellä saman akselin tulisi aina osoittaa hierarkian seuraavaa niveltä. Mayassa useimmiten x-akseli osoittaa perusasetuksena kohti hierarkiaketjun seuraavaa niveltä. Tämän perusasetuksen voi halutessa muuttaa, mutta useimmiten sille ei ole tarvetta. Y- ja z-akselien suunta ei ole yhtä kiveen hakattu. Toivottavaa olisi, että yhden ketjun akselistot osoittaisivat kukin samaan suuntaan.

Toinen suositeltava seikka orientaatioiden tapauksessa on pyrkiä child-nivelien kohdalla pitämään vain yksi orientaatioakseli ei-nollattuna. Tämä ei tietenkään ole aina mahdollista esimerkiksi hierarkian ylimmän nivelen tapauksessa tai uuden nivelketjun alkaessa. Mahdollisimman yhtenäinen orientaatio on helpompi hallita ja rotaatioakselistot käyttäytyvät ennakoidummin (kuva 19.).



Kuva 19. Kuvassa esiintyvien objektien orientaatiot näkyvät värillisinä viivoina. Nuolet osoittavat sen miltä akselilta katsottuna objektia on liikutettu. Kolmio on hierarkioitu neliön alaisuuteen, neliö ympyrän alaisuuteen.

Mustat luvut esittävät objektin sijainnin parentiinsa nähden. Neliö on kulkeutunut 12 yksikköä ympyrän z-akseliin nähden.

Kolmio on kulkenut 12-yksikköä neliön y-akseliin nähden.

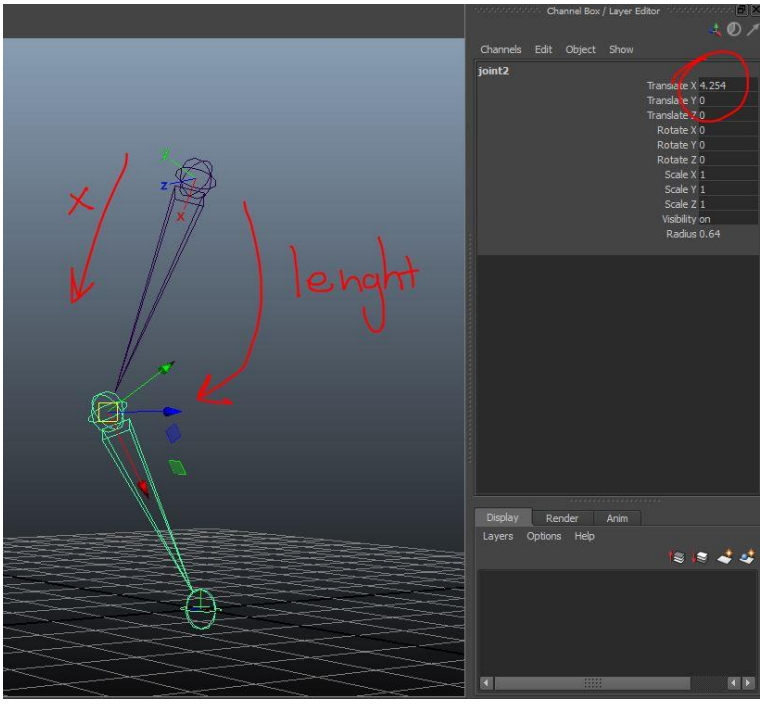
Ongelma muodostuu siitä, että jokaisen objektin akselistot osoittavat eri suuntiin. Jos jokaista objektia kääntäisi yhdellä rotaatioakselilla, niin jokainen objekti kääntyisi eri suuntaan.

Objektien orientaatioiden poikkeamat johtuvat siitä, että jokaisella objektilla enemmän kuin yksi orientaatioakseli on ei-nollattuna.

Translate arvot: Jokaisessa child-nivelessä tulisi olla vain yksi ei-nollattu translaatioarvo (katso jälleen kuva). Useimmissa tapauksissa se on se arvo, jonka akseli osoittaa seuraavaan niveleen. Normaaleissa tapauksessa se on x-translaatio, sillä se on useimmiten nivelien perusasetuksena. Useimmissa tapauksissa perusasetukset sille, mitkä akselit osoittavat seuraavaan niveleen, ei ole merkitystä. Tärkeintä on, että yksittäisessä projektissa seuraavaan niveleen

osoittava akseli on aina sama. Nivelhierarkian korkein nivel voi omata useampia translaatioita (Rigging Dojo). Tässä tutkielmassa seuraavaan niveleen osoittava akseli on x-akseli.

X-translaatio määräytyy sen mukaan mihin nivelen parent osoittaa. Parentin x-akselin on osoitettava täsmälleen child-nivelen suuntaan. Jos muistamme lokaalien napojen tapauksesta sen, miten child mittaa etäisyytensä parentin sijainnin ja akseliston mukaan. Jotta child-nivelellä voi olla translaatiota vain x-akselilla, se tarkoittaa, että childin täytyy poiketa parentin sijainnista vain x-akselin suunnassa (kuva 20.). Parentin x-akselin täytyy siis osoittaa täsmällisesti child-nivelen suuntaan. Jos child-nivelellä on enemmän kuin yksi ei-nollattu arvo translaatiossa, se tarkoittaa että parentin x-akseli ei osoita täsmällisesti childia kohti.



Kuva 20. Valitun nivelen sijainti parentiinsa nähden näkyy channel editorissa ympyröitynä. Tämä luku osoittaa samalla parentin ja childin välisen "luun" pituuden. Huomaa myös, että valittu nivel osoittaa myös omaan child-niveleensä.

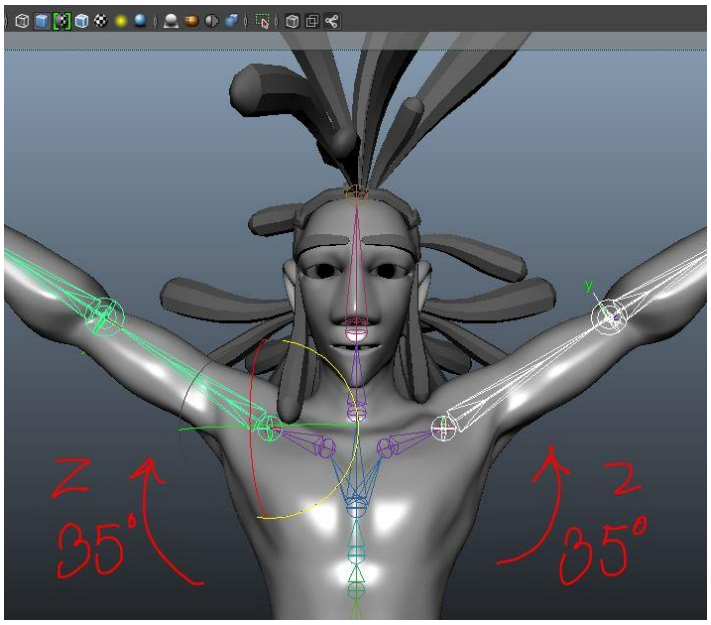
Syitä sille, miksi vain yhden translaation tulisi olla käytössä, on useita. Ensisijainen syy on se, että translaatio osoittaa luun selkeän pituuden, sillä nivelen "luu"-osalla ei ole varsinaista pituutta. Jos useammat translate-ominaisuudet eivät olisi nollattuja, luun pituuden derivointi ei olisi yhtä helppoa. Toinen peruste sille, miksi vain yksittäisen translaation tuli olla ei-nollattuna, on selkeä skaalaus. Jotta yksittäinen nivel voi omata yksittäisen transformaation, se tarkoittaa, että nivelen parentin tulee suoraan osoittaa child-niveleen.

Nivel-ketjun root-nivel, tai kokonaan uuden nivelketjun aloittava nivel, voi kuitenkin omata ei-nollatun arvon useammassakin translate arvossa.

3.3.1 Nivelien peilaus.

Rotaatiotyökalu voi mitata liikkensä monin tavoin. Rotaatiota voi ohjata objektin lokaalien akselien mukaan, tai sitten globaalien (world axis) akselien mukaan. Lokaali tranformaatio tekee mittauksensa objektin lokaalien arvojen ja orientaation mukaan.

Yksi toimivan riggin ominaisuus on peilautuneet arvot. Jos hahmo on sellainen, jonka molemmat puoliskot ovat symmetriset, kuten esimerkiksi ihminen, niin toivottavaa olisi, että molempien puolien rotaatiot olisivat symmetriset. Tämä tarkoittaa käytännössä sitä, että vasemman käden kääntyessä lokaalilla akselistolla positiiviseen suuntaan, niin oikea käsi kääntyisi samaan suuntaan samalla arvolla. Toisin sanoen molempien puolien rotaatiot olisivat symmetriset (kuva 21.).



Kuva 21. Molemmat olkapäät taipuvat symmetrisesti, koska toisen puolen nivelten orientaatio on vastakkainen toisesta.

Globaalilla akselistolla tätä symmetriaa ei pysty toteuttamaan koska globaalilla akselistolla on vain yksi positiivinen ja negatiivinen suunta. Nivelissä on kuitenkin mahdollista saavuttaa identtiset lokaalit muutokset molemmin puolin kääntämällä nivelien orientaatiot toisin päin toisella puolella kehoa.
Mayassa tälle nivelien peilaukselle on oma työkalunsa, mirror joints.
Mirror jointin avulla on mahdollista peilata luusto tiettyjen akselien poikki siten, että se kopioi toisen puolen käyttäytymisen osakseen.

Mirror joints- työkalu mahdollistaa peilauksen kahdella eri tavalla, käyttäytymisen mukaisesti tai orientaation mukaisesti. Käyttäytymisen mukainen heijastus kääntää vastapuolen nivelien orientaation siten, että transformaatiot ovat peilattavan osapuolen peilikuvia. Orientaation mukaisessa heijastuksessa heijastetun nivelen orientaatio on sama kuin heijastetun, jolloin rotaation suunta on molemmilla puolilla sama.

3.4 Meshin sitominen luustoon (skinning)

Meshin sitominen luustoon (skinning, käytän siitä kuitenkin ilmaisua ”skinnaus”) on prosessi, jossa mesh sidotaan valmistettuihin niveliin. Nivelien kääntäminen saa meshin kääntymään sen mukana, tarkemmin ilmaistuna deformoitumaan, eli sen control vertexit siirtyvät normaaleilta sijoiltaan. Käsittelen deformoinnin tarkemmin myöhemmin tutkielman aikana.

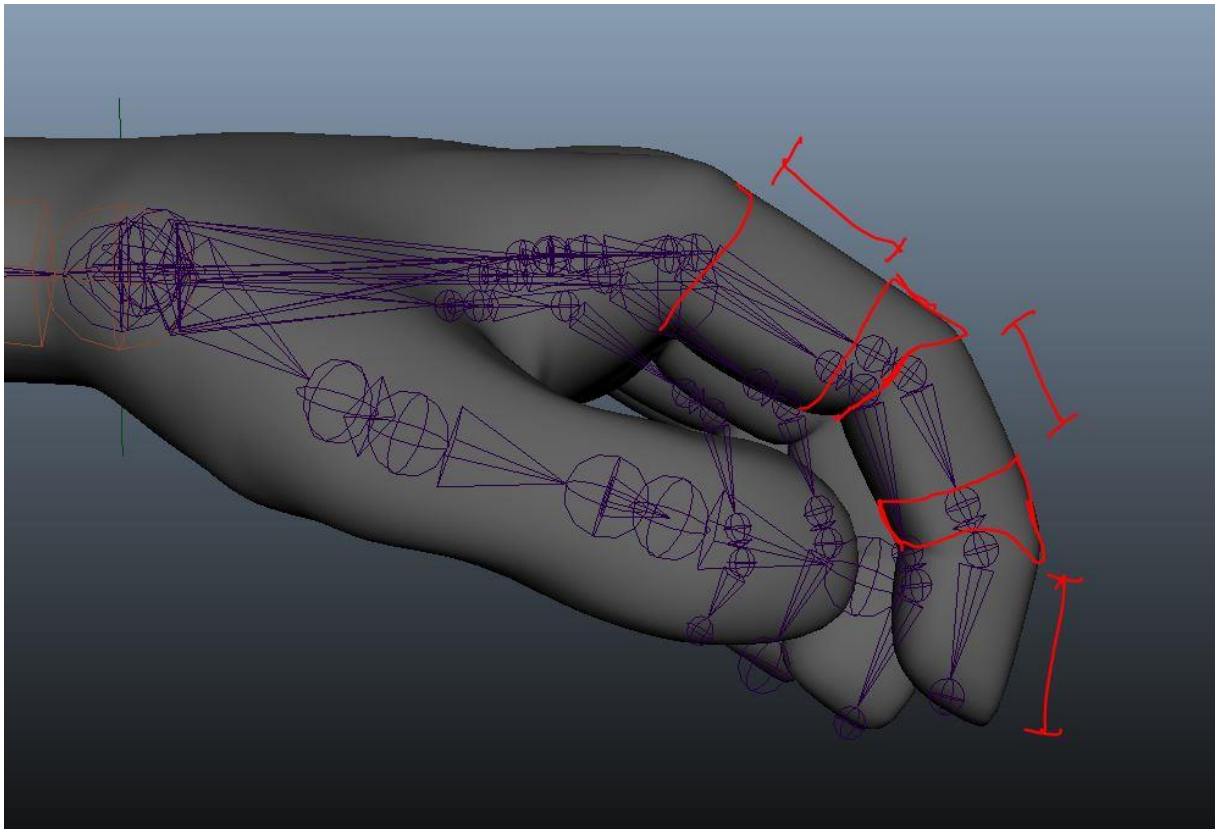
Prosessin ideana on, että meshiin syntyy input-liitos skincluster noodista, joka deformoi meshiä, joka on kiinnitettynä luustoon. Kun meshi valitaan, skinclusterin voi nähdä channel editorissa ominaisuuksien alla.

En tarkasti selitä kaikkia työkaluja skinnausinformaation käsittelyyn, mutta esitän teorian siitä miten nivelet skinnauksessa vaikuttavat meshiin. Tarkempaa informaatiota skinnaus-prosessista ja työkaluista kannattaa katsella Mayan manuaalin ja erilaisten nettioppaiden kautta.

Jokaisella skinnatulla meshin vertexillä on oma painoarvonsa tiettyihin niveliin nähden. Tämä painoarvo määrittää sen kuinka vahvasti kyseinen nivel vaikuttaa vertexiin. Painoarvo mitataan prosentteina. Sadan prosentin painoarvo tarkoittaa, että kyseinen vertex seuraa täysin isäntänivelensä liikettä.

Kun yhden nivelen painoarvo laskee vertexissä alle sadan prosentin, ylimääräinen painoarvo saattaa jakautua toiselle nivelelle. Se miten painoarvo jakautuu, riippuu skinclusterin asetuksista. Useimmissa tapauksissa käyttäjän täytyy säätää meshin painoarvoja, jotta optimaalinen deformaatio tapahtuisi.

Kun painoarvoja käsittelee (Mayassa tätä prosessia varten on skin-valikossa ”paint skin weights” – työkalu) meshin vertexien painoarvot useimmiten jakautuvat eri tavoin meshin pinnalla. Painoarvojen jakautumista voi säätää ominaisuudella ”normalize weights”, jolla on kaksi moodia: Interactive ja post. Interactive ominaisuuden ollessa päällä, vertexin jäljellä olevat painoarvot jakautuvat lähimpiin niveliin aina kun painoarvoja käsitellään. Jakautumista tapahtuu niin kauan kunnes vertexin painoarvo on sata prosenttia (tai 1). Tätä jakautumista kutsutaan normalisaatioksi (autodesk 2015c.) Kun vertexillä on useita niveliä jakamassa painoarvot, vertex seuraa suuremmassa määrin niitä niveliä, joilla painoarvo on suurempi.



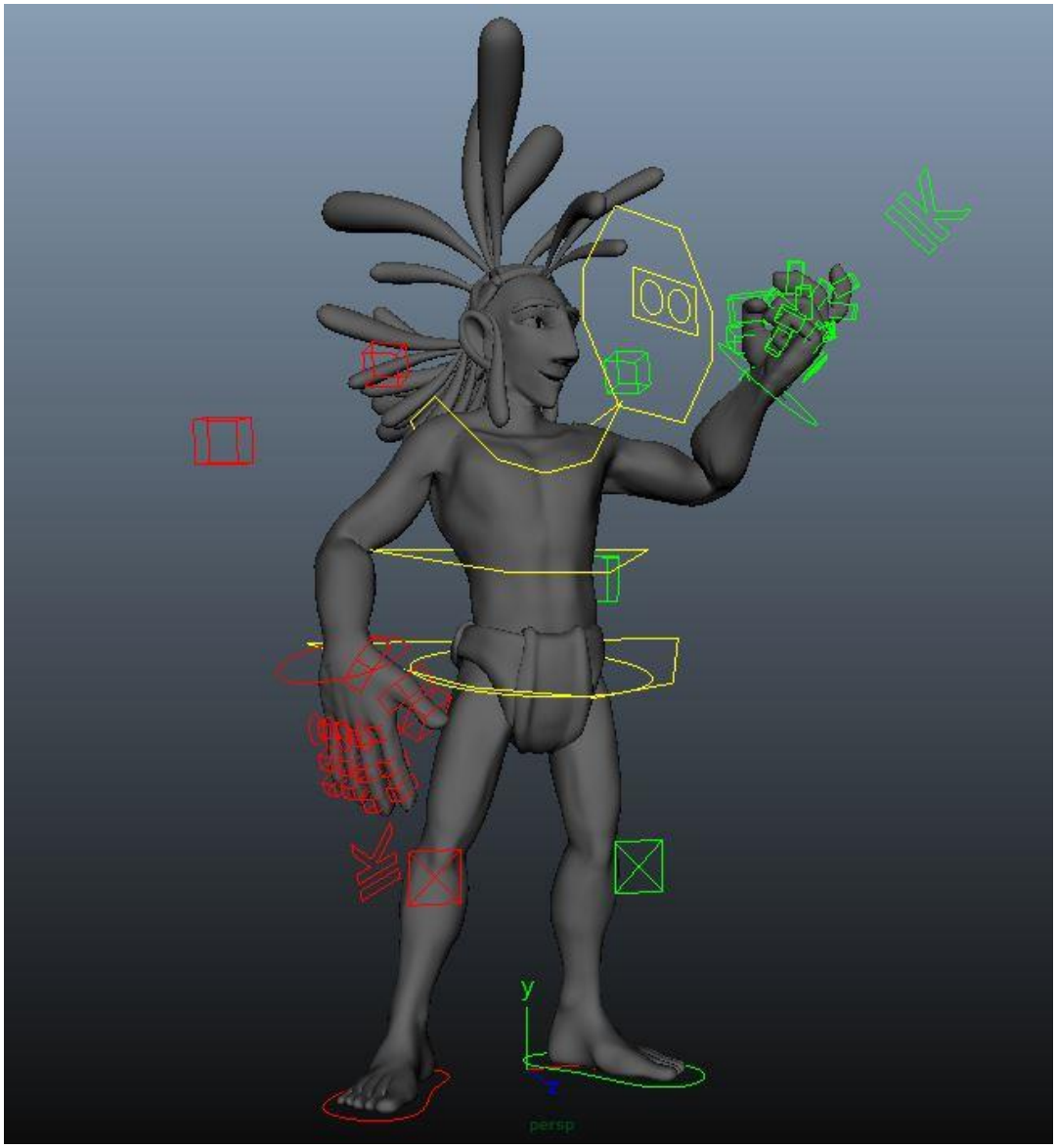
Kuva 21. Sormien riggauksessa olin kokeullut nivelrakennetta, jossa jokaisen anatomisen nivelen kohdilla on kaksi lähekkäistä niveltä. Ideana on, että kaksoisnivelistä ensimmäinen hallitsee merkattua aluetta ylläpitäen siten sormen rakenteen. Seuraava nivel taivuttaa varsinaista sormea.

4 Kontrollien perustyökalut, osa 1. Ohjaimetsidokset ja ik-ketjut

Tässä luvussa käsittelen niitä työkaluja, jotka ovat olennaisia erityisesti kontrollisysteemien pääpiirteiden kanssa. Ensimmäisessä osassa käsittelen kaksi merkittävää työkalua, jotka lähes väistämättä ovat osa Maya-riggausta, ik-ketjut ja sidokset (constraints).

4.1 Ohjaimet (controls)

Ohjaimilla tarkoitan niitä objekteja, joita käytetään hahmon luuston liikuttamiseen. Kontrollien tarkoituksena on toimia pidikkeinä, joita liikuttamalla ja niiden ominaisuuksia animoimalla, saadaan hahmoon liikettä.



Kuva 22. Pacalin ohjaimet koostuivat nurbs-käyristä. Käyrien etuna on niiden vähäinen tila ja eri objekti-tyyppi kuin itse meshissä. Säätämällä meshien valinnan pois päältä, nurbs-käyriin pääsee helpommin käsiksi ilman riskiä meshin valinnasta.

Yksinkertainen esimerkki on tavallinen nurbs-käyrä, joka hierarkioidaan jointin yläpuolelle. Jos nurbs-käyrää käännetään, se liikuttaa jointtia. Jos käyrä palautetaan nollapositioonsa, niin joint palaa myös nollapositioonsa.

Yleinen, hyväksytty sääntö 3D-teollisuudessa on, että kaikkien ohjainten transformaatioiden tulisi olla nollattuna! Nollaus on kriittinen prosessi animaatiota ajatellen. Kun objekteilla on selkeä nollakohta, niin animaattoreilla on selkeä vertailukohta eri animaatioobjektien sijainneille. Esimerkiksi kävely-animaatioissa animaattoreiden on helpompaa asetella jalkojen ja kämmenten animaatiot mahdollisimman symmetrisesti, jos niiden vertailukohta on sama. Ohjainten nollaus on myös yksi syy siihen, miksi nivelien kaikkien rotaatioiden tulisi myös olla nolla. Kun nivelen ja ohjaimen arvot ovat samat, keinot niiden yhdistämiseksi ovat laajemmat kuin silloin, jos arvojen välillä on eroa.

Kontrollin navan sijainnin tulisi myös olla samassa positiossa kuin sen nivelen, jota ohjain liikuttaa.Ideana on, että kontrollien nollakohdassa vastattavat jointit ovat täsmälleen siinä asennossa, jossa hahmo oli silloin kun mesh liitettiin tukirankaan. Luurangon nivelien siirtyminen paikaltaan voi siis aiheuttaa vakavia siirtymiä ja epämuodostumia meshissä, joka vaatisi meshin irrottamista luurangosta ja uudelleensitomista.

Ohjainten ja nivelten yhteispeli toimii siten, että ohjaimen transformaatiot heijastuvat ohjattavien nivelien transformaatioihin. Tämä muodostaa riskejä ohjainsysteemien käydessä liian monimutkaisiksi. Objektien väliset transformaatiot eivät aina ole yhdenmukaisia. Ohjainsysteemissä monet objektit ja apuobjektit ovat sidoksissa ja hierarkioituna toisiinsa ja yksi pieni virhe niiden sidoksien välillä voi kasautua hyvinkin suureksi.

On huomioitava, että objektien asettuessa hierarkiaan, niiden lokaali transformaatio mukautuu parentin sijaintiin. Toinen asia, jota on tarkkailtava, on objektien orientaatio. Jos normaalin objektin transformaatiot nollataan freeze-komennolla, objektin lokaali rotaatioakselisto palautuu nollapositioon (tämä ei tosin päde nivelissä, joiden rotaatio siirtyy orientaatio-ominaisuuteen). Hierarkiassa childin orientaatio määräytyy sen mukaan mihin parent osoittaa. Jos parentin rotaatio on x-akselilla 45 astetta, childin rotaation nollakohta osoittaa tähän suuntaan.

Lokaalien ja globaalien ominaisuuksien tapauksessa puhuin objektin erillisestä local-space ja world-space ominaisuudesta. Käytännössä objektien translaatio voitaisiin pitää

aina nollana ja sen sijaan liikutella itse objektin lokaalia napaa. Tämä ei kuitenkaan välttämättä johda toimiviin ohjainliitoksiin. Jos tietty objekti tarvitsee erikseen tiedon objektin sijainnista globaaliin napaan nähden, tämä etäisyys tulisi ilmetä translaation kautta. Rigin käsiteltävien objektien ollessa pääasiassa transform noodeja, niillä kaikilla on samat perusominaisuudet, translate, rotate ja scale, mutta attribute editorissa esiintyvät ominaisuudet saattavat paljonkin poiketa toisistaan. Nivelien toiminta lokaalien napojen suhteen ei ole täysin samanlainen kuin nurbs-käyrällä. Tämän vuoksi on viisaampaa pitää yhdisteltävät arvot translate, rotate ja scale akselistolla (niin usein kuin mahdollista), koska suurin osa rigeissä käytettävistä objekteista omaavat nämä ominaisuudet.

Se, millaisia itse ohjainobjektien tulisi olla, ei ole kiveen hakattua. Ohjainobjektit voivat olla minkä muotoisia tahansa. Ne voivat olla nurbs käyriä tai polygon-objekteja. Sain kuitenkin Jarolta kuulla hyvän huomion, että nurbs-objektien etuna on se, että ne ovat eri tyyppiä kuin meshit. Mayassa on mahdollista piilottaa erilaiset objektityypit kuten poluygonit tai nurbs-käyrät. Jos mesh ja ohjain-objekti ovat samaa tyyppiä, se voi haitata esimerkiksi animaation esikatselua, koska kaikki ohjain-objektit eivät kätkeydy. Tämä huomio kävi ilmeiseksi työharjoittelupaikkani rigissä, jossa muutamat kontrolliobjektit olivat tavallisia mesh-objekteja. Muita hyötyjä nurbs-käyrien käytössä ovat niiden helppo muokattavuus. Mielestäni tärkein syy on kuitenkin erikoistunut valinta. Mayassa on mahdollista valita juuri tietynlaisia objekteja ja tehdä toisenlaiset objektit sellaisiksi, joita ei voi valita. Objektityyppejä ovat esimerkiksi polygoniobjektit, nivelet, apuobjektit ja nurbskäyrät.

Vaikka ohjainten ulkonäölle ei erinäisiä sääntöjä ole, kontrollien asettelun, muodon ja tehtävän tulisi ilmetä animaattoreille. Kontrollien esteettiseen asuun panostaminen tekee animaattorien työn helpommaksi.

Toinen tärkeä huomioitava piirre on myös se, että objektin transformaatioiden on oltava nollattuina (paitsi scale, jonka nollapositio on 1).

4.1.1 Poikkeama (offset)

Hyvin usein rigauksen opetuksen yhteydessä puhutaan käsitteestä poikkeama (offset). Poikkeamalla tarkoitetaan luonnollisesti eroa globaalin ja lokaalin tai kahden objektin ominaisuuksien välillä. Poikkeama limittyy edellisessä luvussa käsiteltyyn lokaalin ja globaalin navan eroavaisuuksiin.

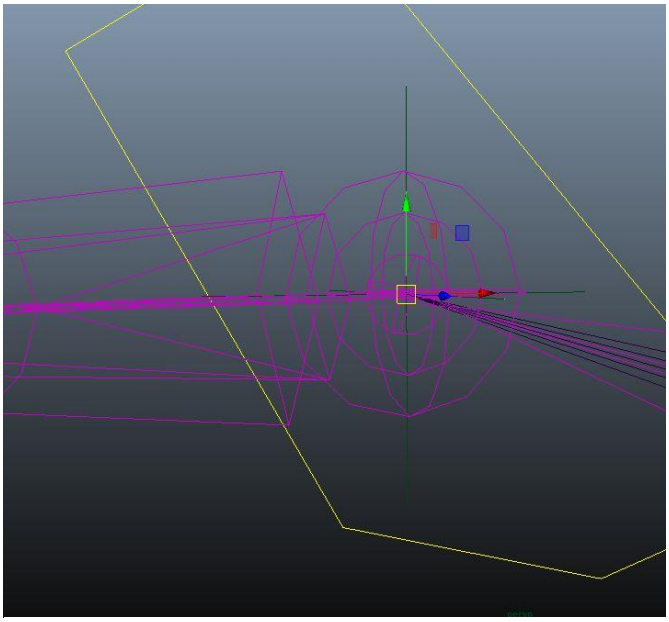
Poikkeamien huomiointi on tärkeää objekteja yhdistellessä. Erilaisten objektien väliset liitokset tarvitsevat joissain tapauksissa informaatiota esimerkiksi sen suhteen, missä objekti sijaitsee globaaliin akseliin nähden tai lokaaliin napaan nähden.

Käyn ryhmien yhteydessä tarkemmin läpi poikkeamien hallintaa.

4.1.2 Apuobjektit

Kutsun tässä tutkielmassa apuobjekteiksi niitä objekteja, jotka eivät ole varsinaisia ohjaimia, mutta ovat osana ohjainten liitoksia ja varmistavat niiden toiminnan. Käytännössä mikä tahansa objekti sopii apuobjektiksi.

Mayassa on kuitenkin erillinen objektityyppi, lokaattori (locator), joka nivelen tavoin on näkymätön rendauskuvassa. Lokaattori on pieni ristinmuotoinen objekti, jolla on samat transformaatioarvot kuin muilla transform noodeilla. Lokaattorit sopivat oivasti erilaisiin tehtäviin rigissä. Ne voivat omaksua nivelen orientaation rotaatioarvoihinsa ja sen asettua hierarkkiseen liitokseen ohjaimen kanssa. Tällä tavoin ohjaimella on varusteenaan oikein hierarkioitu lokaattori, joka puolestaan voi oikean orientaationsa vuoksi ohjata nivelen rotaatiota (kuva 23.).



Kuva 23. Lokaattori on aseteltu yhtenenväiseen orientaatioon nivelien kanssa ja pystyy siten ohjaamaan nivelen rotaatiota oikeassa suhteessa. Tämä on vain yksi mahdollinen käyttötarkoitus.

4.1.3 Ryhmitys

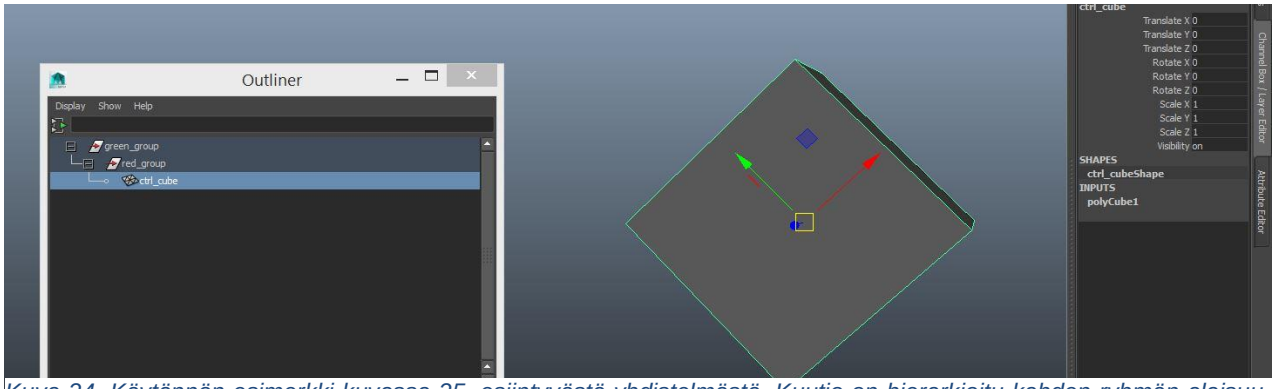
Toinen hyödyllinen ominaisuus, joka kontrollien hallinnassa on eduksi, on ryhmitys (grouping). Ryhmä on käytännössä yksi tyhjä transform noodi (noodiin ei siis ole hierarkioitu shape-noodia), jolla on muuten samat ominaisuudet kuin tavallisella transform noodilla. Ryhmän voi luoda yksinkertaisesti valitsemalla ensin tarvittavat objektit ja sitten group-komennon valitseminen (windowsissa ctrl+G).

Kun useita objekteja pistetään samaan ryhmään, ryhmän napa asettuu automaattisesti globaaliin keskipisteeseen. Tällöin jokainen suoraan ryhmään hierarkkisessa liitoksessa olevan objektin transformaatiota tapahtuvat ryhmän napaan, toisin sanoen globaaliin transformaatioon nähden. Ryhmä-noodi itsessään pystyy kokonaisuudessaan liikuttamaan jokaista sisältämäänsä objektia, ryhmän pivotiin nähden.

Ryhmien käyttö on yksi keino hallita poikkeamia ohjainjärjestelmässä. Esimerkiksi yksi objekti tarvitsee toimiakseen niin lokaalin navan sijainnin kuin translaationkin olevan nollattuina. Tämä ei ole mahdollista pelkän jäädytyksen avulla, sillä jäädytys nollaa translaation, mutta ei lokaalin navan sijaintia. Jotta molemmat, lokaalin navan ja translaation, arvot olisivat nollattuina, niiden tulee sijaita täsmälleen samassa pisteessä. Lokaali napa ei voi muuten olla nollattuna kuin siinä tapauksessa, että se on samassa pisteessä parentinsa kanssa.

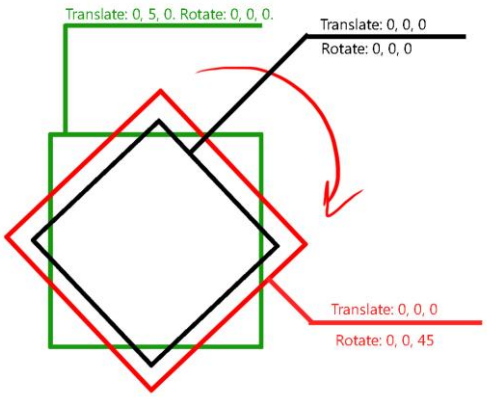
Ratkaisuna on objektin hierarkiointi ryhmän alaisuuteen. Tällöin ryhmän keskipisteestä tulee objektin lokaalin navan nollakohta. Kun ryhmän nollakohta sijaitsee täysin objektin omassa nollakohdassa, niin sekä objektin translaatio, että lokaalin navan sijainti ovat nollassa. Kun ryhmää liikutetaan työtilassa, ryhmän translaatio päivittyy, mutta objektin lokaalin navan sijainti ja translaatio pysyvät nollassa niin kauan kuin ryhmän oma napa pysyy samassa kohtaa objektin kanssa.

Jos ryhmitetty objekti asetetaan hierarkkisesti toisen objektin alaisuuteen, niin ensimmäisenä toisen objektin alaisuuteen pistetään ryhmä, jolloin muodostuu hierarkkinen ketju muodossa objekti-ryhmä-objekti. Tässä liitoksessa ryhmä säilyttää translaationsa, joka nyt kuvaa objektin etäisyyttä tämän parentista.



Kuva 24. Käytännön esimerkki kuvassa 25. esiintyvästä yhdistelmästä. Kuutio on hierarkioitu kahden ryhmän alaisuuteen. Vain ylin ryhmä saa osakseen transformaatioita, jolloin keskimmäinen ryhmä ja objekti seuraavat ylimmän ryhmän liikettä, mutta niiden omat transformaatiot pysyvät nollattuina.

Asian voi viedä vielä pidemmällekin, että ryhmän ja objektin välille asetellaan vielä toinen ryhmä. Tällöin objektia ennen on vielä toinenkin group, jonka kaikki arvot voivat olla objektin tavoin nollattuina (kuva 25.). Tämä on yksi tärkeä aspekti automatisoitujen sormikontrollien kanssa, joista esitän esimerkin viidennessä luvussa.



Kuva 25. Vasemmalla oleva kaavio kuvaa yläpuolella olevan kuution asentoa. Vihreä kuutio kuvastaa hierarkian ylintä ryhmää. Vihreän ryhmän rotaatiot ovat nollattu, mutta translaatio sijaitsee y-akselilla viisi yksikköä positiiviseen suuntaan.

Punainen kuutio on keskimmäinen ryhmä, joka on hierarkioitu vihreän ryhmän alaisuuteen. Punainen ryhmä on samassa pisteessä kuin vihreä ryhmä. Punaisen ryhmän translaatio on siten nollassa, mutta z-akselilla on tapahtunut 45 asteen rotaatio.

Musta neliö on itse kuutio-objekti ja on hierarkkisesti punaisen ryhmän alaisuudessa. Kuution akselit seuraavat punaisen ryhmän rotaatiota ja sijaintia, mutta kaikki sen ominaisuudet ovat nollattuja.

Tämä sama pätee silloinkin, kun ryhmät korvataan muilla objekteilla.

Asettamalla jokaisen kontrolliobjektin erikseen omaan ryhmään on mahdollista saada selville objektin sijainti sen ryhmän kautta, johon objekti on aseteltu.

4.2 Sidokset (constraints)

Sidokset ovat osin dependency liitosten johdannaisia. Nimensä mukaisesti constraintit ovat ikään kuin solmuja kahden noodin välillä. Constraint-sidokset mahdollistavat niiden objektien yhdistämisen, jotka eivät ole keskenään liitoksessa. Constraintin valmistus on simppeliä. Ensin valitaan ohjaava objekti (target) ja sitten ohjattava. Valitsemisjärjestys on siis päinvastainen kuin hierarkialla. Ohjaavan objektin kutsuminen kohteeksi saattaa kuulostaa ensiksi oudolta. Jos kuitenkin ajattelemme, että sidottu objekti pyrkii mukautumaan ohjaavan objektin ominaisuuksiin, ohjaavaa objektia voidaan siten pitää sidotun objektin tarkkauksen ”kohteena”.

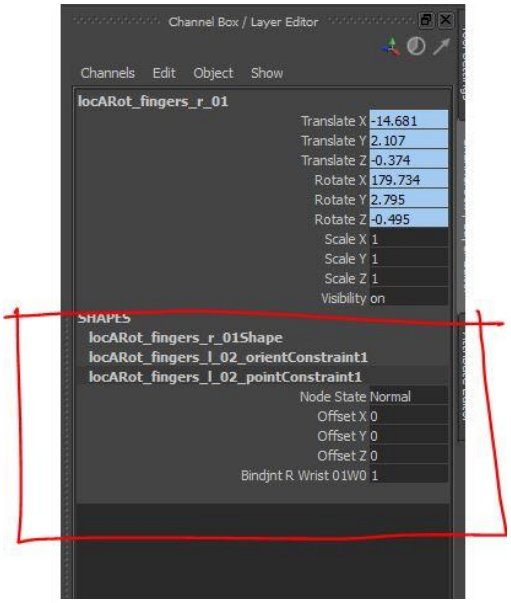
Constraint itsessään on oma noodinsa, jolla myös on omat säädettävät ominaisuutensa. Outlinerissa constraint näkyy huutomerkkinä ja on hierarkioituna constraint –liitoksen child –osapuolen alle.

Aloittelevan käyttäjän ei välttämättä tarvitse tietää sidoksen toimintaperiaatteita niitä käyttääkseen.

Sidoksen ideana on, että sidottu objekti omaksuu sidoksen luonteen mukaisesti kohteen orientaation (rotateX, Y ja Z), translaation (translateX, Y ja Z) tai skaalauksen (scaleX, Y ja Z) arvot. Sidotulla objektilla voi olla myös muita kohteita, jolloin sidottu objekti omaksuu kohteidensa arvojen keskiarvon.

On myös monia muita sidoksen muotoja ja arvoja, joihin sidokset vaikuttavat, mutta käyn läpi vain nämä olennaisimmat.

Constraint on hyvä liitosvaihtoehto silloin kun kahden liitettävän objektin arvot eivät ole yhteneväiset. Normaalissa liitännässä ominaisuuden A arvo = ominaisuuden B arvo. Sidos antaa kuitenkin mahdollisuuden ylläpitää objektien ominaisuuksien välinen eroavaisuus (offset). Tätä poikkeamaa kahden arvon välillä kutsutaan Mayassa nimikkeellä ”offset”, kutsun sitä tässä tutkielmassa kuitenkin “poikkeamaksi”. Käytännössä tämä tarkoittaa sitä kun ominaisuudella A ja ominaisuudella B on eroa x-arvon verran ennen sidosta, niin sidos ei muuta sidotun objektin arvoja samaksi kuin kohteen arvot vaan pitää yllä kohteiden välisen poikkeaman. Channel editorissa ja attribute editorissa tämän eron (offset) voi nähdä sidoksen ominaisuuksissa (kuva 26.).



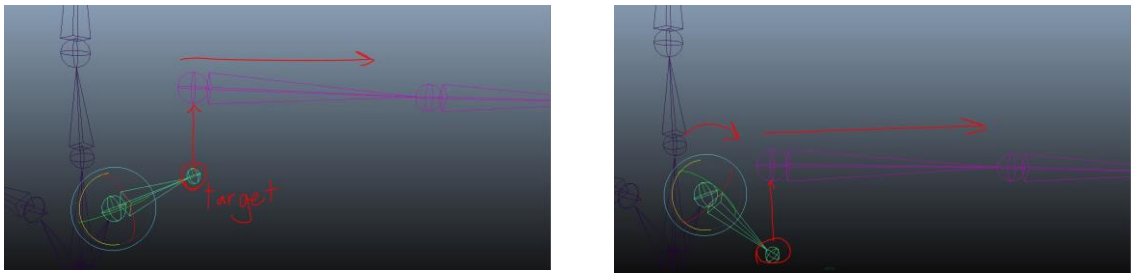
Kuva 26. Ympyröitynä sidoksen ominaisuudet. Tässä tapauksessa offset on nollassa. Sidotun objektin ja kohteen ominaisuuksien välillä ei siis ole eroa.

Channel editorissa ja attribute editorissa sidotut ominaisuudet, näkyvät sinisinä.

4.2.1
Orientaatiosidos yhdistää objektien rotaatioarvot. Sidottu objekti siis pysyttelee paikallaan, mutta sen orientaatio noudattaa kohteen orientaatiota.

4.2.2 Point constraint

Pistesidos (point constraint) sitoo kahden objektin sijainnit toisiinsa. Sidottu objekti seuraa kohteen liikettä, mutta ei orientaatiota. Pistesidos poikkeaa normaalista hierarkkisesta liitoksesta siinä suhteessa, että sidottu objekti pysyttelee alati omassa orientaatiossaan (kuva 27).



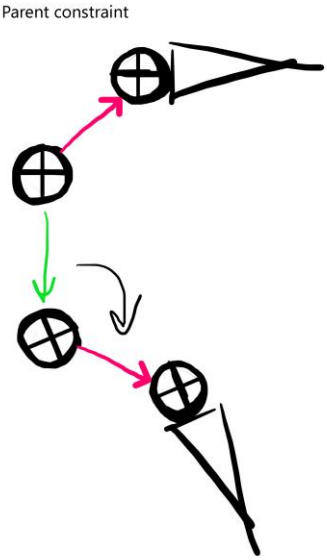
Kuva 27. Ympyröity nivel on point constraintin kohde (target), jonka liikettä ylempi nivelketju seuraa. Huomaa, kuinka kohteen kääntyminen siirtää sidotun nivelen alaspäin, mutta ei muuta suuntaa, johon sidottu nivel osoittaa. Sidotut objektit pystyvät myös pitämään keskinäisen etäisyytensä, koska poikkeaman ylläpito (maintain offset) on päällä.

4.2.3 Scale Constraint

Skaalaussidos yhdistää sidotun objektin kohteen skaalaukseen.

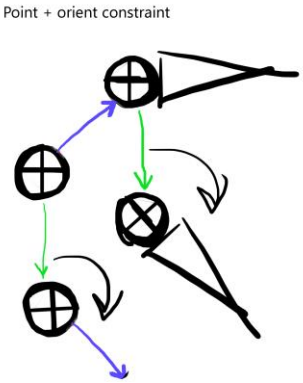
4.2.4 Parent constraint

Kutsun tätä hierarkiasidokseksi, sillä se käytännössä ajaa saman asian kuin hierarkkinen sidos. Erona tavalliseen hierarkkiseen sidokseen on se, että liitoksen voi tehdä niihinkin objekteihin, jotka eivät ole osa samaa hierarkiaketjua. Parent-sidoksella sidottu objekti toimii kohteensa suhteen samoin kuin child parentia kohtaan hierarkiaketjussa (kuva 28.).



Kuva 28. Parent-sidos mukailee hierarkkista liitosta. Oikean puoleinen nivel on sidottu vasemman puoleiseen niveleen. Vihreä viiva kuvaa vasemman nivelen translaatiota alaspäin. Samaan aikaan nivel kääntyy myötäpäivään.

Sidottu objekti kääntyy ohjaavan nivelen navan mukaan.



Kuva 29. Piste+Orientaatio-sidos. Oikeanpuoleinen nivel on sidottu rotaatiostaan ja translaatioltaan oikeanpuoleiseen niveleen.

Oikeanpuoleinen nivel seuraa liikkuu alaspäin samalla kääntyenmyötäpäivään. VaSidottu nivel ei käännykään ohjaajan navan mukaan vaan pyörähtää oman napansa mukaan.

On myös mainittava, että objekti on mahdollista sitoa molemmilla sidoksilla, piste- ja orientaatiosidoksella samaan aikaan. Tapahtuva transformaatio ei kuitenkaan ole sama kuin hierarkiasidoksessa. Hierarkkisessa liitoksessa sidotun objektin napa mitataan kohteen navasta, kuten tavallisessa hierarkkisessa liitoksessa.

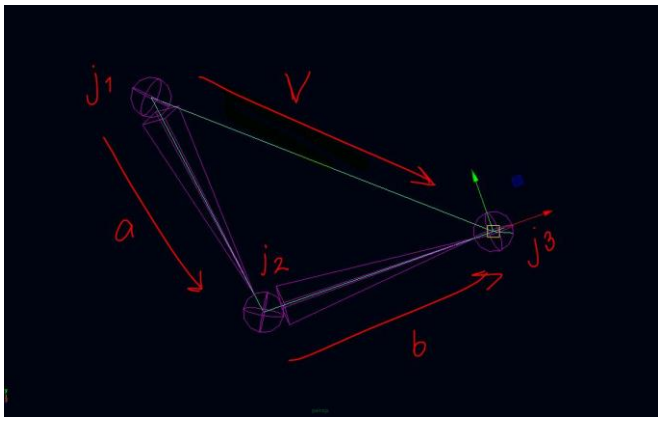
Piste-orientaatio-sidoksen tapauksessa sidottu objekti ei puolestaan seuraa kohdeobjektin napaa vaan kiertää oman napansa

ympäri (kuva 29.). Ellei piste-sidoksen “maintain offset” asetusta ole säädetty päälle sidottu objekti omaksuu kohteen sijainnin, käytännössä siis siirtyy samaan pisteeseen kuin kohde.

4.3 Ik-ketjut

Ik-ketjut (inverse kinematics chain) pystyvät nimensä mukaisesti liikuttamaan niveliä vastakkaisessa järjestyksessä. Normaalisti nivelketjuja voi liikuttaa vain ylemmästä nivelestä alempaan. Käytän tästä hierarkkisesti ylhäältä alaspäin suuntautuvaa nimikettä lineraarinen liike (frontal kinematics). Ik ketju mahdollistaa nivelien liikutuksen hierarkian alapäästä. Ik-ketju luo pidikkeen nivelketjun päähän ja tätä pidikettä liikuttamalla on mahdollista asettaa nivelketju haluttuun asentoon.
Ik-ketju luodaan valitsemalla ketjun ensimmäinen ja sitten viimeinen nivel. Ketjun ensimmäisestä nivelestä käytän nimikettä alkunivel ja viimeisestä loppunivel (Autodesk 2014a.)

Ik-ketju luo viivaston, joka kulkeutuu valittujen nivelien poikki. Loppunivelen kohdille syntyy pidike (handle), jonka liikuttaminen ohjaa ik-ketjuun sidottuja niveliä (kuva 30.).



Kuva 30. Viivat a ja b ovat osa matemaattista laskelmaa kun pidikettä (sijaitsee translaatiotyökalun kohdalla) liikutetaan. Tämä matemaattinen ratkaisu ohjaa ja kääntää niveliä.

4.3.1 Ik:n osat

Pidike (handle)

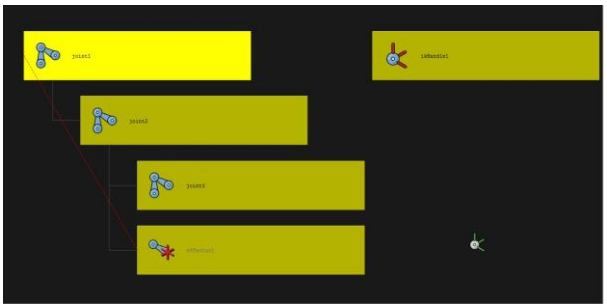
Ik:n pidike (handle) on ristikuvio, joka ilmestyy loppunivelen kohdalle. Pidikkeen liikuttaminen liikuttaa nivelketjua. Pidikkeen sijainti (translate-arvo) mitataan globaalissa tilassa, eli työtilan nollakohdasta. Tämä on hyvä muistaa silloin kun pidike sidotaan toiseen objektiin.

Handle vector

Handle vektor on viiva, joka kulkeutuu alkunivelestä loppuniveleen, ik-handleen.

Vaikutin (End effector)

End effector on komponentti, joka on hierarkioitu ik-ketjun toiseksi viimeisen nivelen alle ja on sijoitettu loppunivelen kohdalle. Vaikutin-noodi syntyy automaattisesti ik:n luomisen yhteydessä ja on asetettu perusasetuksiltaan näkymättömäksi. Vaikutin on varsinainen komponentti, joka liikuttaa ja ajaa (drive) ik-ketjua. Pidike on ominaisuuksiltaan liitetty automaattisesti vaikuttimen ominaisuuksiin. Vaikutinta on kuitenkin mahdollista liikuttaa toisiin paikkoihin tilanteen niin vaatiessa.



Kuva 31. Ik-ketjun hierarkia, handle toimii globaalissa tilassa effectorin ollessa hierarkioituna toiseksi viimeisen nivelen alle.

Solver

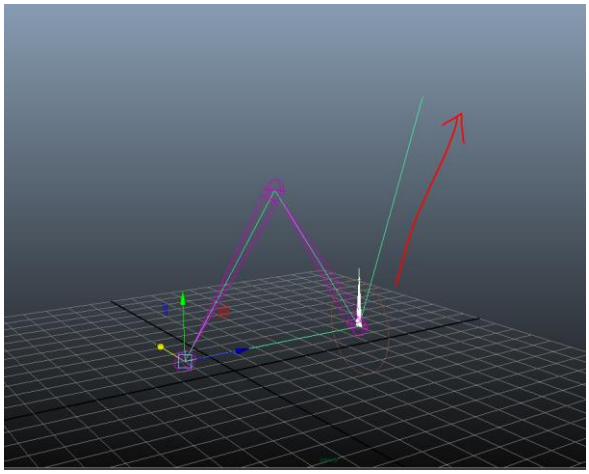
Solver on ik-ketjun osa, joka sisältää matemaattiset laskelmat sille, miten ketjussa mukana olevat jointit liikkuvat. Ratkaisimia on erilaisia, mutta käsittelen niistä vain kahta olennaisinta:

Sc-ratkaisin (single-chain solver)

Sc-ratkaisin huomioi pidikkeen rotaation muutokset ja heijastaa ne nivelien orientaatioon.

Rp-ratkaisin (Rotate-plane solver)

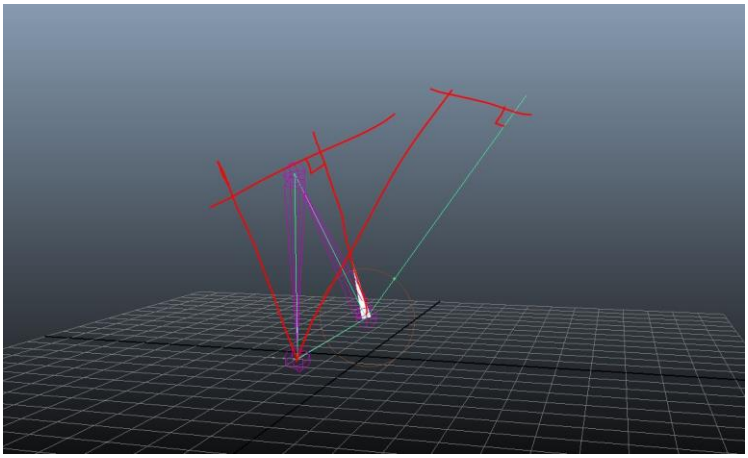
Rotaatiopinta-ratkaisin pyrkii pitämään nivelketjun orientaation yhteneväisenä. Ik-pidikkeen rotaatio ei siirry vaikuttimen rotaatioon, eikä siten vaikuta nivelien asentoon. Sen sijaan rp-ratkaisin luo alkunivelen kohdille ympyrälevyn, jossa on valkea nuoli. Tämä valkea nuoli on nivelien orientaation indikaattori ja osoittaa nivelien orientaation suunnan. Valkean nuolen lisäksi levystä erkaantuu toinen turkoosin värinen viiva. Tämä on pole-vektori (kuva 32.) (Autodesk 2015e.)



Kuva 32. Punainen nuoli osoittaa pole vektorin suunnan. Valkea nuoli kuvaa ik-ketjun kokonais orientaatiota. Kun handlea liikutetaan, valkea nuoli pyrkii osoittamaan mahdollisimman tarkasti pole

Pole vector

Rp ja single-chain ratkaisinten erona on, että rp-ratkaisimessa pidikkeen rotaatio ei vaikuta nivelketjun orientaatioon. Rp-ratkaisimessa nivelten orientaation määrittää pole-vektori. Pole vektori muodostaa referenssipinnan. Nivel-ketjun indikaattori puolestaan määrittää nivelketjun orientaatiopinnan. Normaalisti nivelketjun indikaattori pyrkii osoittamaan pole-vektorin suuntaan (kuva 33.).



Kuva 33. Vasemmanpuoleinen punainen kuviointi kuvastaa nivelien muodostamaa rotaatiopintaa. Oikeanpuoleinen vastaa referenssipintaa. Orientaatiopinta ei seuraa referenssiä, ik-pidikkeen ik-twist ominaisuutta on säädetty.

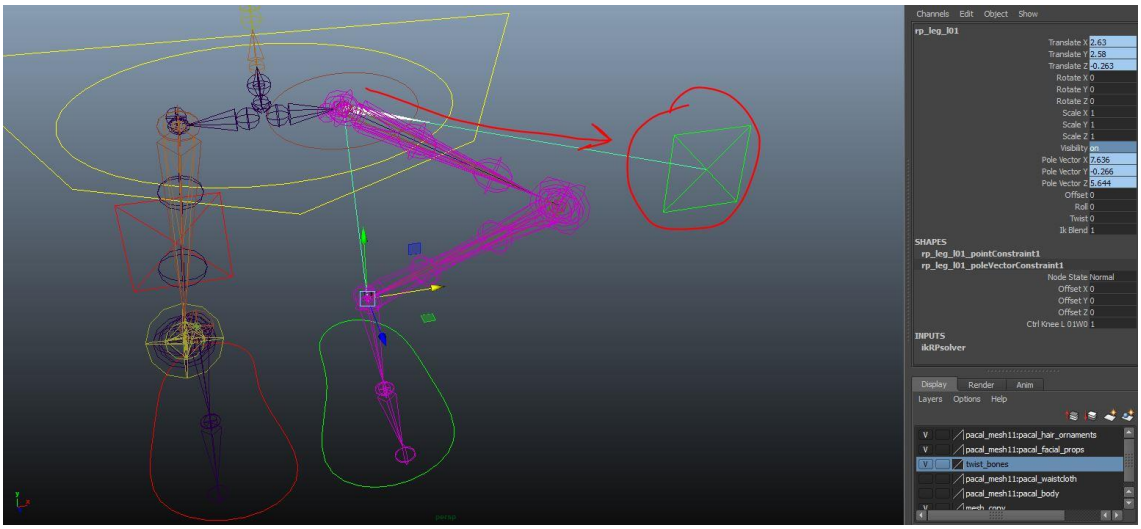
Joskus käy niin, että handlea liikutellessa, pidike ylittää referenssipinnan tai orientoituu vastakkaiseen suuntaan sen kanssa. Tällöin tapahtuu hyppäys (flip), jossa nivelten orientaatio käännähtää yllättäen vastakkaiseen suuntaan.

4.3.2 Twist

Twist on rp-katkaisimessa vaikuttava ominaisuus, joka voi muuttaa nivelketjun orientaatiota suhteessa referenssiorientaatioon. Tämä on yksi keino välttää animoidessa hyppäystä tilanteen niin vaatiessa.

4.3.3 Pole constraint

Pole-sidos (pole constraint) löytyy samasta valikosta kuin muutkin sidokset. Pole constraint mahdollistaa sen, että toisen objektin voi säätää ohjastamaan ik-handlen pole vectorin suuntaa. Tämä on yleinen työkalu polvien ja kyynärpäiden ohjaamisessa.
Ennen constraintin tekoa on hyvä varmistaa, että kyseinen objekti on kohtisuorassa ik-ketjun pole vectorin suuntaan nähden. Suunnan voi määrittää valitsemalla ik-ankkurin ja katsomalla mihin suuntaa planen valkea nuoli osoittaa.



Kuva 34. Rp-ratkaisimen pole-vektori on sidottu ympyröityyn objektiin. Pole-vektorin suunta määrittyy sen mukaan, missä objekti sijaitsee.

4.4 Ik-spline

Ik-spline on erikoisempi tapaus kuin tavallinen ik-ketju. Käytännössä tavallisella ik-ketjulla ei ole maksimia sille, kuin monta niveltä käytettävässä ketjussa voi olla, mutta nivelien määrän noustessa yli neljän, riskinä on, että nivelten liikkeistä tulee hyvin arvaamattomia. Käärmeiden usean nivelen taipuisa luurakenne voi olla haastava toteuttaa tavallisen ik-ketjun kanssa. Ik-spline pyrkii vastaamaan tähän ongelmaan erillisen nurbs-käyrän avulla.

Ik-spline rakennetaan samoin kuin normaali ik. Ensin valitaan alkunivel ja sitten loppunivel. Nivelten väliin muodostuu samanlainen ik-käyrä kuin ik:ssa, mutta valittujen nivelien kohdalle syntyy erillinen nurbs-käyrä. Tämän käyrän control-vertexien hallinta määrittää nivelien asennon.
Ik-splinen työkaluvalikosta voi säädellä splinen muodostaman curven ominaisuuksia.

Ongelmana on se, että control vertexit ovat osa käyrän shape-noodia. Shape noodiin kuuluvat komponentit eivät ole hallittavissa samalla tavoin kuin transform noodin ominaisuudet. Tässä tapauksessa ratkaisuna on deformereiden käyttö (deformers), joita käsittelen kuudennessa luvussa.

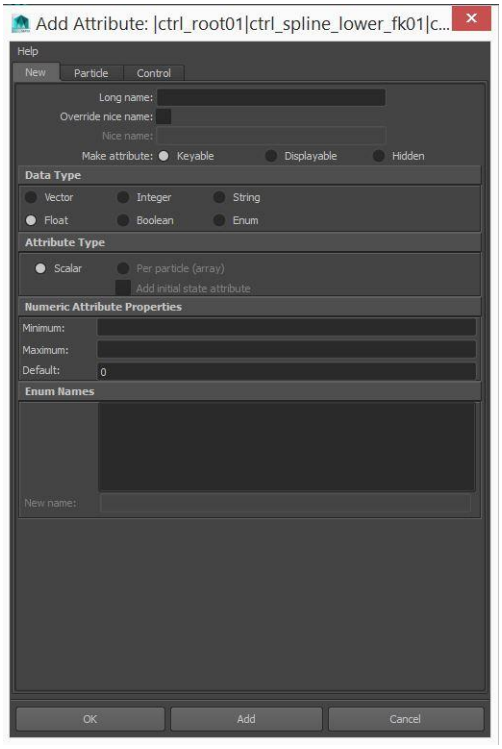
5 Kontrollien perustyökalut, osa 2. Uusien ominaisuuksien valmistus, ominaisuuksien linkitys, utility-noodit ja set-driven key

Tässä luvussa käsittelen sellaisia työkaluja ja rigin osasia, jotka ovat astetta syvemmälle aseteltuina rigiin. Tässä luvussa käsittelen uusien ominaisuuksien valmistamista ja piilotettujen ominaisuuksien yhdistämistä rigiin. Toinen tärkeä käsiteltävä asia on ”set-driven key”-systeemi, joka hyödyntää ennalta säädettyjä animaatiota toiminnassaan. Käsittelen utility-noodien hyödyntämistä liitoksien tekemisessä ja esittelen sidoksien, hierarkioiden ja ”set-driven-keyn” yhteistyötä esimerkin kautta, jossa rakennan sormille erillisen automaattisen animaation kanavan.

Perustelen näiden asioiden käsittelyä erillisessä luvussa sillä, että edellisen luvun työkalut olivat vahvemmin sidottuja transform-noodien sijainteihin ja objektien välisiin konkreettisiin liitoksiin. Tässä luvussa olevat liitokset ovat asteen hienovaraisempia ja pysyttelevät enemmän nooditasolla.

5.1 Lisäominaisuuksien valmistus

Lisäominaisuuden rakennus on varsin simppeliä. Channel editorin ”edit” -painikkeen kautta aukenee valikko, josta valitaan työkalu ”add attributes”. Tämä aukaisee uuden ikkunan (kuva 35.).



Kuva 35.

Valmistettu ominaisuus ei yksinään tee vielä mitään. Toimiakseen ominaisuus on osaltaan liitettävä toiseen valmiiseen ominaisuuteen, jolla on selkeä funktio. Ehtona on, että käytetyn arvon datatyyppi on samanlainen.

Float: ’Leijuva’ numero tarkoittaa desimaalilukua. Valmistettu kanava siis voi sisältää luvun, joka sisältää desimaalin.

Integer: Integer on muuten sama kuin float, mutta koostuu ainoastaan kokonaisluvuista, kuten -1, 0 ja 1.

Enum: Enum on listaus ominaisuus. Ikkunan alareunassa on esiasetuksena nimet “blue” ja “green”. Nimikkeet voi muuttaa valitsemalla nimikkeen ja kirjoittamalla uuden version sen alle. Enum on käytännöllinen silloin, kun ominaisuuden on tarkoitus vaihdella erilaisten tilojen välillä, kuten fk ja ik moodi.

Boolean: Boolean ominaisuudella on vain kaksi moodia. Kyllä tai ei, ”truth” tai ”false”, ”on” or ”off” ja niin edelleen. Toimii hyvin esimerkiksi kytkimissä.

En käsittele “vector”-datatyyppiä tässä tutkielmassa tiukan aikataulun vuoksi.

5.2 Ominaisuuksien yhdistäminen ja mitä niitä yhdistäessä on huomioitava?

Kerroin jo toisessa luvussa dependency liitoksista, joissa eri noodien ominaisuudet jakavat informaatiota ajaen toinen toistaan. Monet työkalut ja sidokset omaavat tarvittavat liitokset jo valmiiksi, mutta omien liitosten tekeminen on varsin simppeliä. Dependency liitoksia voi tehdä connection editorin kautta, yhdistellä niitä suoraan node editorin tai hypergraphin kautta.

Kuten luvussa 2 mainittiin, ohjaavaa ominaisuutta kutsutaan ajuriksi ja ohjattavaa ominaisuutta ajetuksi. Yksi ajuri voi ohjata useampaa ajettua ominaisuutta, mutta sellaisenaan yhtä ominaisuutta ei voi ajaa kaksi ajuria, ellei tarvittavia utility-noodeja ole käytössä. Ajetun objektin ajetut ominaisuudet näkyvät channel ja attribute editorissa keltaisina.

Suorien liitosten tapauksessa on tärkeää muistaa, että liitettyjen ominaisuuksien arvot siirtyvät sellaisenaan. Eli ajavan objektin ominaisuuden arvo siirtyy ajetun objektin arvoon. Tämä pätee silloinkin, vaikka ominaisuudet ajaisivat eri asiaa. Esimerkiksi objektin A rotateZ-arvo siirtyy suoraan objektin B translateX-arvoon. Jos Objektin A rotateZ on esimerkiksi 50 astetta, niin objektin B translateX on myös 50 yksikköä. Tämä johtuu siitä, että siirtyvä data-tyyppi on samanlainen.

Liitosta tehdessä on myös muistettava sekin, että toisin kuin sidoksissa, normaali liitos ei kompensoi liitettävien ominaisuuksien välillä olevia eroja. Jos siis kaksi eri pisteessä olevaa objektia liitetään translaatioiltaan toisiinsa, ajettu objekti siirtyy ajurin pisteeseen. Liitosta tehdessä on siis suositeltavaa, että ennen liitosta käytettävien objektien ominaisuudet ovat yhteneväiset. On hyvä huomioida se, että objekti, jonka tulee seurata vain

toisen objektin rotaatiota, ei välttämättä tarvitse olla samassa pisteessä. Jos taas objektin tulee seurata vain ajurin pistettä, niin silloin vain translaatioiden kannattaa olla yhteneväiset.

Ominaisuuksien yhteneväisyys rigatessa on tärkeää sen vuoksi, että jo pienikin ylimääräinen rotaatio tai translaatio voi heijastua koko muuhun rigiin aiheuttaen lumipalloefektin. Mitä pidempiä rigin liitossarjat ovat, sitä tarkempana on oltava sen suhteen miten liitokset yhdistyvät.

Lopuksi vielä on muistutettava hierarkian vaikutuksista ja etenkin transformaatioiden nollauksesta. Useimmissa tapauksissa objektit liittyvät toisiinsa translate, rotate ja scale ominaisuuksista. Hierarkioista kuitenkin muistamme, että nämä ominaisuudet määriytyvät vahvasti sen mukaan millaiset ovat objektin lokaalit ja globaalit navan sijainnit ja mikä on objektin sen hetkinen rotaatio.

5.2.1 Esimerkki hierarkian ja liitosten keskinäisten vaikutteiden huomioinnista

Oletetaan, että ik-pidikkeen sijainti globaalissa tilassa (ik-pidikkeen sijainti mitataan useimmiten globaalista tilasta) on 30, 20, 12 xyz-koordinaatistossa. Valmistamme lokaattorin, joka siirretään täsmälleen samaan pisteeseen. Päätämme liittää lokaattorin translate ominaisuudet ik-pidikkeen vastaaviin. Tällä hetkellä, jos liikutamme lokaattoria, ik-pidike seuraa perässä. Seuraavaksi rakennamme nurbs käyrän, jonka sitten siirrämme samaan pisteeseen kuin lokaattori. Kun lokaattori hierarkioidaan nurbs-käyrän alle, ik-pidike siirtyy globaaliin nollapisteeseen.

Tämä tapahtui siksi, että lokaattorin siirtyessä nurbs-käyrän alaisuuteen, nurbs käyrästä tuli lokaattorin lokaali napa, ja täten uusi nolla-kohta. Käyrän ja lokaattorin sijaintien ollessa samat ennen hierarkiaa, lokaattorin translate arvot nollaantuivat. Ik-pidike, jonka translate-arvoja lokaattori ajaa, huomioi lokaattorin arvojen muutoksen ja kopioi ne, tässä tapauksessa koordinaatit 0, 0, 0.

Tämän olisi voinut välttää esimerkiksi sillä, että hierarkioinnin yhteydessä lokaattorin sijainti tämän lokaalista navasta olisi pysynyt vakiona. Tämän voi saavuttaa esimerkiksi niin, että objekti, johon lokaattori hierarkioidaan, sijaitsee samassa pisteessä kuin lokaattorin lokaali napa ennen hierarkiointia.

Edellä mainittu esimerkki pätee siinä tapauksessa, jos vain objektin transform ominaisuudet ovat liitoksen kohteena, mutta Maya mahdollistaa myös esimerkiksi lokaalin navan sijainnin liittämisen toiseen ominaisuuteen. En kuitenkaan vielä lähde niin pitkälle tässä tutkielmassa.

5.3 Ominaisuuksien hallinta utility noodeilla

Liitoksien hallinta voi lopulta käydä haastavaksi jos itserakennettujen liitoksien ominaisuudet siirtyisivät sellaisenaan muihin objekteihin. Useimmiten käyttäjien tulee pystyä hallitsemaan näiden liitoksien toimintaa.

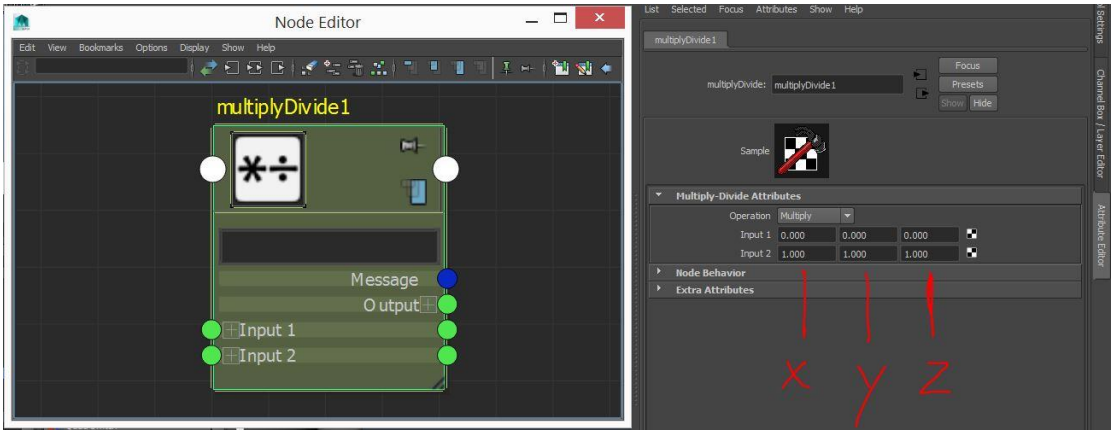
Yksi keino näiden liitoksien hallintaan ovat utility-noodit. Utility-noodien tehtävää on haastavaa summata muutamalla sanalla, sillä niitä on lukuisia erilaisia. Ne voivat toteuttaa laskutoimituksia noodien ominaisuuksien välillä tai liittää eri ominaisuudet yhteen. Yksi tehtävä tuoda ilmi kahden eri ominaisuuden keskiarvo ja liittää sen toiseen noodiin (plus-minus-average). Toinen on curven pituuden mittaus ja saatavan arvon siirtäminen noodin toiseen ominaisuuteen (curve info) tai noodien kesken kulkeutuvan arvon kertominen tai jakaminen (multiply divide).

En koe tarpeelliseksi selittää jokaista näistä erikseen, se vaatisi jo oman oppaansa. Esitän kuitenkin esimerkin kautta miten utiliy-noodin yhdistäminen noodien verkkoon tapahtuu. Suosittelen selvittämään eri utility noodien tehtävät mayan manuaalin kautta. Oikein asetellut utility noodit pystyvät kasaamaan monipuolisen ohjainjärjestelmän kun käyttäjä tietää mitä tekee.

Mainittavan arvoinen asia utility-noodeissa on käytettävät termit. Monissa noodeissa on käytössä nimikkeitä kuten inputX, color Green, Blue, Red tai tavallinen numeraali. Nämä nimikkeet eivät välttämättä tarkoita itsessään mitään vaan ovat simppelisti indikaattoreita, jotka merkkaavat input ja output porttien nimikkeet. Color Green porttiin voi siis lähettää numeraaliarvoja, jos se siis kyseisen noodin tehtävään sopii. Suosittelen selvittämään Mayan manuaalin kautta asiat tarkemmin.

5.3.1 Esimerkki utility noodien käytöstä: Multiply/divide

Yksi yleisimpiä käytettyjä utility-noodeja ovat matemaattisia kaavoja ratkaisevat noodit. Niiden tehtävänä on luonnollisesti muuttaa ajavan objektin arvot sellaiseen suuruuteen, että ajettu objekti toimii halutulla tavalla. Ajettavan objektin rotaatio voi olla esimerkiksi puolet ajurin rotaatiosta. Ideana on, että noodin input-porttiin voidaan liittää numero-ominaisuus, jota halutaan käsitellä. Vaihtoehtoisia kanavia on kolme (input1X, input1Y, input1Z), mutta nimistä huolimatta ei ole niinkään väliä asettaako esimerkiksi translateX ominaisuuden input1Y porttiin, mutta input1X-portin käyttö tekee toiminnasta selkeämpää (kuva 36.).

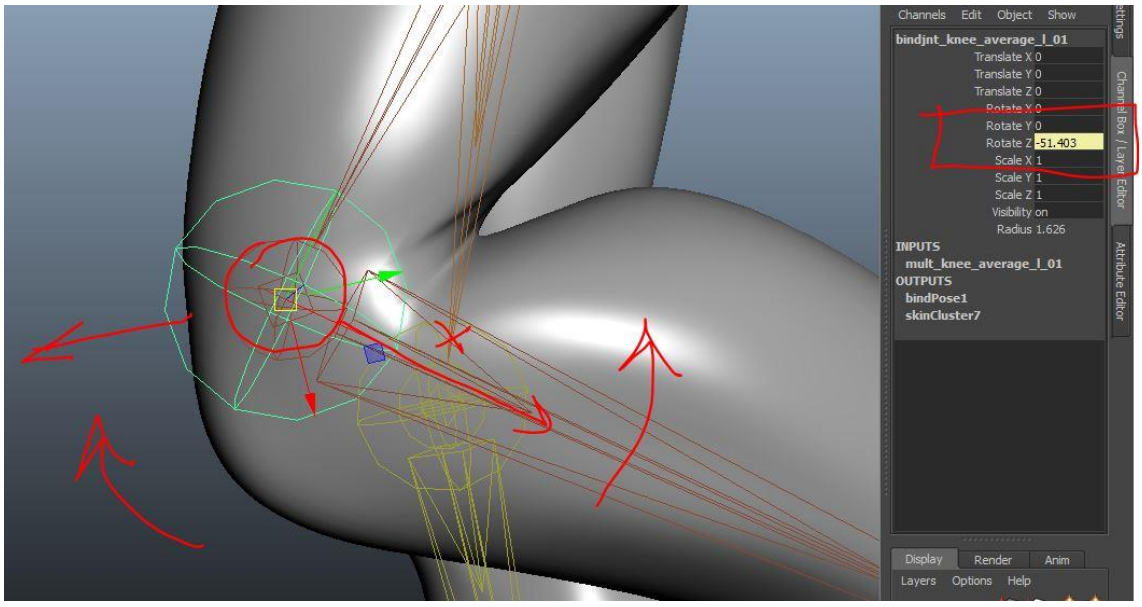


Kuva 36. Multiply noodi. Oikealla sen ominaisuudet attribute editorissa. Operaatio-sarakkeesta on mahdollista muuttaa laskutoimitustapaa.

Jokaisen input-ominaisuuden alla on omat sarakkeensa (input2X, input2Y, input2Z), jossa on luku, joka toimii ylemmän luvun kertoimena, jakajana tai potenssina. Näihin input2-portteihin on myös mahdollista liittää ominaisuuksia. Ajurin ominaisuus voi siis itsessään toimia laskijana.

Yksi kohta, jossa hyödynsin kyseistä noodia, oli polvien deformaatioiden hallinta. Normaalisti polvissa on astetta terävämpi kulma polven kääntyessä, mutta yksittäisen nivelen muodostama deformaatio saattaa olla turhan pyöreä ollakseen uskottava. Oikeanlaisilla skinnauksen painoarvoilla yksinään ei välttämättä saa aina parhainta lopputulosta.

Tässä on kyse apunivelistä. Vaikka rigin nivelten voi ajatella noudattavan osin anatomisia lakeja, pelkkä anatominen korrektius ei vielä tee deformaatioista uskottavia. Joissain tapauksissa tarvitaan extra-anatomisia niveliä, jotka pitävät meshin deformaation kasassa. Nämä apunivelet eivät välttämättä liiku anatomisen oikeaoppisesti.



Kuva 37. Valittu nivel on apunivel, jonka rotaatiosuunta on vastakkainen polvinivelen rotaatioon. Tuloksena on täten kiinteämpi deformaatio polvessa.

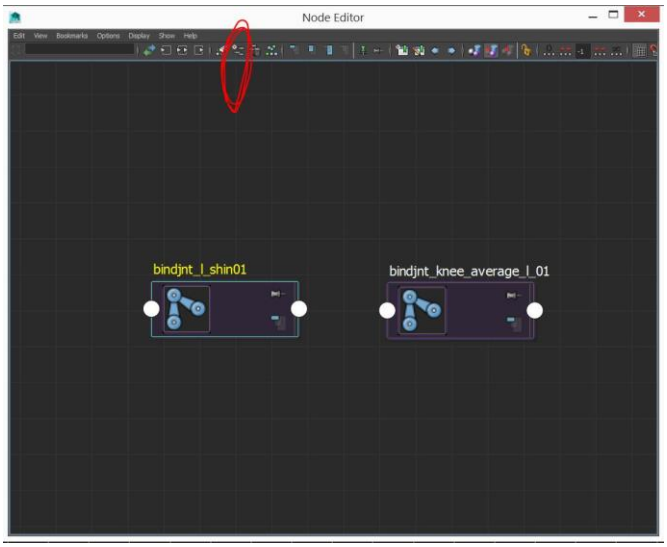
Polveen liitettävän apunivelen tarkoituksena on kääntyä vastakkaiseen suuntaan kuin varsinainen polvinivel. Jos Polvitaipeen alueella sijaitsevat vertexit sidotaan tähän apuniveleen, mesh deformoituu hallitummin (kuva 37.).

Tässä kohtaa on huomioitava myös yksi syy, miksi nollatut nivelien rotaatiot ovat elintärkeitä. Kaikkien rotaatioiden ja orientaation ollessa yhteneväisiä, niiden keskisten liitosten hallinta on helpompaa. Ideana on, että polven koukistuessa sisäänpäin, apunivel kääntyy vastakkaiseen suuntaan ylläpitäen meshin muodon. Apunivelen rotaatio on siten käänteisarvo polvinivelen rotaatiosta.

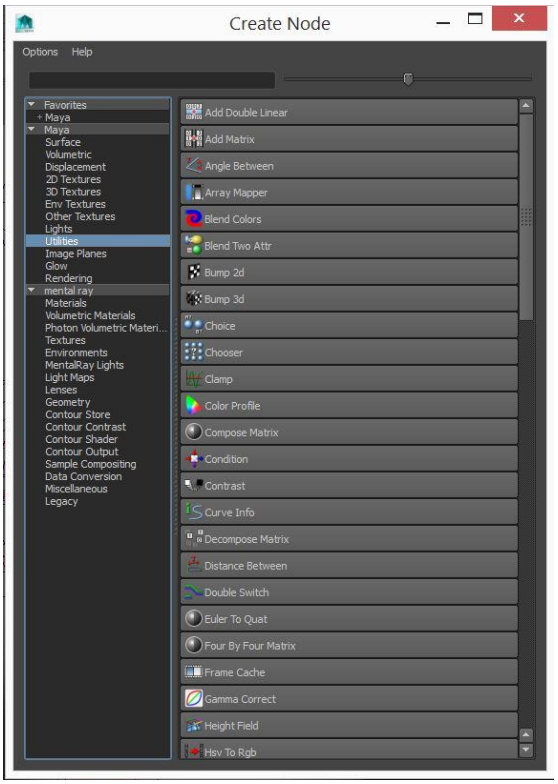
Ensimmäiseksi polvinivelestä otetaan kopio. Tältä kopiolla poistetaan child-nivelet ja se asetetaan hierarkkiseen liitokseen polvinivelen alle. Toivottavaa olisi, että apunivel on täsmälleen samassa positiossa ja orientaatiossa kuin polvinivel. Normaalisti apunivel seuraisi polvinivelen rotaatiota. Apunivel voi kuitenkin itsessään rotaatioitua. Ideana olisi saada apunivel rotaatioitumaan vastakkaiseen suuntaan sitä mukaa mitä enemmän polvinivel koukistuu sisäänpäin.

Tämän vastakkaisen rotaation voi saavuttaa multiply/divide-noodin avulla. Ideana on muuttaa polvinivelen rotaatio vastakkaiseksi ja siirtää tämä rotaatioarvo apunivelen vastaavaan rotaatioon.

Ensin valitaan polvi- ja apunivel. Sen jälkeen avataan node-editor windows-valikon kautta. Valitut nivelet voi asettaa näkyviin node-editorissa painamalla näppäintä, joka on ympyröitynä kuvassa (kuva 38.). Sen jälkeen klikataan hiiren oikealla node-editorin työtilaa kunnes ilmestyy valikko. Valikosta valitaan painike ”create node”. Ilmestyvän valikon vasemmassa reunassa listan alapäässä on sarake ”utility nodes”. Klikkaa sitä ja etsi listasta noodi ”multiply/divide” ja klikkaa sitä. Nyt node-editoriin ilmestyy uusi noodi.



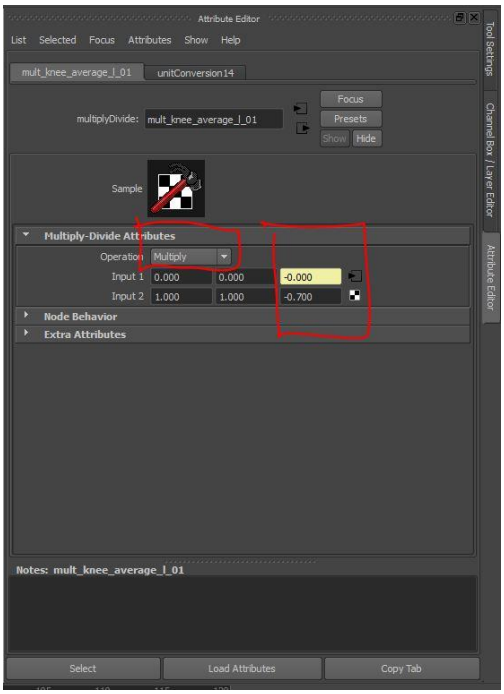
Kuva 38. Polvinivel on vasemmalla puolen ja apunnivel oikealla. Ylhäällä oleva ympyröity näppäin lisää valittujen objektien noodeja node editoriin tarpeen vaatiessa.



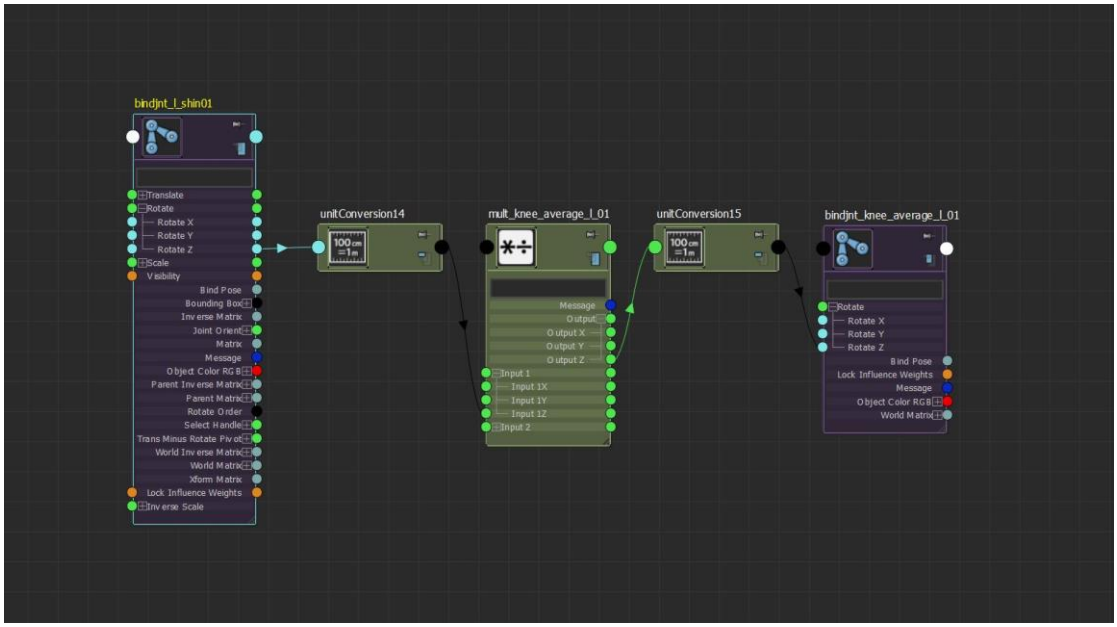
Kuva 39. ”Create-node” -valikko. Noodit ovat lajiteltuna aakkosjärjestyksessä.

Seuraavaksi valitaan polvinivelen noodi. Klikataan sen ”rotateZ” ominaisuuden oikealla puolella olevaa palloa ja vedetään ilmestyvä nuoli multiply/divide noodin vasemmassa reunassa olevaan porttiin. Kun pitää päättää, mihin ominaisuuteen tarkalleen ominaisuus rotateZ liitetään, valitse ”input1Z”. Seuraavaksi klikataan multiply/divide-noodin oikealla puolella olevaa porttia ja valitaan ominaisuus ”outputZ”. Tämä ominaisuus liitetään puolestaan polven apunivelen rotateZ-ominaisuuteen. Polvinivelen ja apunivelen rotaatiot ovat nyt yhdistyneet niin, että polven z-rotaatio ajaa apunivelen z-rotaatiota. Toistaiseksi kuitenkin apunivelen rotaatio seuraa täsmälleen polven z-rotaatiota eikä käänny päinvastaiseen suuntaan toisin kuin tarve olisi.

Ratkaisu piilee valmistetun multply/divide-noodin laskutoimituksessa. Valitse kyseinen noodi ja avaa sen attribute-editor ikkuna. Input1X-portissa näkyy keltaisena polvinivelestä saapuva rotaatioarvo. Ideana on, että input1X-sarakkeessa olevalle luvulle suoritetaan kerto-, jako- tai potenssilasku sillä luvulla, joka vastaavasta input2-sarakkeesta löytyy. Perusasetuksena luku on numero yksi, jolloin muutosta ei tapahdu. Asetetaan lakutoimitukseksi kertolasku. Seuraavaksi input2Z-sarakkeeseen kirjoitetaan luku -0.70. Nyt tapahtuu siten, että polven apunivelen z-rotaatio on yhtä kuin polvinivelen z-rotaatio kertaa -0.70. Apunivel siis kääntyy nyt vastakkaiseen suuntaan kuin polvinivel tällä akselilla. Jos apunivel tuntuu kääntyvän liikaa tai liian vähän vastakkaiseen suuntaan, laskutoimituksen arvoja voi säädellä alaspäin.



*Kuva 40. Input 2 kanavaan on z-kanavan kohdille kirjoitettu -0.70 ja operaatio on kertolasku. Tällöin apunivelen rotaatio on aina polvinivelen rotateZ * -0.70.*

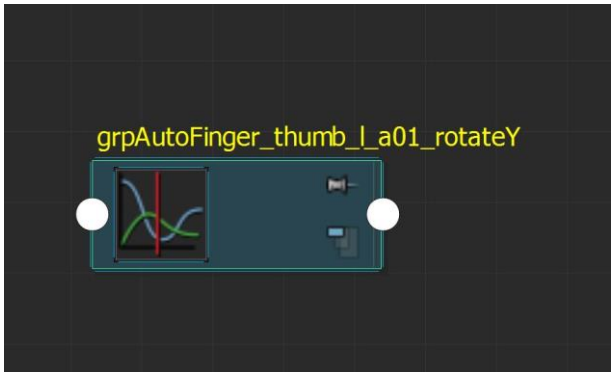


Kuva 41. Valmis noodiketju. Multiply/divide-noodin ja nivelien väliset ovat unitConverse noodeja, jotka ilmestyvät useimmiten automaattisesti kun kaksi ominaisuutta yhdistetään. Ne säätelevät noodien välillä kulkevia yksikköjä ja säätävät ne oikeanlaisiksi.

5.4 Set driven key

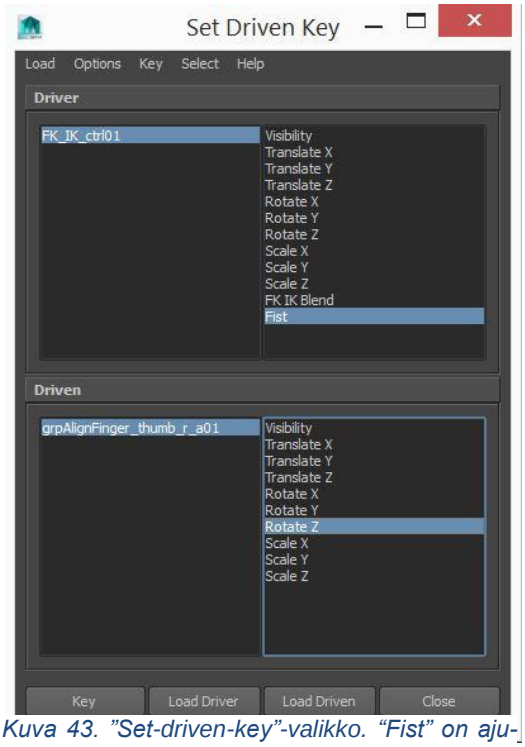
”Set-driven-key”-työkalu muodostaa liitoksen kahden ominaisuuden välille kuten riippuvaisliitoksessa, mutta tekee sen animaatiokäyränä. ”Set-driven-key”-ajurio minaisuuden valmistuksen yhteydessä syntyy erillinen noodi (kuva), joka sisältää liitettyjen ominaisuuksien välisen animaatiokäyrän. Aivan kuten animaatiokäyrää, myös driven key ominaisuutta voi ajaa epälineaarisesti. ”Set-driven-key”-ikkunan saa avattua animate > ”Set Driven Key” valikon kautta.

Koska ”set-driven-key” yhdistelmä on noodi, niin senkin ominaisuuksia pystyy hallitsemaan. Kun outlinerin ”show DAG objects only” -asetus on kytketty pois päältä, luotuja ”driven-key”-noodeja voi valita outlinerin kautta. Kun noodi on valittu, noodin sisältämää animaatiota voi hallita animaatiokäyrän tai attribute editorin avulla (kuva).



Kuva 41. Driven keyn noodi ja kuvio. Noodi on aina nimetty ajettavan objektin ja ajettavan ominaisuuden mukaan.

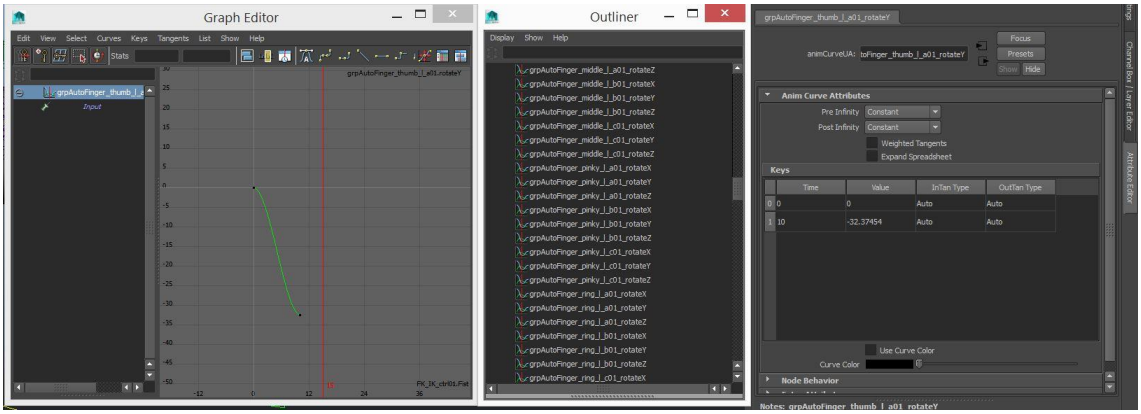
Ajetun arvon rakennus on varsin simppeliä. Ensin valitaan objekti, jota halutaan käyttää ajurina. ”Set-driven-keyn” –valikosta klikataan painiketta ”load driver”, joka tekee objektista ajurin. Seuraavaksi valitaan objekti (tai useampi objekti), jonka ominaisuuksia olisi tarkoitus ajaa ja painetaan ”load driven” painiketta (kuva 43.). Ajuriobjekti sijaitsee nyt yläsarakkeessa ja ajettava objekti alasarakkeessa. Seuraavaksi valitaan ylä- ja alavasemmasta laatikosta objektit, joita on tarkoitus käsitellä. Ylä- ja alaoikeaan sarakkeeseen ilmestyvät molempien objektien ominaisuudet. Klikkaamalla vasemmanpuoleisia objekteja, ne valitaan työtilassakin. Ensin valitaan ominaisuus, jonka on tarkoitus ajaa ja sen arvo asetellaan sopivaan kohtaan. Seuraavaksi valitaan ajettava objekti.



Kuva 43. ”Set-driven-key”-valikko. “Fist” on ajuriominaisuutena ja rotate ajettavana ominaisuutena.

Ajettavasta objektista valitaan ajettavat ominaisuudet. Ajettavien ominaisuuksien arvojen tulisi olla sellaiset, joissa ne haluttaisiin olevan silloin kun ajava on nykyisessä asennossaan. Kun ajettavat ominaisuudet on säädetty, valitaan yläoikeasta sarakkeesta ajuriominaisuus ja alaoikeasta ajettavat ominaisuudet ja painetaan näppäintä ”key”. Ajettavan ominaisuuden pitäisi nyt muuttua punaiseksi. Seuraavaksi valitaan taas ajuriominaisuus ja siirretään se toiseen arvoon. Sen jälkeen palataan ajettaviin arvoihin ja asetellaan ne toiseen arvoon. Jälleen valitaan molemmat ominaisuudet, ajuri ja ajettavat, ja painetaan ”key”. Jos nyt siirrytään ajuriominaisuuteen ja sen arvoja säädetään, ajettavat ominaisuudet muuttuvat sitä mukaa riippuen siitä missä kohdin ”key” näppäintä painettiin. Kutsun tätä suomalaisittain ”merkkaukseksi”, jossa ajettava ominaisuus merkitään näyttämään tiettyä arvoa kun ajuri näyttää tiettyä toista arvoa. Ajettava ominaisuus ikään kuin animoituu sen mukaan miten ajuri päivittyy.

Ajettava ominaisuus voidaan asetella vaihtumaan eri suuntiin aivan kuten animaatiokäyrä voi aaltoilla eri suuntiin. Esimerkiksi ajurin edetessä arvosta 1 arvoon 5, käyrä voi kulkea alaspäin, mutta arvosta 5 arvoon 10, taas ylöspäin. Ajettavien ominaisuuksia voi merkata useammassa kohdassa.



Kuva 44. Kolme ikkunaa, joissa ajettua animaatiota voi tarkata. Noodin sisäistä animaatiota voi säätää attribute-editorissa, tai animaatiokäyrällä.

5.5 Esimerkkityö. Sormikontrollijärjestelmän rakennus

Kun riippuvaisliitokset ja ajuri-animaatiot ovat käsiteltyinä, koen parhaaksi esitellä yhden esimerkkitekniikan näiden hyödyntämisestä sormien animoinnissa. Tässä esimerkissä voimme myös hyödyntää muitakin oppimiamme asioita, kuten sidoksia ja esittää käytännön esimerkki set-driven keyn teosta.

Alla on kuva mallintamani hahmon, Pacalin, kädestä ja nivelistä, jotka sormiin on aseteltu. Yksityiskohtana mainitsen, että sorminivelet rakentuvat kahdesta peräkkäisestä nivelestä. Tämä siksi, että sormien deformaatio olis mahdollisimman teräväkulmainen

Animoinnin monipuolisuuden kantilta on eduksi, että jokaisella sorminivelellä on oma henkilökohtainen, nollattu ohjain. Kaikkien sormien hallinta ja ohjaus tiettyihin asentoihin, esimerkiksi nyrkkiin, voi kuitenkin viedä paljon aikaa. Yksi keino tähän on automatisoidut animaatiot. Tällä tarkoitan esimerkiksi yhtä ominaisuus-kanavaa channel editorissa, jonka muuttuessa luvusta 0 lukuun 10 toteuttaa kokonaisen animaation, jossa käsi puristuu nyrkkiin. Tämä tapahtuu luonnollisesti siten, että tästä automaatiokanavasta on tehty sormien ajuri.

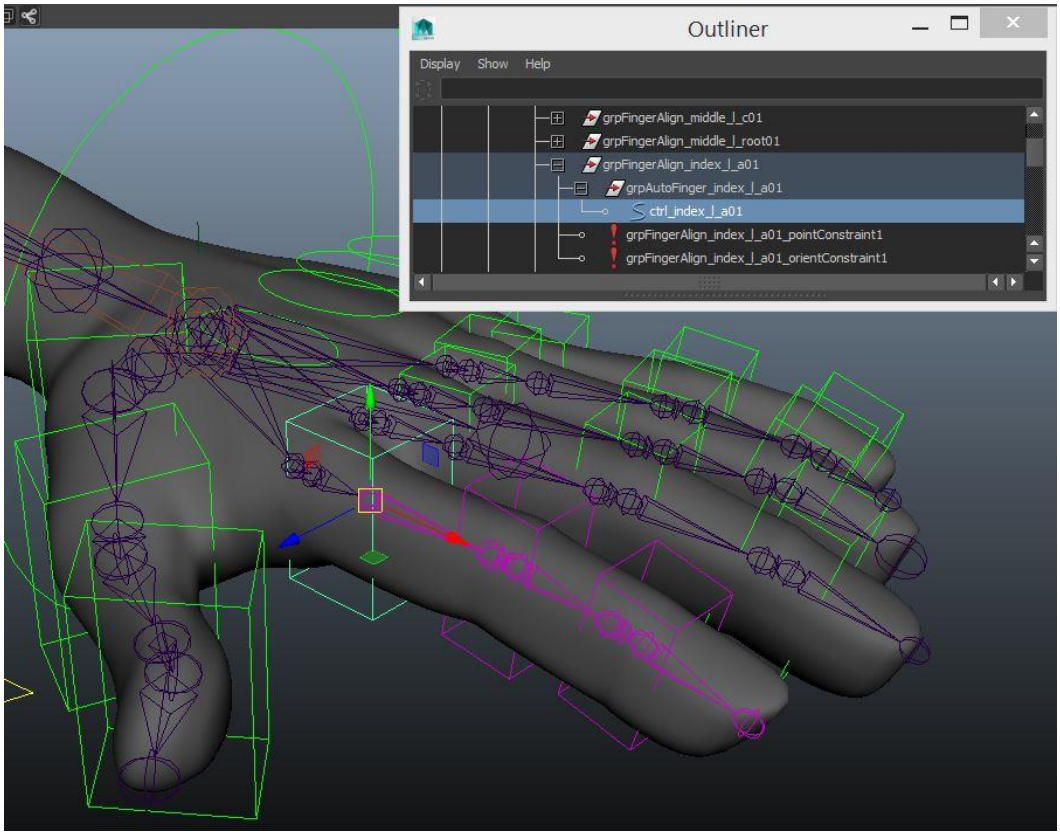
Asian voi tehdä yksinkertaisesti ja vain valita haluttu kanava ajuriksi ja sormen ohjaimet ajettaviksi ja asetella sormet niin, että ajurin ollessa luvussa 10, sormet ovat nyrkissä, ja luvussa 0 kämmen on normiasennossa.

Tämä muodostaa kuitenkin sen ongelman, että ohjainten kontrollit lukittuvat ajurin kanavaan. Objektin ominaisuus, jota toinen ominaisuus ajaa, ei ole enää itsessään hallittavissa. Kaiken lisäksi ohjainten rotaatiot olisivat ei-nollattuja. Tämä olisi lopulta hyvin pulmallinen tilanne animaattorille.

5.5.1 Sormien ohjainjärjestelmän rakennus

Keino ohjainten lukituksen välttämiseksi löytyy ryhmistä. (Annan kunnian tämän tekniikan löydöstä Digital Tutorsin sivuston videosarjalle: ”Gain the knowledge you need to enchancing your rigs further”, 2013, jossa tätä tekniikkaa sovellettiin.) Jos muistelemme luvussa 4 käsittelemiämme ryhmiä (groups) ja esimerkkiä, jossa objekti oli hierarkioitu kahden keskenään hierarkkisen ryhmän alle, saamme tästä vastauksen pulmaan. Ideana on asettaa jokainen sormi-ohjain kahden ryhmän alaisuuteen (ryhmät kannattaa nimetä asianmukaisesti). Kun ryhmiä luodaan, on tärkeää varmistaa ensin, että luotu ohjain on globaalissa keskipisteessä. Tällöin molempien luotujen ryhmien, ennen kaikkea hierarkian ylimmän ryhmän napa, on valmiiksi asettautuneena linjaan objektin sijainnin kanssa.

Seuraavaksi ohjain tulee liikuttaa sorminivelen kohdille. Ohjainta tulee liikuttaa ennen kaikkea niin, että liikutetaan hierarkian ylintä ryhmää. Tällöin keskimmäinen ryhmä sekä ohjain säilyttävät nollatut ominaisuudet. Ylimmän ryhmän tulisi myös osoittaa rotaatioiltaan samaan suuntaan kuin nivel, johon ohjain kiinnitetään. Yksi keino orientointiin on valita ensin ryhmä ja sitten (shift alas painettuna) nivel ja hierarkioida ryhmä nivelen alaisuuteen (näppäin P). Tällöin ryhmän ominaisuudet määräytyvät sen mukaisiksi kuin mitä ne ovat parentin positiosta lukien. Valitse kaikki ryhmän ominaisuudet ja nollaa ne (translaatio ja rotaatio). Ryhmän pitäisi siirtyä täsmälleen nivel kohdalle ja sen akselien suuntautua kuin kyseinen nivel. Ryhmän orientointi nivelen mukaiseksi on tärkeää keskimmäisen ryhmän ja ohjaimen kannalta, koska näiden kahden orientaation tuli olla yhdenmukainen. Lopuksi valitse ylin ryhmä ja poista se hierarkiasta (shift+P). Ylimmän ryhmän ominaisuuksien pitäisi nyt määräytyä globaalin keskipisteen mukaan (kuva 45.).



Kuva 45. Ryhmitetyt sormiohjaimet. Kuvassa on jo myös liitetty tarvittavat sidokset.

5.5.2 Sorminivelien yhdistäminen ohjaimiin

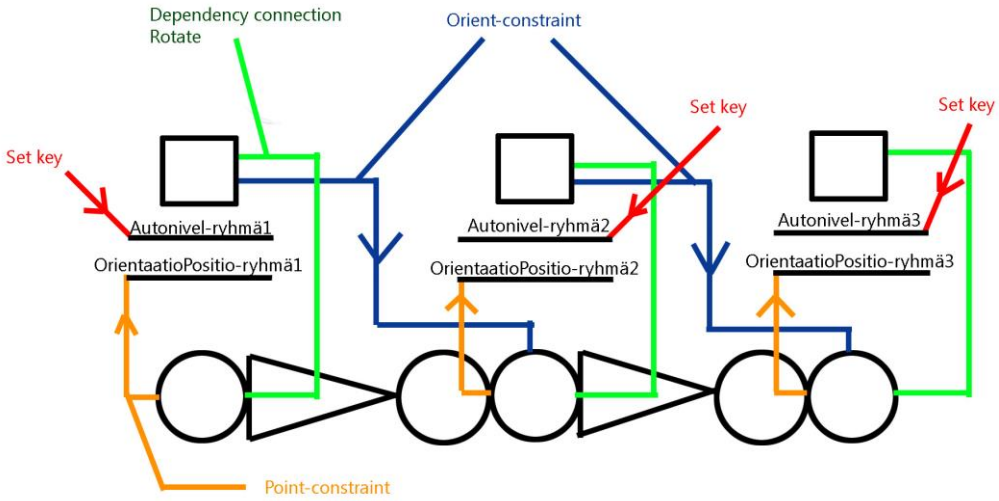
Tämä sama prosessi tehdään muillekin sormelle (digital tutorsin videossa ryhmien rakennus ja sijoitus täsmälleen ohjaimen alle, oli automatisoitu koodilla). Seuraavaksi sorminivelet täytyy tietenkin saada liitettyä ohjain systeemiin. Aluksi itse ohjain-objekteista tehdään riippuvaisliitos ohjattavaan niveleen connection editorin kautta. Tee ohjaimesta ajuri ja nivelestä ajettava. Valitse sen jälkeen rotaatioakselit, joiden mukaan tahdot sormia liikuttaa ja liitä ne nivelen vastaaviin akseleihin. Tässä kohtaa nivelien rotaatio on nyt sama kuin ohjaimen rotaatio. Jos yhdistelmän aikana nivel liikahtaa, se tarkoittaa, että ohjaimen ja nivelen orientaatiot eivät olleet yhteneväiset. Tässä tapauksessa kannattaa tarkastaa onko ohjain ja ryhmät tosiaan samassa linjassa nivelen kanssa. Tämä vuoksi nivelien rotaatioiden tulee olla nollattuja, sillä ohjainten ollessa myös nollattuja, todennäköisyydet virheille ovat alhaisemmat. Toinen syy voi olla, että ylin ryhmä ei ole orientoitunut oikein niveleen nähden, ja siten keskimmäisen ryhmän ja ohjaimen orientaatio olisi myös vääristynyt.

Asettele muutkin sormen nivelten ohjaimet samoin. Seuraavaksi tulisi pitää huolta siitä, että sormien ohjaimet seuraavat niveliä kämmenen liikkuessa, sillä vaikka ohjain on liitetty nivelen orientaatioon, itse ryhmät eivät ole liitoksissa kämmeneen. Ohjain siis kyllä rotaatioi sormea, mutta ohjain ei seuraa nivelen positiota. Ratkaisuna on ohjaimen ylimmän ryhmän sitominen vastaavaan niveleen piste-sidoksella. Vaikka ohjain liikuttaakin vastaavan nivelen rotaatiota, se ei kuitenkaan vaikuta nivelen translaatioon, ja aiheuttaisi silloin syklisen reaktion ylimmän ryhmän kanssa, joka puolestaa liikkuu nivelen position mukaan. Sido muidenkin ohjainten ryhmät samalla tavoin vastaamaansa niveleen.

Kun nyt käännetään rystystä lähinnä olevaa niveltä, se saa sormen muut ohjaimet seuraamaan liikettä. Sormen ulommat nivelet eivät kuitenkaan seuraa niinkään itse ohjainta vaan niveliä, joita rystysiä lähinnä oleva ohjain liikuttaa. Toinen huomioitava seikka on, että muut nivelet eivät käänny rotaation mukaan rystysiä lähinnä olevan ohjaimen kääntyessä, vaan osoittava suoraan kuten nollapositiossa. Tämä johtuu siitä, että nivelien ollessa sidottuina ohjaimeen, ne mukautuvat vain ohjaimen liikkeeseen. Ohjain taas seuraa hierarkian ylempien ryhmien liikettä. Jotta ulommat sorminivelet saataisiin mukailemaan orientaatiota, niiden ohjainten ylimmät ryhmät on orientaatioiltaan sidottava niitä edeltävän nivelen ohjaimen orientaatioon. Tämä onnistuu orientaatio-sidoksen (orient constraint) avulla. Ennen sidoksen asettelua on syytä huomioida, että ohjaimen ja sitä seuraavan ohjaimen ylimmän ryhmän rotaatiot eivät ole samat. Näiden arvojen välillä on

siis poikkeama (offset). Jotta tästä seuraava ongelma voitaisiin välttää, ennen sidoksen luomista tulisi asettaa päälle sen poikkeamia ylläpitävä omianisuus (maintain offset), joka löytyy sidoksen säätövalikon ylälaidasta. Kun sidos nyt tehdään, ohjain huomioi kääntyvän nivelen valmiina olevat rotaatiot ja päätyy ainoastaan lisäämään oman rotaationsa tähän rotaatioon. Tässä on toinen syy sille, miksi nivelketjujen orientaation pitäisi olla yhdenmukainen. Y-akselit osoittavat kaikki tiettyyn suuntaan ja z-akselit samoin. Kun ohjaimet lopulta liitetään näihin niveliin, varmistetaan se, että ketjutettavat ohjainsarjat ovat yhdenmukaiset rotaatioiltaan.

Kun nämä prosessit ovat suoritettuina jokaiselle nivelelle, sormien pitäisi jouhevasti seurata. Kannattaa ottaa selkoa vielä mahdollisista ongelmista, sillä automatisoidut sormiliikkeet edellyttävät toimivaa sormien ohjainjärjestelmää (kuva 46).



Kuva 46. Kaavio sorminivel-systeemin rakentumisesta. Alareunassa ovat sormen nivelet kämmenestä sormen päähän. Kuutiot ovat ohjaimia ja viivat niiden alla kuvaavat niiden vastaavia. Vaikka OrientaatioPositio-ryhmä on alin, niin hierarkkisesti se on ylin ohjainten järjestelmässä.

Vihreät viivat kuvaavat riippuvaisliitoksia, jotka lähtevät ohjaimen rotaatiosta vastaavaan niveleen. Siniset viivat ovat orientaatiosidoksia, joissa ohjainten rotaatio ohjaa seuraavan ryhmän nivelen rotaatiota. Oranssi viiva kuvaa nivelestä OrientaatioPositio-ryhmää ohjaavaa pistesidosta.

Punaiset viivat kuvaavat Autonivel-ryhmään tulevaa ”set driven key” animaatiota.

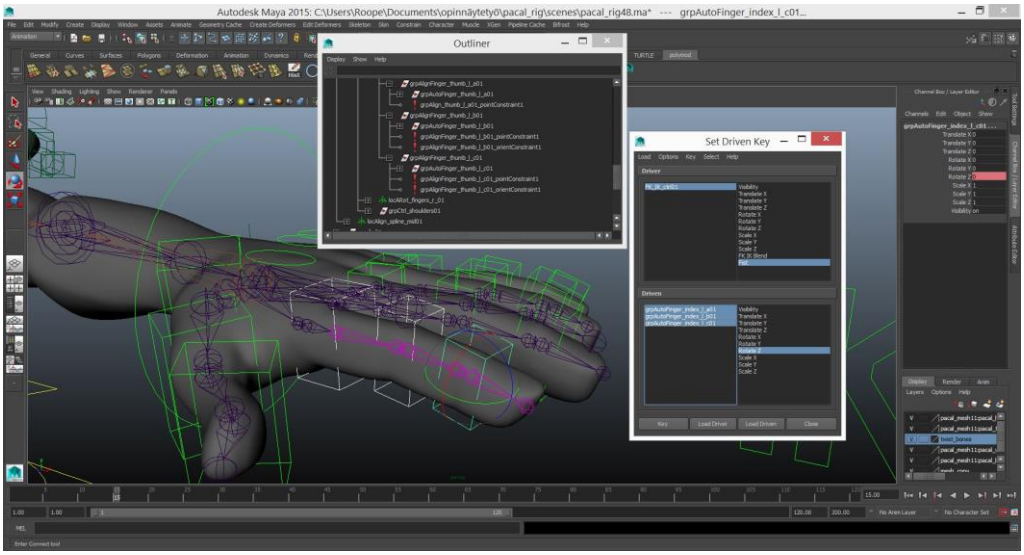
5.5.3 Automaatiokontrollien valmistus

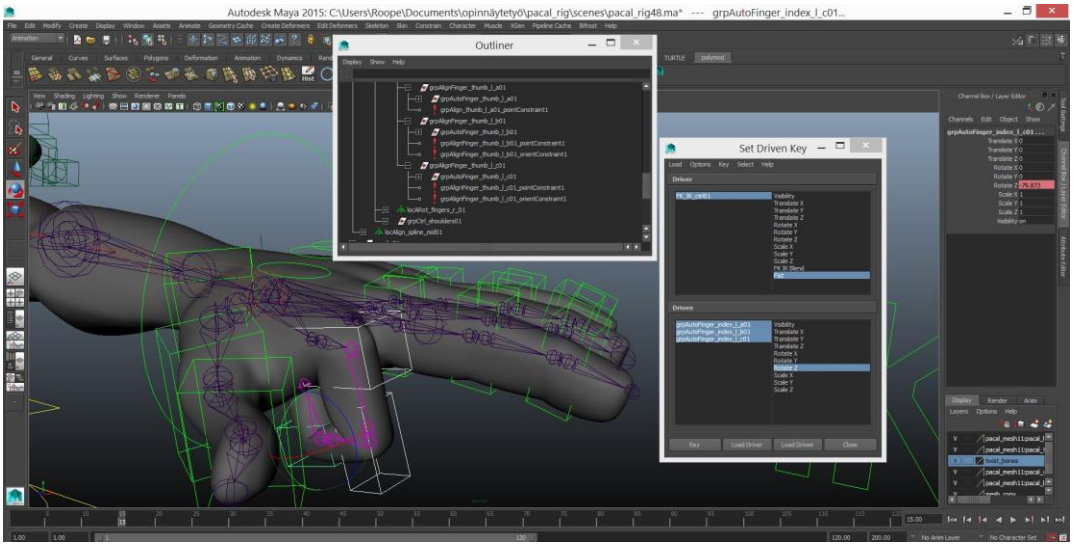
Automaatioiden valmistuksessa hyödynnetään “set-Driven key”-työkalua. Ensimmäisenä valmistetaan ohjainkanava erillisille sormien asennoille. Tässä tapauksessa tavoiteltava asento on nyrkki. Ei ole varsinaista sääntöä sille mihin ohjaimeen tämän uuden ominaisuuden, mutta suositeltavaa olisi pistää se mahdollisimman lähelle kättä. Nimetään uusi ominaisuus lyhyesti sanalla ”fist” ja asetellaan sen miniarvoksi 0 ja maksimiarvoksi 10. Arvossa nolla kämmen on normipositiossa ja arvossa 10 kämmen on nyrkissä.

Tee objektista, jossa ajuri-ominaisuus on, ajuri ja sen jälkeen valitse sormien ohjainnivelten keskimmäiset ryhmät. Tässä piilee syy sille, miksi ryhmiä on kaksi. Keskimmäisen ryhmän on tarkoitus toimia ajettavana objektina ”set-Driven key”:n kautta rakennettavalle ajurille. Keskimmäinen ryhmä liikuttaa edelleen ohjainta, joka kuitenkin säilyttää itse nollatut transformaatiot, koska itse ohjaimen arvoihin ei kosketa. Jos asiat ovat sujuneet oikein, keskimmäisten ryhmien transformaatioiden tulisi olla nolla.

Lopulta ainoa jäljellä oleva asia on keskimmäisten ryhmien rotaation ja ajuriominaisuuden arvojen yhdistäminen. Suosittelen, että ensin ajuri asetellaan nollapositioon. Sen jälkeen jokaisen keskimmäisen ryhmän kohdilla painetaan ”set-key”. Keskimmäisiä niveliä ei vielä tarvitse liikuttaa, koska ajurin nollapositio ei aiheuta muutoksia keski-ryhmiin. Tällä tavoin kuitenkin ajettaville ominaisuuksille on luotu tarvittava ”set-driven key” noodi, joka sisältää ominaisuuden ja kyseisen ajurin keskeiset animaatiot.

Seuraavaksi ajuri-ominaisuus siirretään maksimiarvoon. Valitse taas keskimmäiset ryhmät ja muuta niiden rotaatioita niin, että sormet muodostavat nyrkin. Paina jälleen ”set key” muutettujen rotaatioiden ollessa ”driven key” –ikkunassa valittuna. Kun ”fist”-ominaisuutta säädellään, kämmenen pitäisi avautua ja sulkeutua riippuen ”fist”-ominaisuuden arvosta. On kuitenkin myös mahdollista, että muutosprosessi avoimen kämmenen ja nyrkin välillä ei ole aina luonteva. Tässä tapauksessa on vain siirrettävä ajuri sellaiseen arvoon, jossa ongelmia on ja liikutella keski-ryhmien rotaatioita ja painaa ”key” kun asento on mieluinen (kuva 48.).





Kuva 48. Automatisoitu etusormi. Huomaa hierarkiarakenne outlinerissa.

Erityisenä etuna ryhmien automatisoinnissa on, että niiden transformaatio ei näy perusohjainten toiminnassa vaan ne ovat yhä nollattuja. Niihin ei myöskään kohdistu hierarkian lisäksi mitään muita liitoksia hierarkian lisäksi ja täten niiden jokainen ominaisuus on käytettävissä. Tässä kohtaa kerronkin siis vielä yhden suosituksen näkyvien ohjainten valmistuksessa: Varsinaisia ohjaimia kannattaa pyrkiä välttämään tekemästä sidosten, tai liitosten alaisiksi ja pitää ne ennen kaikkea ajureina. Sidoksilla on se seuraus, että idottuja objektin ominaisuuksia ei enään itsessään pysty hallitsemaan ja siten rajoittaa osin ohjaimen tehtävää.

Ajuriobjektiin voi tarvittaessa lisätä enemmän ominaisuuksia eri asennoille sitä mukaa kun niille tulee tarvetta.

5.6 Tuplatransformaatio

Liitoksien rakennuksessa on tärkeää huomioida tuplatransformaation riski. Tuplatransformaatiossa objektin tai komponentin ominaisuuteen kohdistuu muutos, joka vaikuttaa kaksinkertaisesti. Objektiin, johon vaikuttaa jo yksittäisen ominaisuuden transformaatio, kohdistuu toinenkin transformaatio samaan ominaisuuteen. Tämä ongelma ilmenee etenkin hierarkioiden ja riippuvaisliitosten tai sidosten yhdistyessä. Child seuraan parentin liikettä luontaisesti, mutta parentista saattaa lähteä myös riippuvaisliitos, joka vaikuttaa myös childin ominaisuuksien muuttaen niitä kahdesti. Tuplatransformaatio ei itsessään ole aina pahasta, mutta useimmiten se on kutsumaton vieras ja aiheuttaa ennakoimattomia muutoksia. Useimmiten Maya varoittaa oikeassa alareunassa olevassa viestipalkissa keltaisella varoituksella mahdollisista ennakoimattomista liikkeistä. Jos kyseessä on paradoksaalinen liitos, useimmiten toimintoa ei silloin pysty edes suorittamaan (Autodesk 2014a.)

Otetaan esimerkkinä lokaattori-A, jonka rotaatio ajaa objektin-A rotaatiota. Lokaattori-A:n orientaatiota ajaa puolestaan lokaattori-B:n rotaatio. Normaalisti lokaattori-B:n rotaatio kääntäisi lokaattori-A:ta ja lokaattori-A:n rotaatio siten heijastuisi objekti-A:han. Toisin sanoen syntyy ketju: lokaattori-B.rot=lokaattori-A.rot=objekti-A.rot.

Tässä tapauksessa ominaisuuksien välinen ketju kulkee lineaarisesti. Ongelma syntyy kuitenkin silloin, jos lokaattori-B:n rotaatio liitetään myös objekti-A:n rotaatioon. Tällöin objekti-A:n transformaatio olisi yhtä kuin lokaattori-A:n ja lokaattori-B:n transformaatio yhteenlaskettuna, koska lokaattori-B:n ajaa myös lokaattorin-A:n transformaatiota.
Tämä ei tietenkään ole ongelma, jos tämä on rigin varsinainen tarkoitus, mutta useimmiten selkeä lineaarinen komentojärjestys on helpompi hallita. Tuplatransformaation ongelmana on, että siinä liitoksien välinen syy-seuraus-suhde hämärtyy. Tämä lopulta tekee virheiden havaitsemisesta haastavaa. Ylipäätänsä komentoketjun seuraavien laskelmien teko voi hankaloitua, koska seuraavan liitoksen laskelmat voivat käydä arvaamattomiksi.

6 Meshin hallinta – Deformaatiot

Tämän luvun aihe osaltaan limittyy edelliseen, koin tarpeelliseksi jakaa sen omaksi luvukseen. Tässä luvussa käsittelen vielä deformerit. Edellisissä luvuissa käsittelemämme

tekniikat ja prinsiipit koskivat pitemmälti transform noodeja, ja muita noodeja, jotka omasivat samoja ominaisuuksia. Deformerit poikkeavat tässä suhteessa siinä, että ne vaikuttavat shape noodiin.

Nimensä mukaisesti deformerit muokkaavat shape noodien sisältämien kontrollivertexien (control vertex) rakennetta. Normaalisti shape noodin komponentteja ei pysty muokkaamaan normaalin transformaation tavoin. Komponenttien vertexien sijainti on suhteessa aina sama riippumatta siitä mihin transform noodi, johon shape-noodi on parentoitu, liikutetaan.

Käsittelen deformereita viimeisenä, koska koen niiden tuomien efektien olevan päällimmäisenä rigin rakennushierarkiassa. Se ei kuitenkaan tarkoita, että ne ovat merkityksettömiä. Päinvastoin.

Deformerit ovat omiaan hienostuneempien meshin muokkausten tekemiseen. Osat deformereista ovat käyttökelpoisia myös mallinnustyössä. Deformaattorit ovat itsessään myös noodeja. Näiden noodien ominaisuuksien hallinta tekee siten niistä käyttökelpoisia rigeihin.

Deformereiden kirjavuudesta huolimatta, niillä on useimmiten varsin yhteneväinen toimintamekanismi. Jokainen deformerin valmistuksen yhteydessä syntyy tweak-noodi. Tweak noodi liittyy deformerin input porttiin ja siten toimii ennen itse deformeria. Koska deformeri säätelee kontrollipisteitä kuin transform-noodi, deformerin on ensin laskelmoitava pisteiden sijainti ja liikkuminen, jotta deformeri voisi tehdä tehtävänsä.

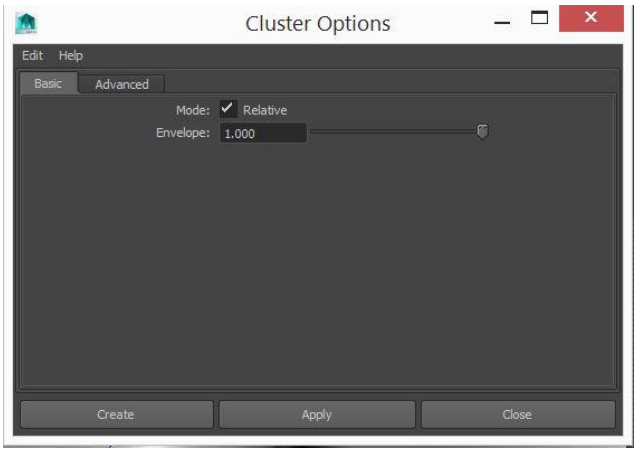
Deformereita on useita erityyppisiä, mutta käsittelen niistä vain kaksi olennaista esimerkkiä.

6.1 Cluster

Cluster handle on transform objekti, johon pystyy liittämään control vertexejä. Control vertexit ovat ikään kuin koukkuja, jotka on mahdollista kiinnittää vertexeihin. Clusterilla on samat perusominaisuudet kuin transform noodilla.

Clusteri valmistetaan valitsemalla ensin halutut control vertexit. Sen jälkeen ”create deformers”-valikosta valitaan ”cluster” -komento. Tämä muodostaa valittujen control vertexien keskelle “c”-kirjaimen, jolla on samat liikutteluominaisuudet kuin transform -noodilla.

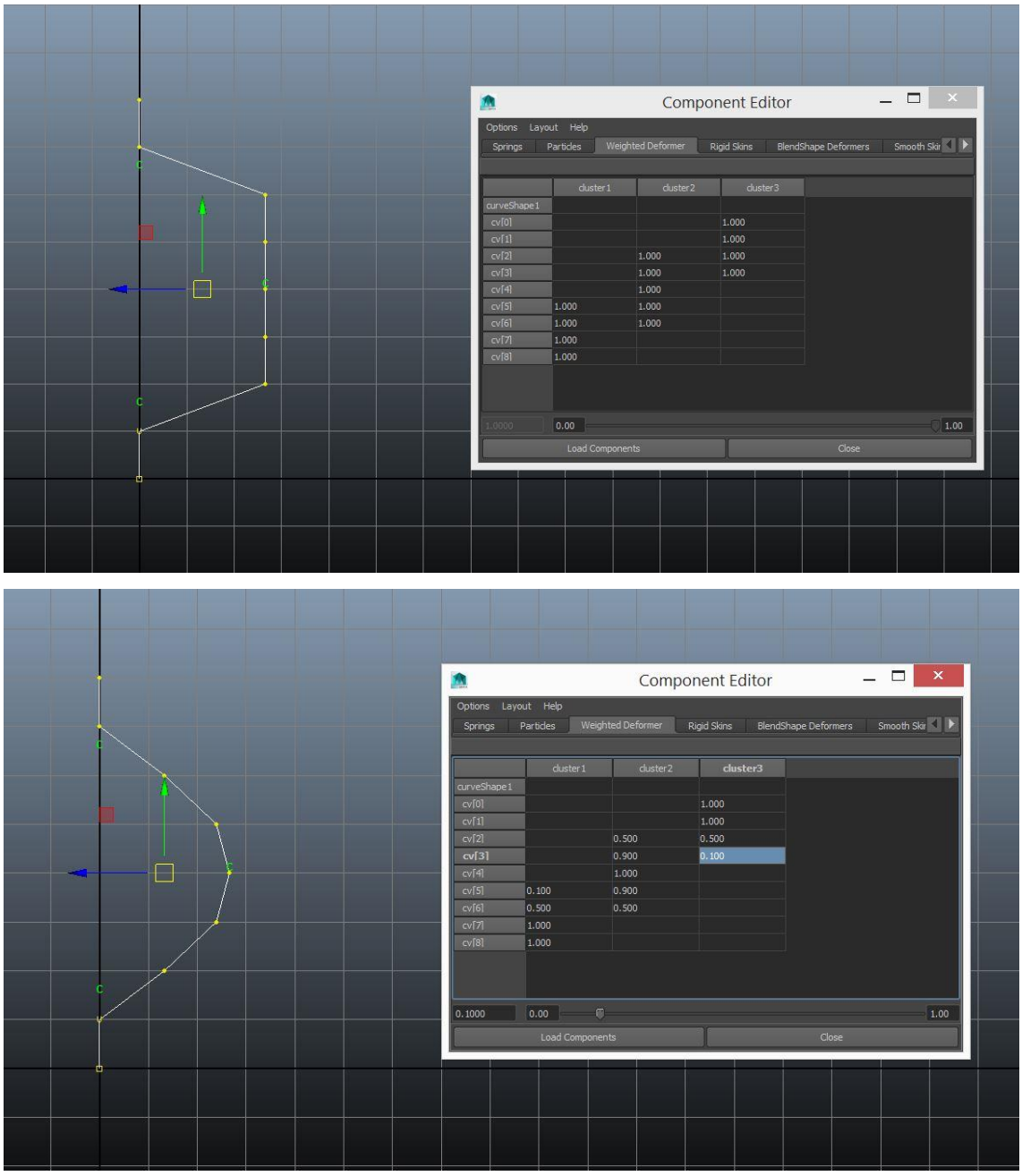
Clusterin voi asettaa hierarkiaan tai sidokseen kuten tavallisen objektin. Yhdelle control-vertexille on mahdollista asettaa useita clustereita. Jos vertexillä on useita clustereita, niin jokaisen clusterin painoarvo vertexeille on aina sata prosenttia. Vertex seuraa siis tässä tapauksessa täysin kummankin clusterin liikettä. Tämä poikkeaa siten skinclusterin toiminnasta meshin sitomisessa, jossa yhden vertexin painoarvo jakautuu automaattisesti niin, että useamman painoarvon yhteistulos on sata prosenttia.



Kuva 49. Clusteria rakentaessa perusoptiona on relative-moodin käyttö. Kun relative on kytketty päälle, cluster deformoi meshiä vain silloin kun clusterin parentia itsessään liikutetaan.

Joissain tapauksissa on kuitenkin välttämätöntä säätää useamman clusterin omaavan vertexin painoarvoja siten, että toisen clusterin painoarvo on tietyn vertexin kohdilla pienempi kuin toisen. Tämä onnistuu component editoria käyttämällä. Component editor pystyy säätelemään erinäisiin vertexeihin kohdistuvia painoarvoja ja toimintoja. Valitse ensin clusterit, joiden painoarvoa haluat säätää ja valitse sitten component editorista sarake ”weighted deformers”. Oikeassa näet valittujen vertexien numerot ja niiden oikealla puolella kaikki ne deformerit, jotka vaikuttavat vertexien painoarvoihin. Painoarvoja saa mutettua simppelisti kirjoittamalla halutun painoarvon vastaavan vertexin kohdille.

Jos haluat saada käyrän muutokset mahdollisimman sulaviksi, kannattaa useamman clusterin omaavissa vertexeissä pitää arvot sellaisina, että ne yhteenlaskettuna muodostavat luvun yksi. Kuvasarjassa 50. on esimerkki säätelemättömästä ja säädellystä käyrästä. Keskimmäinen cluster on osittain siirretty sijoiltaan ylemmän ja alemman clusterin ollessa paikallaan.
Ensimmäisessä kuvassa keskimmäisen clusterin arvot koko hallinta-alueeltaan on yksi, jolloin deformaatio on varsin suoraviivainen. Alemmassa kuvassa vertexien painoarvoja on säädelty niin, että keskimmäisen clusterin muodostama deformaation on hienostuneempi.



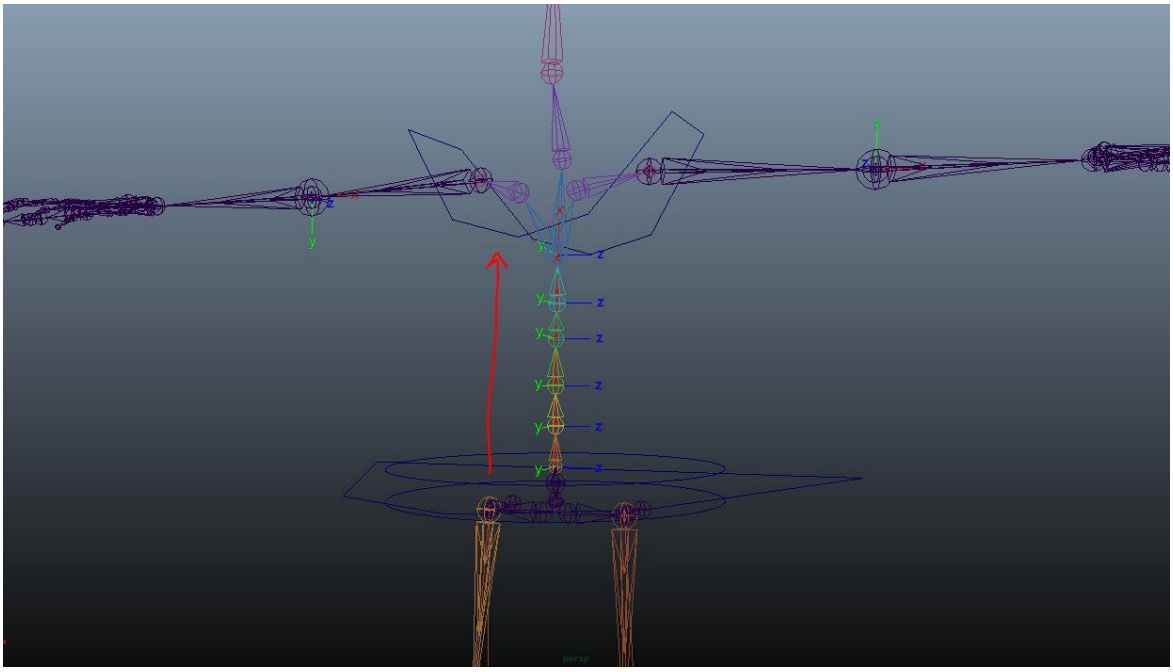
Kuva 50. Ylhäällä on säätelemätön nurbs-käyrä kun keskimmäinen clusteri on siirretty sivuun. Alemmassa kuvassa on sama tilanne, mutta clusterien painoarvot vertexeihin on muutettu.

6.1.1 Käytännön esimerkki, ik-spline-ketjun hallinta

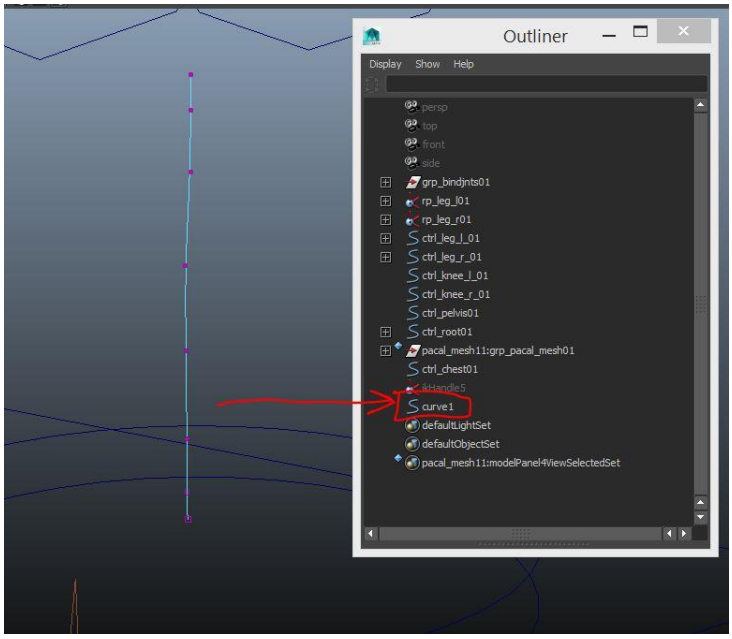
Neljännessä luvussa mainitsin jo ik-spline ketjun, jonka muoto määräytyy sen mukaan miten siihen liitetty käyrä muotoutuu. Ongelmana on, että nurbs-käyrän muokkaus tavallisen transform-noodin tapaan ei ole yhtä simppeliä. Jotta käyrän pisteiden paikkaa voi hallita, siihen tarvitaan deformereita. Clusterit ovat simppeli ja kätevä apu tässä asiassa.

Sovelletaan nyt käytännössä clustereita ja samalla katsastamme ik-splinen toiminnan tämän esimerkin tiimoilta. Asettelemme ik-splinen Pacalin selkärangaksi. Tavoitteena olisi, että selkäranka taipuisi keskeltä.

Ensin valmistetaan ik-spline ketju. Ennen sen valmistusta on hyvä ottaa selvää, että nivelet osoittavat oikeaan suuntaan. Ik-splinen tapauksessa on tärkeää, että nivelketjun toiseen niveleen osoittava akseli on x-akseli, sillä vielä toistaiseksi ik-splinen edistyneemmät ominaisuudet tekevät laskelmansa sen pohjalta (Rigging Dojo, 2014). Näitä ik-splinen edistyneempiä ominaisuuksia en kuitenkaan käsittele tässä tutkielmassa.



Kuva 51. Käsiteltävä nivelketju. Huomaa, että jokaisen selkärangan nivelen y- ja z-akseli osoittaa samaan suuntaan. Tällöin jokaisen nivelen rotaatioakselisto on samanlainen.



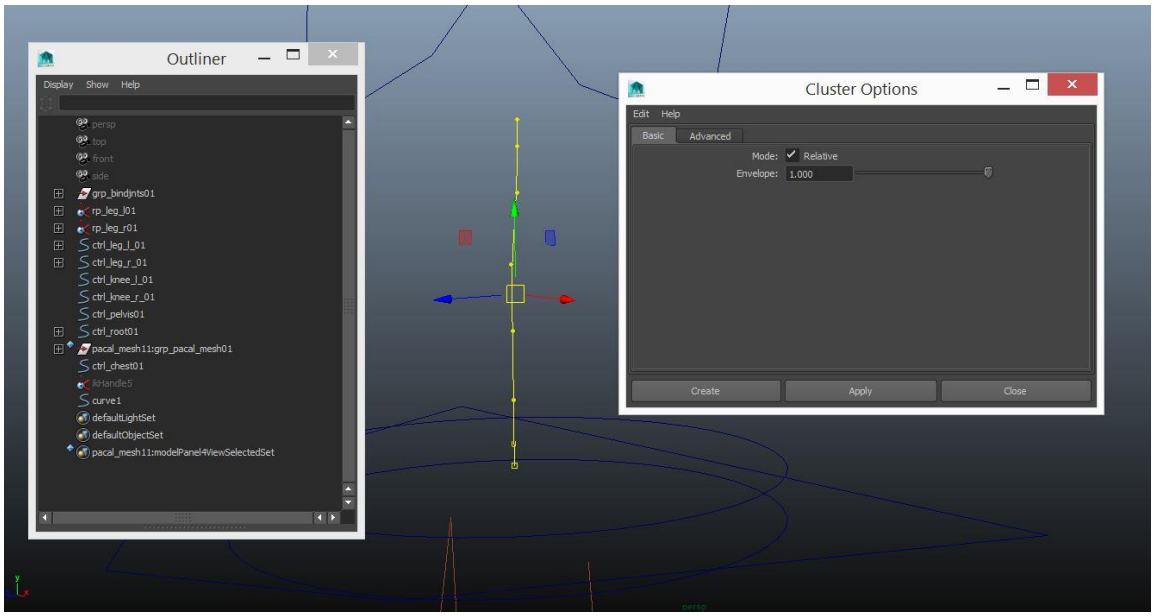
Ik-ketju rakennetaan alaselän nivelestä rintakehän niveleen (kuva 51). Ik ketjun pidikkeellä itsellään ei ole paljoa käyttöä splinen liikkeen pohjautuessa pääasiassa ik-ketjun luomisen yhteydessä syntyneen nurbs-käyrän

Kuva 52. Nivelketjuun muodostunut nurbs-käyrä. Ik-splinen työkaluvalikosta löytyy optio ”simplify curve”, joka minimoi syntyvien kontrolli vertexien määrän. Muussa tapauksessa kontrollivertexejä syntyy useimmiten saman verran kuin nivelketjussa on niveliä.

liikkeeseen. Jotta nurbs-käyrän muotoa voi hallita, sen kontrolli-vertexit (control vertex) täytyy asetella clusterien ohjattaviksi. Jos ik-spline työkalun ”Auto Parent Curve” ominaisuus on päällä, curve löytyy parentoituna sen nivelen alle, jonka alainen nivelketjun alkunivel on. Muussa tapauksessa curve on hierarkioituna työtilaan (kuva 52.).

Ideana on ohjata selkärankaa kolmesta kohdasta: Alhaalta, ylhäältä ja keskeltä. Tämä tarkoittaa, että vertexejä tulee pystyä ohjaamaan kolmesta kohtaa, joka puolestaan edellyttää kolmea clusteria.

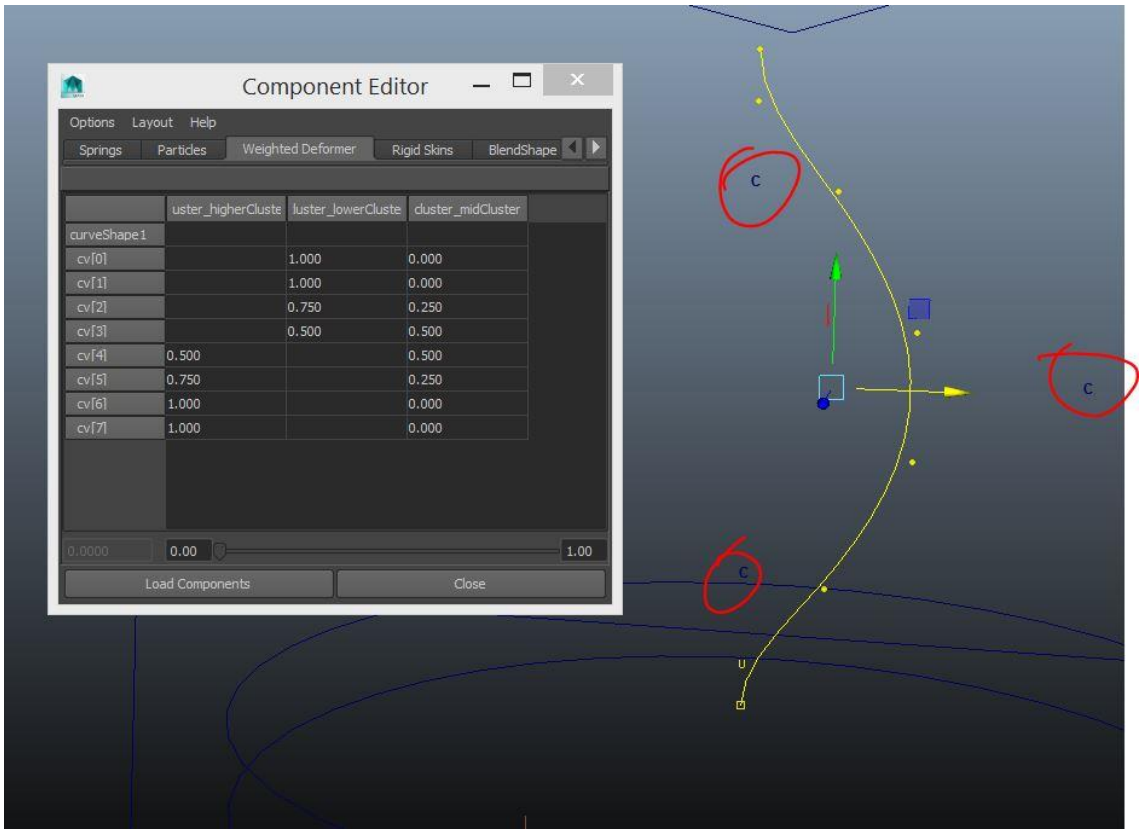
Ensimmäisenä valitaan varmuuden vuoksi jokainen vertexi nurbs-käyrästä. Tämän jälkeen valmistetaan cluster. Cluster tullaan asettamaan ohjain-käyrän alaisuuteen, joten relative-valinnan tulisi olla päällä. Seuraavaksi valitaan neljä ylintä vertexiä ja valmistetaan ylempi cluster. Sama toistetaan neljään alempaan vertexiin. Seuraavaksi valitaan ylempi cluster ja siirretään sen napa ylimmän vertexin kohdille (näppäimet d + v pohjassa). Sama toteutetaan alemmalle clusterille. Nyt splinen käyrällä on tarvittavat ohjaimet ylä, ala keskikohdassa.



Kuva 53. Keskimmäisen clusterin alaisuuteen asetetaan ensin kaikki käyrän vertexit. Clusteri ilmestyy siihen kohtaan, missä translaatiotyökalu nyt on.

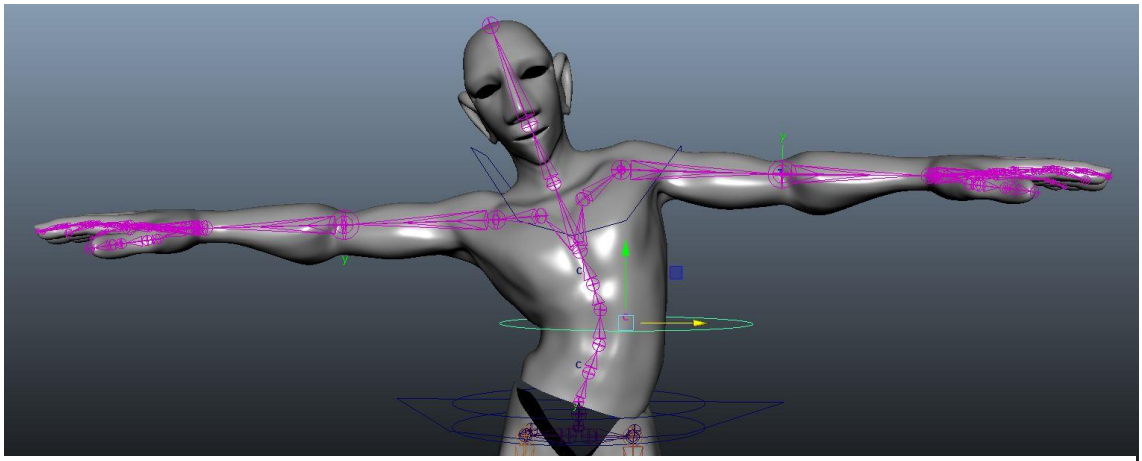
Ongelmana on nyt kuitenkin se, että keskimmäinen cluster liikuttaa koko käyrää, jolloin selkärangan taipuminen ei ole kovin luontevaa. Tässä kohtaa clustereiden painoarvoja vertexeihin tulee säätää component editorin kautta. Säädetään painoarvot niin, että keskimmäisellä clusterilla ei ole lainkaan painoarvoja kahdessa ylimmässä ja alemmassa

vertexissä. Nämä vertexit ovat tällöin joko ylemmän tai alemman clusterin hallinnassa. Loput vertexit säädellään siten kuin kuvassa näkyy (kuva 54.).



Kuva 54. Säädellyt clusterit.

Lopputuloksena keskimmäisen clusteri taivuttaa selkää luontevammin. Lopuksi valmistetaan kontrollit clustereita varten. Asettele ohjaimet täsmälleen vastaavien clustereiden kohdille (v-näppäin alaspainettuna samalla kun objekteja liikutellaan). Tämän jälkeen sido clusterit vastaaviin ohjaimiin piste-sidoksella ja raakile selkärangan ohjainsysteemi on kasassa. Puhun raakileesta, sillä ohjainsysteemien asettelu ei ole välttämättä kaikkein toimivin (kuva 55.).

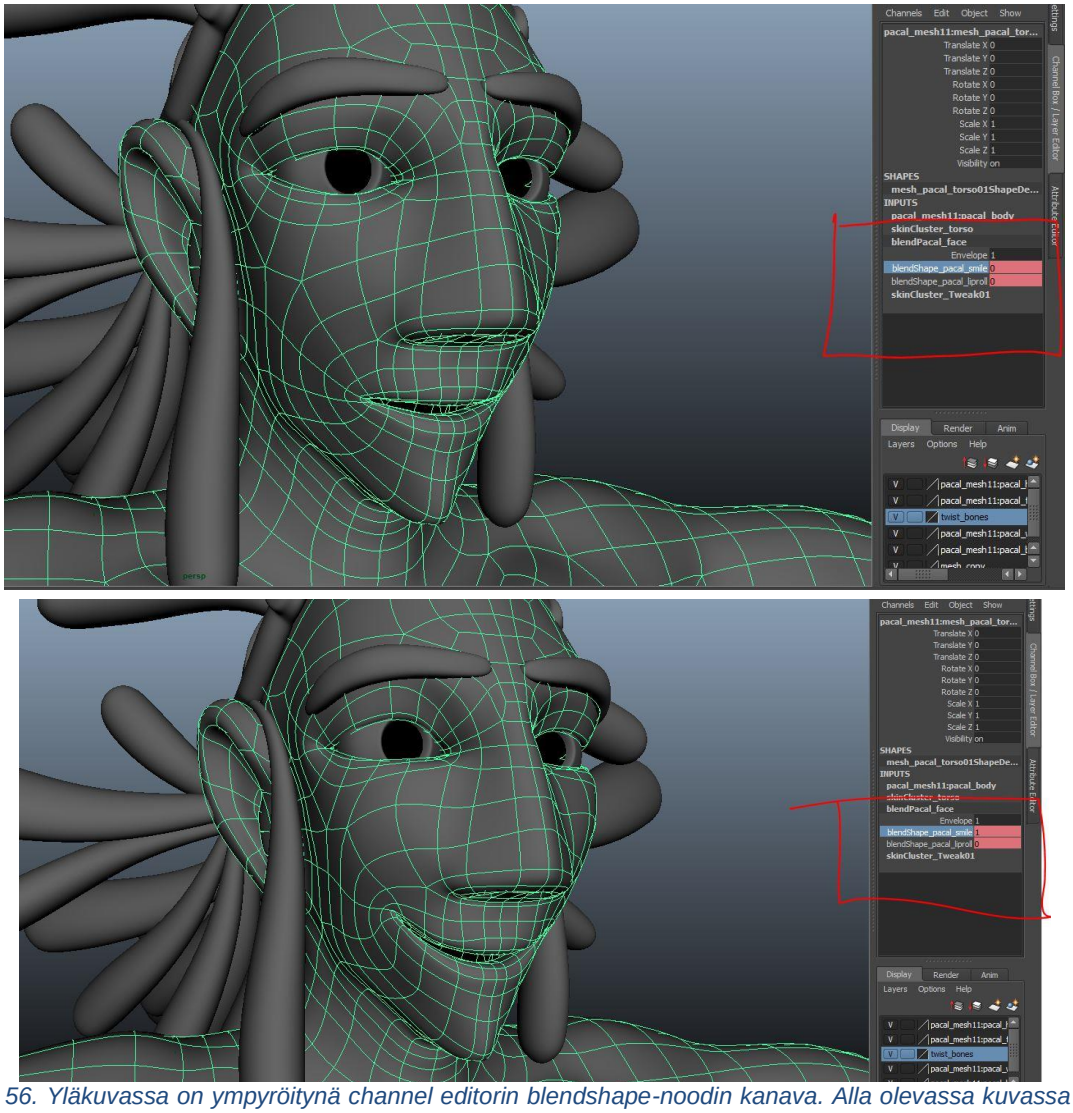


Kuva 55. Keskivartalon taivutus on nyt mahdollista. Ala- ja yläselän ohjaimia ei ole liitetty toisiinsa hierarkkisesti, jolloin jokainen ohjain voi liikkua vapaasti muista riippumatta.

6.2 Blend shape

Blend shape on yksi kasvorigeissä eniten käytettyjä työkaluja. Ideana on, että objektista on kaksi kopiota. Toinen objekti on varsinainen mesh, meshin kopio taas on muokattu versio alkuperäisestä objektista. Kopioita voi olla useampia.

Muokatusta objektista tehdään toisen objektin blendshape. Alkuperäinen objekti saa osakseen blend shape noodin, joka sisältää jokaisen lisätyn blend shapen kanavan. Kanavan arvo voi vaihdella arvosta 0 arvoon 1. Kun arvo on 0, blendshape on pois päältä, kun arvo on päällä, shape on päällä. Tämän lisäksi arvoilla voi olla desimaalisia väliarvoja. Hahmon kasvot voidaan siis asetella muokkaantumaan blendshapen muotoon reaalisessa ajassa (kuva 56.).



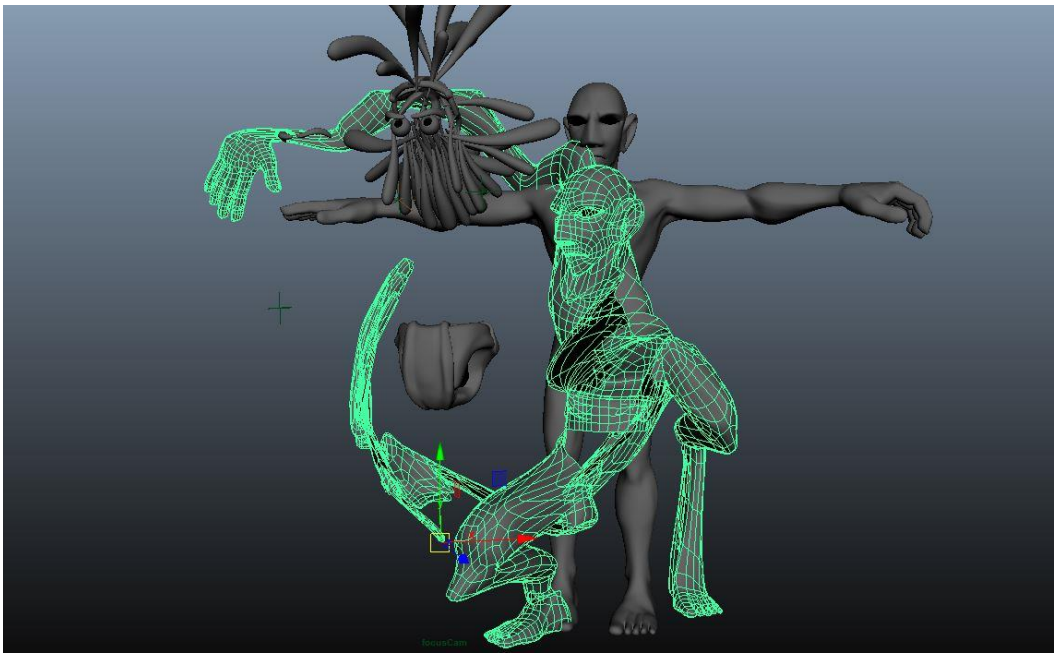
Kuva 56. Yläkuvassa on ympyröitynä channel editorin blendshape-noodin kanava. Alla olevassa kuvassa blend shape on päällä.

Kuva 57. Kun blendshape-noodi rakennetaan, käyttäjä voi päättää mitataanko blendin muutoksien alkupisteet lokaalien arvojen mukaan vaiko globaalien. Tämä määrittää sen pyrkiikö alkuperäisen meshin vertexit mukautumaan blendshapen vertexien sijaintiin globaalilla tasolla vaiko lokaalilla. Lokaalilla tasolla alkuperäisen meshin vertexit pyrkivät omaksumaan blendshapen vertexien arvot sen mukaan, miten ne on mitattu blendshapen lokaalista navasta lähtien. Globaalissa systeemissä erot mitataan globaalin navan mukaan. Alkuperäisen meshin vertexit pyrkivät siinä tapauksessa siirtymään täsmälleen blendshapen vastaavien vertexien kohdalle.

Lokaalin etuna on, että blendshape-objektit voidaan siirtää työtilassa mihin suuntaan tahansa ja vain shape noodiin tehdyt muutokset heijastuvat alkuperäiseen meshiin. Toinen etu on, että alkueräisen meshin sijaintia voi muuttaa vapaasti ilman, että se vaikuttaa siihen, miten blendshapet muokkaavat meshiä.

Yllä olevassa kuvasssa on hyödynnetty lokaalia alkupistettä. Oikealla olevan blendshapen napa sijaitsee sivussa alkuperäiseen meshiin nähden. Alkuperäinen mesh pyrkii omaksumaan blendshapen vertexien sijainnit sen mukaan, missä ne sijaitsevat blendshapen oman objektin navan mukaan.

Alla on sama tilanne globaalin navan tapauksessa. On huomioitava, että alkuperäistä meshiä on liikuteltu nivelien avulla, jolloin monien alkuperäisen meshin vertexien sijainnit ovat globaalissa mittakaavassa hyvin sekalaisissa suunnissa, johtaen hyvin… arvaamattomaan tulokseen. Mahdollisuus liikuttaa hahmoa nivelien sekä antaa sen meshin muokkaantua blendien mukaan tekee siitä hyödyllisemmän vaihtoehdon useammissa tapauksissa.



Useimmiten objektilla tarvitsee olla vain yksi blendshape noodi, joka taas sisältää kaikki blendshape-kanavat. Kun kursoria liikuttaa, alkuperäinen noodi saa kyseessä olevan shapen muodon. Kun haluttuun blendshape noodiin tarvitsee lisätä muita muotoja, tulee ensin valita haluttu blendshape ja sitten pohjamesh. Seuraavaksi ”edit deformers” –valikosta valitaan ”blend shape” ja edelleen ”add” –työkalu. Työkalun asetuksista valitaan haluttu blendshape noodi, johon uusi shape lisätään. Shape saa oman kanavan channel editoriin blendshape noodin alaisuuteen.

Pohjimmiltaan kyse on siitä, että alkuperäisen objektin control vertexien arvot muuntuvat kopion arvoiksi. Ehtona on, että vertexien määrä kopioissa on sama. Jos siis alkuperäiset kasvot vaativat muokkausta, muokkaukset on tehtävä ennen blendshapen ottamista.

Toinen huomioitava seikka, eritoten kun kopioita on useita, on huomioida miten blendshapen mekanismi toimii matemaattisesti. Blendshape-muotoja on mahdollista yhdistellä keskenään uusien ilmeiden muodostamiseksi, mutta lopputulokset voivat olla hyvin arvaamattomia, ellei käyttäjä tiedä, mitä logiikalla blendshapet muokkaavat objektien kontrolli-vertexejä.

Jason Osipa kuvaa kirjassaan ”Stop Staring” (2010) blend shapen mekanismin seuraavasti: Blend shapet toimivat ennen kaikkea lisäävästi (adding). Jokaisella control vertexillä on lokaationsa xyz-avaruudessa, blend shape huomioi erot näissä sijainneissa ja lisää nämä eroavaisuudet vastaavien vertexien sijaintiin. Vaikka sijainnin muutos olisi muutamilla akseleilla negatiivinen, kyseessä on yhä aritmeettinen lisäys ”+(-x)”. Yhden blendshapen kohdilla tämä ei ole ongelma. Useamman shapen kanssa tilanne on toinen. Useamman shapen muokatessa ne ennen kaikkea kasaantuvat päällekkäin eivätkä kohtaa keskiarvossa, eli morphaudu (morph) (Jason Osipa, 2010, 101-102.)
Asia on helpompaa kuvata esimerkillä. Blend shapella on kaksi kopiota. Kumpikin kopio vaikuttaa osaltaan vertex.1:n sijaintiin. Kopio A muuttaa vertexin sijaintia yhdellä akselilla kaksi yksikköä positiiviseen suuntaan. Kopio B muuttaa vertexin sijaintia samalla akselilla kolme yksikköä negatiiviseen suuntaan. Jos molemmat blend shapet pistetään päälle, vertexin sijainnin muutos on (-3) + 2 = -1. Tuloksena oleva muoto ei siis ole kahden muodon keskiarvo, vaan kahden eroavaisuuden yhteenlasku (kuva 58.).

Kuva 58. Vasemmalla ja keskellä olevat kuvat esittävät yksittäisiä blendshapeja. Oikealla kuva esittää meshiä, jossa molemmat shapet ovat päällä. Kun kaksi eri shapea muokkaa samaa meshiä, tuloksena ei ole kahden muodon keskiarvoa, vaan molempien meshien eroavaisuudet yhdistyvät muodostaen lopputuloksen.

Kasvoanimaatioissa tätä kaksoismuutosta pyritään välttämään esimerkiksi kohdistamalla yhden shapen muutokset vain yhteen alueeseen. Pienemmissä projekteissa, joissa erilaisia blendejä on vähemmän, tämä ei ole niin suuri ongelma, mutta suurteollisuuden tuotannoissa blendejä saattaa olla yhdellä hahmolla jopa satoja. On huomioitava myös sekin, jos alkuperäisiin kasvoihin tehdään muokkauksia shape-tasolla, blendshapet tekevät laskelmansa niiden vertex-arvojen mukaan, jotka alkuperäisellä meshillä oli ennen muutosta.

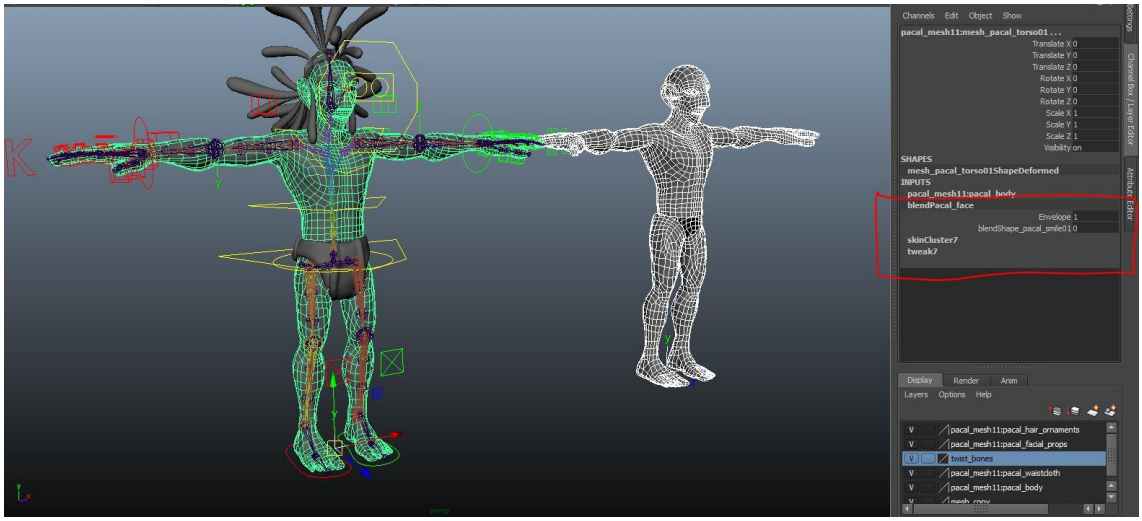
6.2.1 Esimerkkityö, Pacalin ilmeet

Esitän tässä esimerkin Pacalin ilmeiden rakennuksesta ja erinäisen ajurin rakentamisesta ajamaan blendshape ominaisuuksien arvoja. Aluksi on pidettävä huolta siitä, että hahmon ohjaimet ovat täysin nollattuja. Kopion ottaminen muokatusta hahmomallista saattaa aiheuttaa ylimääräisiä muutoksia alkuperäisessä meshissä.

Seuraavana otetaan kopio Pakalin meshistä, se siirretään sivummalle. Kannattaa nimetä kopio sopivalla nimellä navigoinnin helpottamiseksi. Suositeltavaa on lisätä nimeen myös ilme, jota kopion on tarkoitus esittää. Tässä tapauksessa ensin tehdään hymyshape. Kopion kasvot mallinnetaan hymyilemään. Translate-työkalun ”tool settings”-valikossa voi säätää ”soft selection”-ominaisuuden päälle, joka tekee mallinnuksesta sulavampaa.

Seuraavaksi valmistetaan blendshape-noodi. Ensin valitaan kopio, sitten varsinainen shape. ”Create Deformer”-valikosta valitaan ”blend shape”. Työkaluvalikossa nimeä

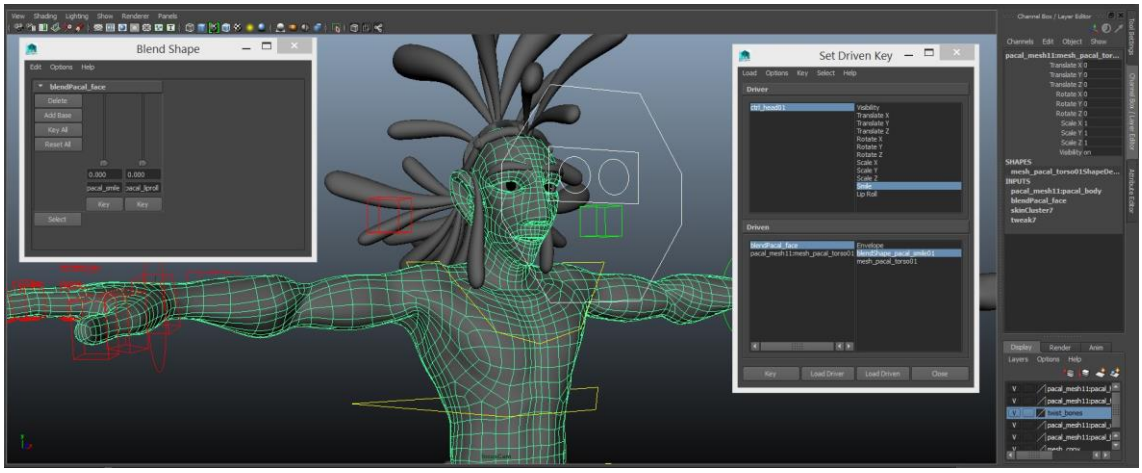
blendshape-noodi ja valitse alkuperäpisteeksi (origin) ”local”. Tällä tavoin alkuperäinen mesh laskee blend shapen vertexien paikan mitattuna itse shapen lokaalista nollakohdasta eivätkä alkuperäisen meshin vertexit pyri siirtymään globaalissa tilassa shapen vertexien kohdalle. Blendshape-noodin ominaisuudet näkyvät alkuperäisen meshin channel editorissa. Ominaisuuksia voi säätää klikkaamalla blendshape-noodin nimeä channe editorissa. Blendshapen ominaisuus-valikossa pitäisi näkyä smile-kopion nimike. Muuta sen viereisen kanavan arvo ykköseksi, jolloin alkuperäisen meshin pitäisi nyt muokkaantua smile-kopion mukaiseksi.



Kuva 59. Oikealla on Pakalin meshin kopio. Ensin on valittu kopio, ja sitten alkuperäinen, jonka jälkeen ”blend shape”-noodi on valmistettu (ympyröity).

Seuraavaksi valmistetaan smile-shapen arvon ajuri-ominaisuus. Ajuri ominaisuus voidaan asetella kasvo-ohjaimeen extra-ominaisuudeksi. Ominaisuudesta tehdään ”float”-luku ja minimiarvo on nolla ja maksimi kymmenen ja ominaisuuden nimeksi laitetaan ”smile”. Seuraavana avataan ”set driven key”-työkalu ja kasvo-ohjain valitaan ajuriksi ja ”smile” ajuriominaisuudeksi. Blendshapen asettaminen ajettavaksi objektiksi vaatii sen, että se valitaan alkuperäisen meshin channel editorista. Ei siis riitä, että alkuperäinen mesh on valittuna. ”Smile”-blendshape asetetaan ajettavaksi ominaisuudeksi.

Loppuosa on sama prosessi kuin edellisissäkin ”driven key” tilanteissa. Ajuriominaisuuden nollakohdassa ”smile”-ominaisuus on nollattu ja ajuriominaisuuden maksimiarvossa ”smile” on täysissä arvoissa. Kun teet lisää ilmeitä muilla kopioilla, ei kannata enää tehdä kopiosta blend shapea ”create deformers”-valikon kautta, sillä se loisi vain uuden shapen. Lisää uudet shapet aina valmiiseen blendshape-noodiin ”edit deformers”-valikon kautta. Ensin valitaan kopio, sitten alkuperäinen ja ”edit deformers” > ”blend” > “add” ketjun kautta työkalu-asetuksista ”blend shape”-noodi, johon uusi shape halutaan lisättävän.



Kuva 60. Kasvo-ohjaimen ”smile” ominaisuus on ”set-driven-key” ikkunassa asetettu ”blend shape smilen” ajuriksi. Vasemmalla näkyvä ikkuna on blendshape-noodin sisältämien blendien hallitsemiseen. Ikkuna löytyy valintaketjusta: window > animation editors > blend shape.

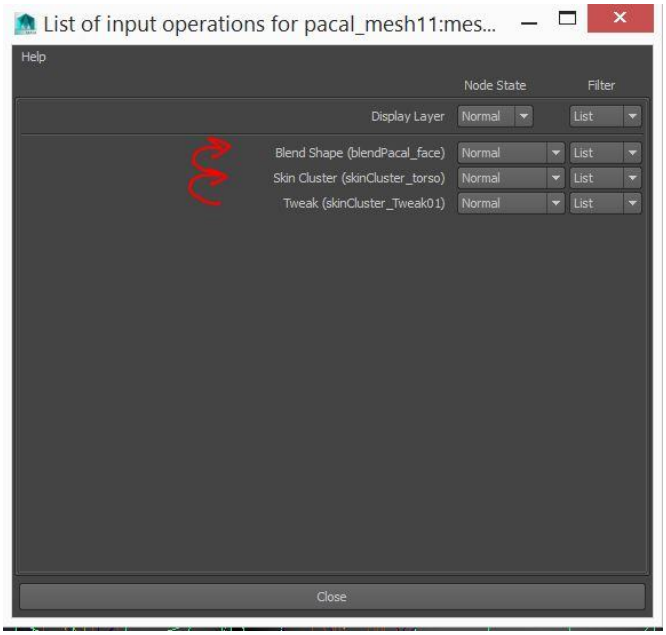
Tämä kasvorigi ei ole kaikkein sulavin, sillä suurissa animaatioprojekteissa, joissa blend shapejen määrä kulkee sadoissa, rigi-järjestelmä on paljon sulavampi.

6.3 Deformaatiojärjestys

Deformereita valmistellessa on huomioitava myös niiden rakennusjärjestys. Kun deformeri muuttaa meshiä, se pyrkii muokkaamaan ennen kaikkea meshiä, joka sillä on sillä hetkellä käytössään. Jos jokin deformeri on lisätty sitä ennen osaksi meshiä, uusin deformeri muokkaa jo muokattua meshiä. Joissain tapauksessa useampi deformeri johtaa siihen, että meshi ei deformoidu toivottavalla tavalla, koska deformerit ovat vääränlaisessa järjestyksessä.

Tämä ongelma tuli vastaan Pacalin rigissä silloin kun tein pari blendshape-ilmettä hänelle. Ongelmana oli, että Pacalin blendshapen ollessa päällä meshi ei deformoidu nivelien mukana. Tämä johtuu siitä, että skincluster deformeri toimii ennen blendshapea. Skincluster säätää meshin muodon vastaamaan nivelien liikettä, mutta blendshape, jonka meshin muoto vastaa normiasentoa, muokkaa meshin vertexit takaisin tahan positioon.

Tämän asian voi simppelisti korjata vaihtamalla järjestystä, jossa deformerit vaikuttavat meshiin. Jos blendshape ensimmäisenä vaikuttaa meshin muotoon, niin skincluster taas vaikuttaa tämän blendshapen määrittämään muotoon. Niin kauan kun blendshape-mesh omaa saman määrän vertexejä kuin alkuperäinen mesh, niin alkuperäiseen meshiin asetellut skinclusterin painoarvot ovat pitkälti samat blendshapessa.



Kuva 61. Input-liitännäisten järjestys, jossa niiden tekemät muutokset suoritetaan. Järjestystä voi vaihtaa valitesemalla liitännäisen ja hiiren keskipainike pohjassa siirtää liitäntää listassa ylemmäs tai alemmas.

Deformaatioiden järjestystä voi muuttaa valitsemalla deformoitava objekti. Sen jälkeen klikataan hiiren oikeaa näppäintä. Aukenevasta valikosta valitaan ”input” – sarake ja siitä edelleen ”all inputs”. Avautuvassa listassa näkyvät kaikki objektin deformerit. Klikkaamalla yhtä deformereista hiiren keskimmäisellä näppäimelle ja vetämällä sen listassa alemmas tai ylemmäs vaikuttaa siihen, missä järjestyksessä deformerit päivittyvät. Listan alimmat deformerit prosessoituvat ensin ja päivittyvät sitä mukaa kun listassa edetään ylöspäin (kuva 61.).

7 Päätesanat

7.1 Tutkimuksen taustat

Tämä tutkielma on käynyt läpi lukuisia muutosvaiheita. Muutosvaiheet ovat liittyneet niin tutkielman aiheeseen kuin siihen, mitä kaikkea tutkielmassa kerrotaan. Kolmannen vuoden opiskelujeni aikana olin Alankomaissa vaihdossa, missä minun tuli rakentaa lukukauden lopuksi portfolio työskentelystäni. Valitettavasti olin silloin jättänyt portfolioni kasaamisen liian viimetinkaan, johtaen astetta epäselvempään lopputulokseen. Silloinkin loppuraporttini aiheena oli riggaus.

Alankomaissa tekemäni tutkielma koski Autodesk Maxilla riggausta. Silloin lähestymistapani muistutti enemmän askel askeleelta tehtyä ohjeistusta, jossa kuvailin hahmon rigin valmistuksen askel askeleelta. Silloin aloin ymmärtää miten haastavaa riggauksesta on lopulta puhua kirjoitetussa asussa. Useimmat käyttämäni oppaat ja tekniikat olen löytänyt esimerkkien ja video-ohjeistusten avulla. Maxin etuna oli kuitenkin sen aloittelijaystävällisempi käyttöjärjestelmä.

Mayan tapauksessa asia ei enää ollut niin yksinkertainen. Mayan looginen, mutta hyvin pienien osasien muodostama kokonaisuus voi tehdä työskentelyprosessista hyvin vaikeaa.

Se, miten Alankomaiden kokemukseni poikkeaa tästä tutkielmasta, on se miten päädyin sen kanssa tähän kiireessä loppuun rakennettuun tulokseen. Toisin kuin Alankomaissa, olin jo syksyllä pohtimassa aihetta ja vielä keräilemässä jopa materiaalia. Ongelmana oli, että aiheeni oli silloin hyvin erilainen. Anatomian hyödyntäminen mallinnuksessa ja riggauksessa. Aihe oli lopulta turhan laaja-alainen ja kaiken lisäksi syynä valintaan oli enimmäkseen turhamaisuus. Kun lähdin pohtimaan tutkielmaani, kävin päässä monimutkaisia ajatusrakennelmia ja jopa hyvin kaukaa haettuja referenssejä. Kaikki tämä siksi, että saisin luotua jonkinlaisen illuusion kirjaoppineesta ja syvällisestä tutkielmasta.

Tässä piili lopulta sen virheen siemen, joka levittäytyi tähänkin tutkielmaan kun päätin vaihtaa aihetta ja palata riggauksen pariin. Minulla kävi jo aluksi mielessäni, että tekisin tarkemman analyysin Maya-riggauksen toiminnasta ja tekisin sen eritoten heille, jotka olisivat aloittelemassa Maya-riggausta.

Aloin rakentaa jonkinlaista hahmomallia ja rigiä, mutta ennen pitkää sain huomata, että motivaattorinani ei niinkään ollut toimivan tutkielman ja avustavan oppaan teko, vaan omilla kyvyilläni retostelu. Opinnäytteen tarkoitus onkin esitellä oppilaan taitoja, mutta motivaationi oli varsin heikkolaatuinen ottaen huomioon, että oppaan oli tarkoitus auttaa muita lukijoita ymmärtämään Mayaa paremmin. Tämä johti lopulta siihen, että olin kehitellyt kaikenlaisia outoja esimerkkejä, suureellisia suunnitelmia ja esittelyjä omasta osaamisesta ja Mayan toiminnoista. Käytin kyllä myös jonkin verran aikaa materiaalien kasaamiseen, mutta minulla ei lopulta ollut selkeää kuvaa siitä, mitä esittelen. Otin myös paniikin siitä, millaiseen materiaaliin minun olisi tullut tarttua. Koin, että minun olisi pitänyt valita hyvin syväluotaavia analyysejä ja käsitteistöjä sen sijaan, että olisin vain pohtinut sitä, oliko tarvittava informaatio hyödyllistä.
Lopputuloksena olikin sitten se, että materiaaleja kerättiin vain materiaalien vuoksi ja esimerkkitöitä tehtiin pitkälti itse esimerkkitöiden vuoksi. Ne eivät lopulta palvellut viimeisintä pistettä, tutkinnon valmistumista.

7.2 Päätesanat tutkittavasta asiasta

Lopputulos ei ole kaikkein kaunein, selkein, saatikka edes avuliain opas Mayalla riggaukseen. Hyvin paljon olennaisia asioita jäi sanomatta ja pelkäänpä, että en ole antanut kattavinta opastusta Mayan työkalujen toimintamekanismeista. Hyvin monta asiaa oli itsellenikin uusia vaikka olen ollut niiden kanssa tekemisissä lähes kaiken aikaa Mayalla työskennellessäni. Esimerkiksi lokaalien ja globaalien napa-pisteiden käsitteitä en ole tullut edes ajatelleeksi. Maya on laaja ja monipuolinen työkalu, mutta se tosiaan on työalusta, joka vaatii käyttäjältään, jos käyttäjä on siihen valmis, astetta syvempää perehdytystä ja kenties analyyttisempaa katsantoa.

Tämä analyyttinen katsanto ei kuitenkaan ole välttämättä haitaksi tuleville riggaajille. 3D-animaatio on lopulta hyvin teknologiakeskeinen ala. Matemaattisempien ja teknisten asioiden pohtiminen on aikalailla väistämätöntä alalla työskennellessä. Aivan kuten 2D-animaattorien täytyy hallita työkalunsa, myös 3D-animaattorien ja artistien on tehtävä samoin. Ymmärtämällä mekanismit Mayarigien takana, riggaajalle on suuremmat edellytyksensä antaa oma panoksensa työhön. Tämä on universaalimpi teesi, eikä se rajoitu vain Mayaan. Mikä tahansa 3D-ohjelmisto on osaavissa käsissä vahva työkalu.
Se, miksi itse aikoinaan sen haasteista ja vioista huolimatta kiinnostuin Mayasta, oli sen jatkuva kasvuvara ja vaatimus tarkempaan pohdintaan. Uskallan väittää, että Maya on hyvä alusta tällaisen analyyttisen havainnoinnin kehittämiseen.

Hyvin paljon asiaa olisi voinut vielä kertoa Mayasta ja sen toiminnasta, mutta ajan käydessä vähiin, oli käsiteltävät asiat priorisoitava. Itse uskaltaisin nostaa hierarkioiden ja riippuvaisliitosten toimintaperiaatteet ja lokaalien ja globaalien napojen toiminnan yhdeksi tärkeimmistä asioista Mayan toiminnassa. Mayalla työskentely on opettanut minulle sen, että kaikki asiat, joita kuulemme ja asiat, joita opimme, tulee suodattaa ja pureskella tarkoin. Meidän tulee ennen kaikkea pyrkiä analysoimaan oppimamme asiat ja uskaltaa pohtia miksi jotkin asiat toimivat kuin toimivat. Siten voimme saada parhaat palat irti siitä mitä teemme.

7.3 Mitä tulevaisuudessa?

Tässä oppaassa käsitellyt aiheet ja havainnot ovat lopulta vain pintaraapaisu sille, mitä Mayalla on tarjottavanaan. Olen yrittänyt parhaani mukaan esitellä perus-toimintamekanismeja Mayan rakenteen takana. Näiden perusominaisuuksien hyödyntäminen puolestaan sallii sisäänpääsyn sellaisille tekniikoille kuin venyvät nivelet, tekstuurikarttojen animointi tai jopa rendaustulosten hallinta.

Tässä oppaassa on jätetty pitkälti huomiotta MEL-skriptin käyttö ja sen reaaliaikaisen variaation, expression editorin, käyttö. Kenties nämä asiat tulevat ennen pitkää olemaan uuden tutkielman aihe hamassa tulevaisuudessa.

Kiitokset mielenkiinnostasi tätä tutkielmaa kohtaan.

Lähteet

Harovas Perry, Kundert-Gibbs John, Lee Peter, 2000, Mastering maya complete 2, SYBEX inc., Alameda, California.

Osipa Jason, 2010, Stop Staring, Wiley Publishing inc., Indianapolis, Indiana, Canada.

Digitaltutors, Delano Athias, 2013 (?), Gain the knowledge to enchance your rigs. http://www.digitaltutors.com/tutorial/1472-Pushing-Your-Character-Rigs-Beyond-the-Basics-in-Maya

Rigging Dojo, 2014, Everything you thought you knew about maya joint orient is wrong! http://www.riggingdojo.com/2014/10/03/everything-thought-knew-maya-joint-orient-wrong/

Wikipedia 2015a, Node graph architecture http://en.wikipedia.org/wiki/Node_graph_architecture

Autodesk, 2015a, Nodes and attributes, Autodesk knowledge network http://knowledge.autodesk.com/support/maya/getting-started/caas/CloudHelp/cloudhelp/2015/ENU/MayaLT/files/Nodes-and-attributes-Nodes-and-attributes-overview-htm.html

Autodesk, 2015b, Node types, Autodesk knowledge network

http://knowledge.autodesk.com/support/maya/getting-started/caas/CloudHelp/cloudhelp/2015/ENU/MayaLT/files/Nodes-and-attributes-Node-types-htm.html

Autodesk 2015c, DAG-objects, Autodesk knowledge network http://knowledge.autodesk.com/support/maya/getting-started/caas/CloudHelp/cloudhelp/2015/ENU/MayaLT/files/GUID-5029CF89-D420-4236-A7CF-884610828B70-htm.html

Autodesk 2015d, Smooth Skin Weight Normalization, Autodesk Maya 2015 Help
http://help.autodesk.com/view/MAYAUL/2015/ENU/?guid=Skinning_Controlling_smooth_skin_weight_normalization

Autodesk 2015e, Rotate Plane IK solver, Autodesk knowledge network
http://knowledge.autodesk.com/support/maya/learn-explore/caas/CloudHelp/cloudhelp/2016/ENU/Maya/files/GUID-9942FFB5-65C2-46E2-B5A3-297667A9FB5D-htm.html

Autodesk 2013a, World space, object space and local space, Autodesk knowledge network
http://knowledge.autodesk.com/support/maya/learn-explore/caas/mnehelp/global/docs/maya2014/en_us/files/Transforming-objects-World-space-object-space-and-local-space-htm.html

Autodesk 2013b, Transformations, Autodesk Knowledge network
http://knowledge.autodesk.com/support/maya/learn-explore/caas/mnehelp/global/docs/maya2014/en_us/files/Transforming-objects-Transformations-htm.html

Autodesk 2014a, IK handles, Autodesk Knowledge Network
http://knowledge.autodesk.com/support/maya/learn-explore/caas/CloudHelp/cloudhelp/2015/ENU/Maya/files/CSS-IK-handles-htm.html

Autodesk 2014b, IK solvers, Autodesk Knowledge Network
http://knowledge.autodesk.com/support/maya/learn-explore/caas/CloudHelp/cloudhelp/2015/ENU/Maya/files/CSS-IK-solvers-htm.html

Autodesk 2014c, Double transformation effects
http://download.autodesk.com/global/docs/maya2014/en_us/index.html?url=files/Skinning_Double_transformation_effects.htm,topicNumber=d30e306047

Kaikki yllä olevat linkit ovat luettu 30.4.2015.

Kaikki kuvat ovat peräisin omasta projektista tai itse rakennetuista esimerkeistä.