Harri Aaltonen

Rigaustyökalut Autodesk Maya-ohjelmassa ja niiden sovelluksia pelimoottoreille

Metropolia Ammattikorkeakoulu

Medianomi

Viestinnän koulutusohjelma

Opinnäytetyö

4.4.2016

Metropolia

Tekijä(t) Otsikko Sivumäärä Aika	Harri Aaltonen Rigaustyokalut Autodesk Maya-ohjelmassa ja niiden sovelluksia pelimoottoreille 58 sivua + 45 liitettä 24.4.2016
Tutkinto	Medianomi (AMK)
Koulutusohjelma	Viestinnän koulutusohjelma
Suuntautumisvaihtoehto	3D-animointi ja visualisointi
Ohjaaja(t)	Peke Huuhtanen Jaro Lehtonen

Tässä opinnäytetyössä käsittelen rigaustyökaluja ja niiden sovelluksia Autodesk Maya 2016-ohjelmassa, jotka ovat yhteensopivia nykyaikaisten pelimoottorien kanssa. Aluksi käyn läpi erittäin kattavasti mitä rigaukseen liittyviä työkaluja Maya sisältää. Sen jälkeen esittelen muutamia hyvin yleisesti käytettyjä rigisovelluksia, kuten soft-IK, ribbon-spine, IK/FK-blend ja twist-jointit. Osa näistä sovelluksista sisältää itse kehittelemiäni toiminnallisuuksia. Lopuksi käsittelen mitä tarkoittaa shadow-rig, jonka avulla rigi tehdään pelimoottoreihin yhteensopivaksi.

Samat työkalut löytyvät lähes kaikista hyvin varustelluista 3D-ohjelmista, mutta usein eri nimillä ja saattavat toimia hieman eri tavoilla. Myös samat rigisovellukset on mahdollista rakentaa lähes kaikissa 3D-ohjelmissa. En kuitenkaan syvenny vertailemaan eri 3D-ohjelmistoja, vaan keskityn täysin rigaamiseen Mayalla.

Työskentelen teknisenä animaattorina pelialalla, jossa joudun animoimaan itse rakentamillani rigeillä. Tämä on auttanut minua näkemään rigauksen animaattorin näkökulmasta. Kykenen näkemään, minkälaiset järjestelmät on nopeita ja helposti käytettäviä animaattorille ilman, että minun tarvitsee odottaa arviota erikseen animaattoreilta. Pystyn myös ymmärtämään toiveita rigien käytettävyyteen liittyen animaattoreiden näkökulmasta.

Tämän opinnäytetyön tavoitteena on selventää aloitteleville rigaajille asioita, jotka voivat olla hankalasti ymmärrettäviä, kuten gimbal-lukko ja local rotation axes. Toisena tavoitteena on auttaa ja kannustaa aloittelevia rigaajia kehittelemään omia sovelluksia Mayan rigaustyökaluilla. Tästä opinnäytetyöstä on eniten hyötyä lukijoille, joilla kuitenkin on jo jonkinlainen perustuntemus 3D-animaatiosta ja rigaamisesta.

Avainsanat	riggaus, Maya, 3D

Author(s) Title Number of Pages Date	Harri Aaltonen Rigging Tools in Autodesk Maya Program and Their Applications for Game Engines 58 pages + 45 appendices 24 April 2016
Degree	Bachelor of Arts and Culture
Degree Programme	Media
Specialisation option	3D Animations and Visualisation
Instructor(s)	Peke Huuhtanen Jaro Lehtonen

In this thesis, I focus on rigging tools and their applications in Autodesk Maya 2016 software, which are compatible with modern game engines. At first, I comprehensively walk through rigging related tools included in Maya. After that, I discuss a few very commonly used rigging applications such as soft-IK, ribbon-spine, IK/FK-blend and twist-joints. Some of these applications includes self-developed functionalities. Lastly, I explain what does a shadows-rig mean, which is used for making the rigs compatible with game engines.

The same tools can be found in almost all well-equipped 3D-programs but mostly with different names and they may work in slightly different ways. The same rigging application are possible to build in almost all other 3D-applications as well. However, I will not delve into comparing different 3D-applications but I will focus entirely on rigging with Maya.

I work as a technical animator in the game industry, where I have a chance to animate with my self-built rigs. This has helped me to see rigging from the animator's point of view. I am able to see what kinds of systems are fast and easy for the animators to use without having to wait for the review from the animators separately. I can also understand requests related to usability of rigs from the animator's point of view.

The purpose of this thesis is to clarify subjects that may be difficult for beginner riggers such as gimbal lock and local rotation axes. The second purpose of the thesis is to help and encourage beginner riggers to develop their own applications with Maya's rigging tools. However, this thesis is most useful for readers who already has some kind of basic knowledge about 3D-animation and rigging.

Keywords	rigging, Maya, 3D

Sisällys

1 Johdanto

Tämän opinnäytetyön tarkoitus on syventyä rigaustyökaluihin ja sovelluksiin Maya 2016-ohjelmistolla, jotka ovat yhteensopivia nykyaikaiseten pelomoottorien kanssa, kuten Unity 5 ja Unreal Engine 4. Työskentelen itse teknisenä animaattorina pelialalla, jossa joudun animoimaan itse rigaamani hahmot. Tässä asemassa minulla on ollut mahdollisuus kehittää rigaustaitoani myös animaattorin näkökulmasta.

Keskityn täysin pelimoottoreissa toimiviin rigeihin, jotka käyttävät ainoastaan klassista smooth skinnausmetodia ja ylimääräisenä korjauksena blend shapeja. Käsittelen aihetta Maya 2016-ohjelmiston kautta, mutta lähes kaikki mainitsemani työkalut löytyvät muistakin ohjelmista. Eroja muihin ohjelmiin voi olla lähinnä työkalujen nimissä, niiden sisältämissä ominaisuuksissa ja matemaattisessa toteutuksessa, mikä saattaa joko monimutkaistaa tai helpottaa riggausprosessia. Keskityn kuitenkin rigaamiseen vain Maya ohjelmalla, enkä vertailuun muihin 3D-ohjelmiin.

En käy seikkaperäisesti läpi, miten kokonainen hahmorigi rakennetaan, vaan käsittelen pääasiassa erilaisia rigauksen työkaluja, joiden avulla pystytään luomaan lähes mikä tahansa nykyaikainen pelimoottoreissa toimiva rigi. Ensiksi käsittelen erittäin kattavasti Mayan sisältämiä rigaustyökaluja ja muutamia Maya ohjelmaan liittyviä mekanismeja, joiden ymmärrys on erittäin tärkeää rigaajille (Luku 3). Sen jälkeen siirryn käsittelemään yleisesti rigeissä käytettyjä sovelluksia ja miten niitä pystytään rakentamaan Mayan sisäänrakennetuilla työkaluilla (Luku 4).

Rigaus on aiheena hyvin laaja, joten en pysty käsittelemään aivan kaikkea 3D:n perusteista lähtien. Tästä johtuen, tämä opinnäytetyö sopii parhaiten aloitteleville rigaajille, joilla on jo jonkinlainen perustuntemus 3D:n peruskäsitteistä, 3D-animaatiosta ja rigaamisesta.

2 Käsitteiden määrittely

Node

Nodella tarkoitan dependency graph-nodea (DG-node), jolla on sisääntuloja (input) ja ulostuloja (output). Node ottaa sisään arvoja, joita se käyttää matemaattisissa operaatioissa. Matemaattisen operaation tuottama arvo palautetaan outputin kautta. Noden output-arvo voidaan taas kytkeä seuraavaan nodeen, jolla on omat matemaattiset operaatiot. Erilaisia nodeja voi olla esimerkiksi distanceBetween, joka ottaa sisään kaksi positiovektoria ja laskee niistä etäisyyden. (Autodesk 2016a.)

Objekti

Tässä opinnäytetyössä objektilla tarkoitan mitä tahansa Mayan skeneen ilmestyvää objektia eli DAG (Directed Acyclic Graph) -nodea, jolla on transform-arvoja eli positio (translate), rotaatio ja skaala. Objekti voi olla skenessä näkyvä tai näkymätön, kunhan sillä on paikka skenen avaruudessa.

Parent/child-hierarkia

3D-ohjelmissa käytetään niin sanottua parent/child-hierarkiaa. Child-objektiksi kutsutaan sellaista objektia, joka perii joltain toiselta objektilta sen transform-attribuutit, ja se objekti jolta transform-attribuutit peritään, on parent-objekti. Mayassa hierarkiaa voi muuttaa parent- ja unparent-komennoilla, joita kutsun parentoinniksi ja unparentoinniksi.

Kuvio 1. Esimerkki parent- ja child-objektista.

Attribuutti

Kaikki objekteilta luettava tai muutettava informaatio on niiden attribuuteissa. esimerkiksi transform-objektin translateX, translateY ja translateZ. Joitain attribuutteja pääsee manipuloimaan vain ohjelmoimalla.

Solveri

Solveri on tietokoneohjelma, jolla ratkaistaan jokin matemaattinen ongelma. Esimerkiksi fysiikkamoottori on eräänlainen solveri, joka ratkaisee, miten objektit liikkuvat tietyissä olosuhteissa.

Twist

Twistillä tarkoitan tässä opinnäytetyössä tähtäyskohteeseen osoittavan vektorin ympäri pyörivää rotaatioakselia.

Nurbs

Nurbs tulee sanoista non-uniform rational basis spline. Maya käyttää näitä geometrioiden ja käyrien laskentaan (Autodesk 2016b).

Pinta

Planella tarkoitan tässä opinnäytetyössä matemaattisesti tasoa avaruudessa. Se voi olla polygoneista tai nurbseista muodostuvaa geometriaa. Kutsun näitä tässä työssä pinnoiksi.

Bindaus

Bindauksella tarkoitetaan operaatiota, jolla geometria kytketään deformereihin eli muokkaajiin, tässä tapauksessa jointteihin, jotka ovat rigaamiseen erikoistuneita muokkaajaobjekteja.

Maailman avaruus ja paikallinen avaruus

World-space tarkoittaa transform-attribuuttien arvoja suhteessa skenen avaruuteen ja local-space on transform-attribuuttien-arvoja suhteessa parent-objektiin. Käytän World-space-sanan tilalla maailman avaruutta ja local-space-sanan tilalla paikallista avaruutta. Transform-arvot saadaan maailman avaruudessa transform-objektin worldMatrix-attribuutin kautta decomposeMatrix-nodella. (Autodesk 2016c.)

Muokkaaja

Deformer eli muokkaaja on node, jonka avulla muokataan objekteja komponenttitasolla. Pelimoottoreissa muokkaajina yleisesti voidaan käyttää vain smooth skin-algoritmia ja blend shapeja.

Kytkös

Connectioni eli kytkösellä tarkoitetaan sitä, kun noden output-attribuutti syötetään toisen noden input-attribuuttiin. Attribuutteja myös kytketään erilaisten apunodejen avulla, kuten constrainteilla.

Leivonta

Baking eli leivonta tarkoittaa sitä, kun proseduraalista tai dynaamista dataa tallennetaan pysyvään muotoon (ARLAb). Esimerkiksi fysiikkasimulaatio voidaan leipoa animaatiodataksi, jolloin tietokoneen ei tarvitse laskea raskasta simulaatiota uudestaan. Sama prosessi joudutaan tekemään, kun animaatiot siirretään pelimoottoriin. Kompleksit animaatiokäyrät muutetaan yksinkertaisempaan muotoon, jota pelimoottori pystyy varmasti lukemaan. Pelimoottorit voivat tämän jälkeen tehdä omia prosessointejaan animaatiodatalle, kuten pakkauksen.

3 Maya-ohjelman sisäänrakennetut rigaustyökalut

Tässä luvussa käyn läpi Maya 2016-ohjelmasta löytyviä sisäänrakennettuja työkaluja. Rigaus on yksi 3D-alan teknisimmistä osa-aluista. Ilman kattavia perustietoja rigausprojektit usein jumiutuvat selittämättömiin ilmiöihin, kuten dependency-looppiin, jointtien rotaatiot hyppäävät 180 astetta tai animaattori huomaa kontrolliobjektien olevan gimbal-lukossa liian usein.

3.1 Transform-, shape- ja joint-nodet

Mayassa kaikilla skenessä nähtävillä objekteilla on parent-objektina transform-objekti. Transform-objekti sisältää objektin skaalan, rotaation, translaation ja muita ominaisuuksia (Autodesk 2016c). Transform-objektin child-objektina voi olla shape-node, joka määrittää objektin tyypin ja sen erikoisominaisuudet. Esimerkiksi polygoniobjektin shape-node sisältää kaiken polygonin komponentteihin liittyvän informaation, kuten verteksit.

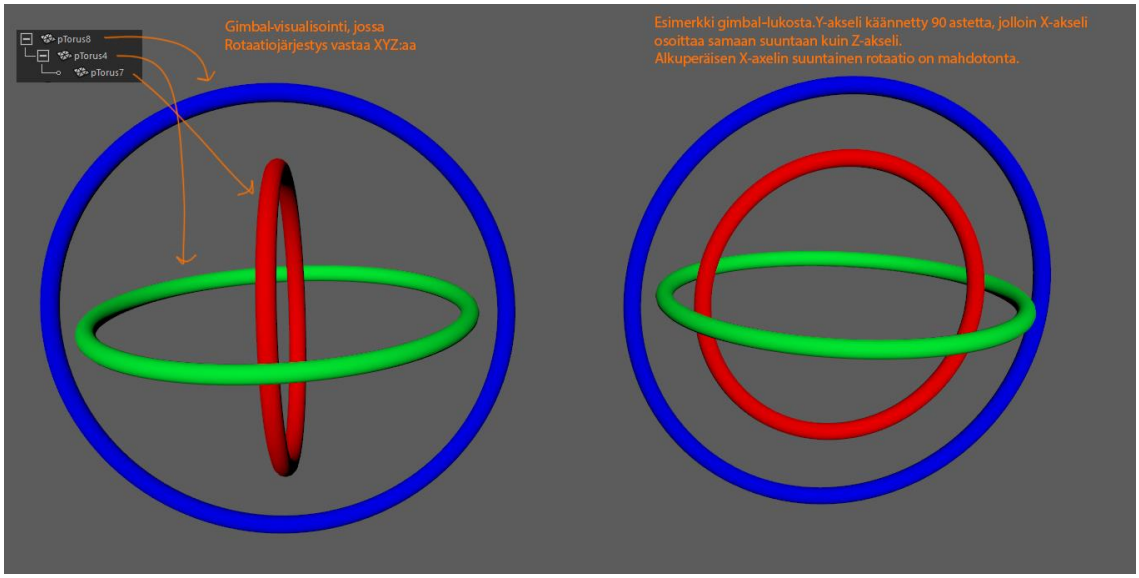
Kuvio 2. Shape ja transform-objekti outlinerissa

Joint-objekti Mayassa on rigaukseen erikoistunut transform-objekti, joka sisältää rigaukseen erikoistuneita ominaisuuksia, kuten joint orientation ja segment scale compensation. Näiden erikoisominaisuuksien takia Mayan jointit ovat hyödyllisiä erittäin monissa eri rigaukseen liittyvissä tilanteissa. (Autodesk 2016d.)

3.1.1 Gimbal, rotation order ja euler-kulmat

Mayassa ja lähes kaikissa vastaavissa 3D-ohjelmissa rotaatioita manipuloidaan euler-kulmilla. Euler-kulmilla on 3 akselia, X, Y ja Z, jotka yhdessä määräävät lopullisen rotaation kolmiulotteisessa avaruudessa. Näillä akseleilla on järjestys, jota kutsutaan rotation orderiksi. Rotation order määrittää, missä hierarkiassa kukin akseli on toisiinsa nähden. Esimerkiksi jos rotation order on (X, Y, Z), viimeinen akseli on kaikkien muiden akselien parent. Y-akseli on Z-akselin child ja ja X-akseli on Y-akselin child. Jos käännetään Z-akselia, Y- ja X-akseli kääntyy Z-akselin mukana, mutta jos käännetään Y-akselia, vain X-akseli kääntyy Y-akselin mukana. X-akseli taas kääntyy täysin itsenäisesti. (Autodesk, 2016e.)

Tästä euler-rotaatioiden käyttäytymisestä johtuen joissain tapauksissa rotaatiot voivat joutua niin kutsuttuun Gimbal-lukkoon. Jos rotation order on (X, Y, Z) ja käännetään esimerkiksi Y-akselia 90-astetta, X-akseli ja Z-akseli ovat nyt täysin samassa linjassa keskenään. Jos tarkastellaan kuvaa (kuva 3), niin huomataan, että kaksi akselia ovat täysin samassa asennossa, emmekä pysty kääntämään objektia kuin kahteen eri suuntaan. (Wikipedia 2016.) Kuvasta on myös liitteenä videoesimerkki ja skenetiedosto (Liite 1, Liite 24).



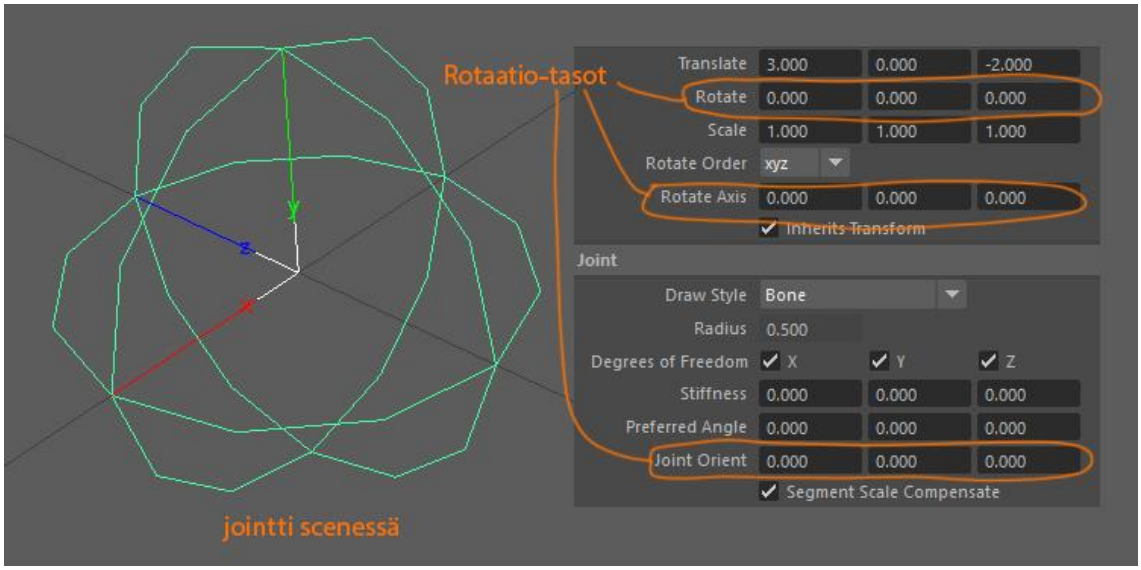
Kuvio 3. Esimerkki gimbal-lukosta

Tämä Gimbal-lukko saattaa aiheuttaa rigaajalle ja animaattorille ongelmia monissa tapauksissa. Yleisesti rigauksessa kannattaa yrittää välttää euler-kulmia monissa tilanteissa, joissa tarkoitus on automatisoida monta rotaatioakselia yhtäaikaisesti. Monet

mayan sisältämistä constrainteista käyttävät jotain muuta matematiikkaa kuin euler-kulmia, kuten quaternion-interpolointia.

3.1.2 Joint-orientaatio ja Rotate-akselit

Transform-objektilla on kaksi rotaatioiden tasoa, rotaatio ja rotate-akselit. Rotaatioita käytetään yleensä animoidessa FK-joint-ketjuja. Rotate-akselien muutos on aina suhteessa rotaatioihin. Voisi ajatella, kuin kyseessä olisi kaksi objektia, joista toinen on toisen child-objekti. Child-objekti rotatoituu aina parent-objektin avaruudessa eli paikallisessa avaruudessa. Yleensä kannattaa pitää rotate-akselit nollattuina, jotta vältytään ylimääräiseltä kompleksisuudelta ja ongelmilta. Jointilla on vielä yksi rotaatioiden taso näiden kahden lisäksi, Joint-orientaatio, joka määrittää jointin local-rotation-akselien orientaatiot. Rotaatioiden muutos tapahtuu aina joint-orientaatioiden jälkeen. (Autodesk 2016f.)



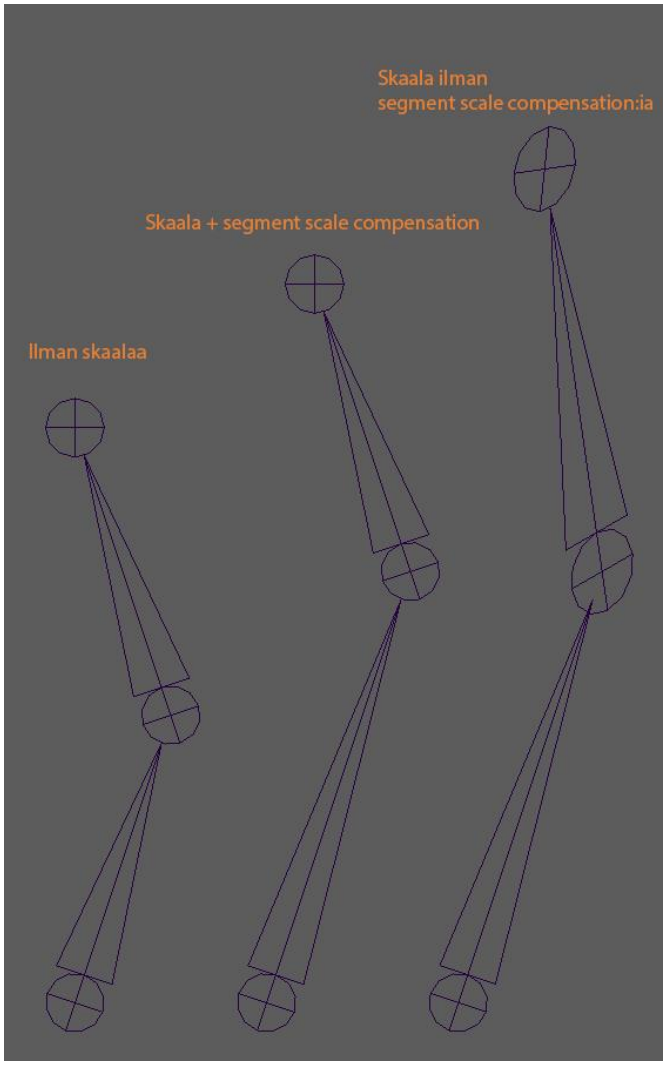
Kuvio 4. Joint ja sen attribuutit

Tämän takia on hyvin tärkeää, että joint-orientaatiot säädetään huolella niiden luonnin jälkeen, jotta objekti rotatoituu haluttuun suuntaan. Joint-orientaatioilla voidaan säätää tarkkaan, käyttäytyykö viereinen jointti peilikuvana vai täysin täysin identtisesti. Esimerkiksi kääntämällä toisen jointin x-orientaatioakselia 180 astetta saadaan y- ja z-akseli kääntämään objektia vastakkaisiin suuntiin. Tämä on hyödyllistä esimerkiksi käsi- ja jalkarigeissä.

Väärin säädetyt joint-orientaatiot voivat aiheuttaa ongelmia myös esimerkiksi IK-solvereissa. Näistä kerron myöhemmin lisää Inverse Kinematics-luvussa (3.3.6).

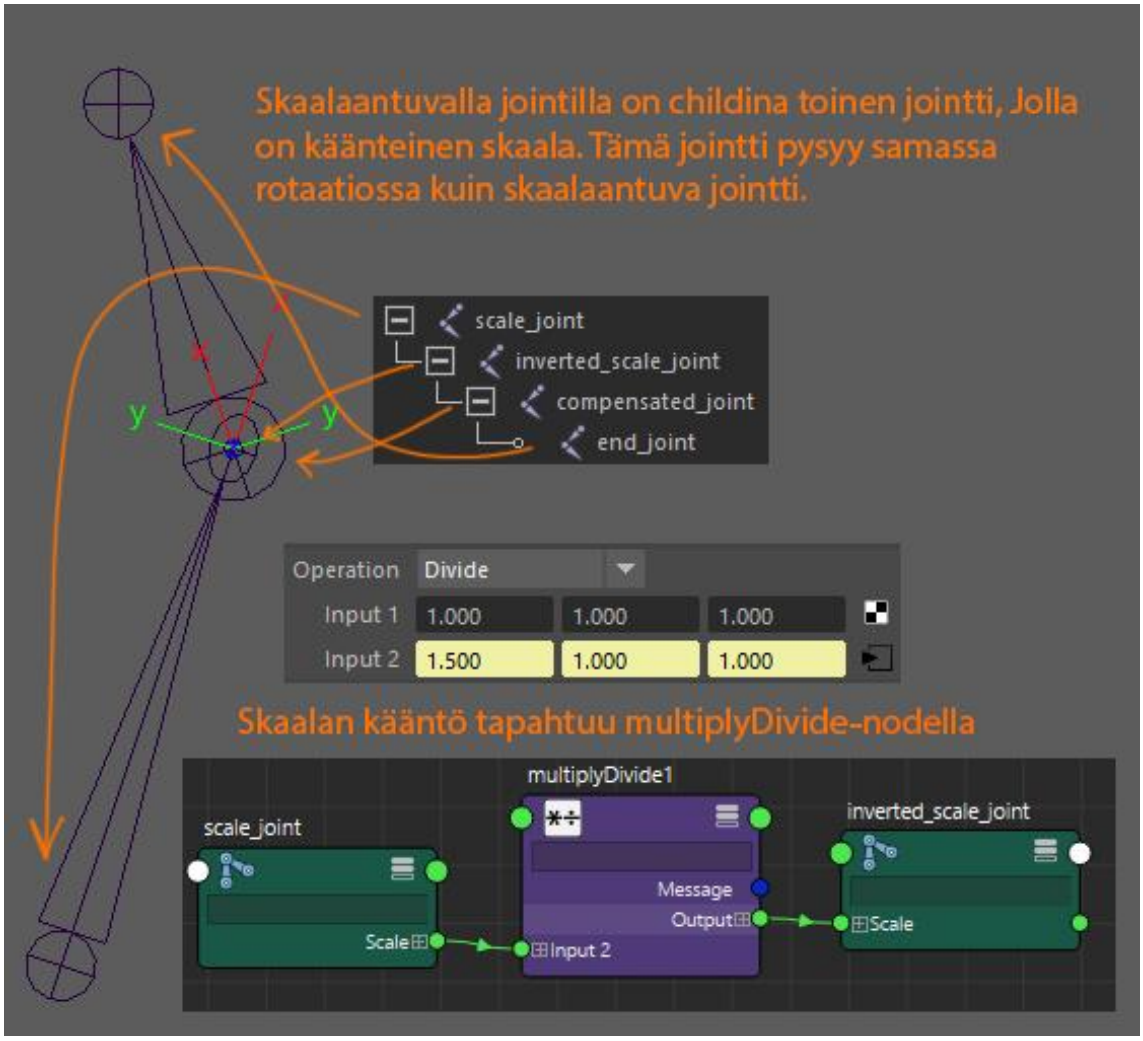
3.1.3 Segment scale compensation

Objektit perivät tavallisesti parent-objektinsa transform-attribuutit, jolloin parent-objektia skaalatessa epätasaisesti child-objekti saa skaala-arvot samassa avaruudessa kuin sen parent-objekti. Jos child-objektia rotatoidaan, sen perimät skaala-arvot pysyvät parent-objektin avarudessa, joka aiheuttaa yleensä epätoivottua venymistä. Jointeilla on segment scale compensation-asetus, joka laskee child-jointeille käänteisein skaalan suhteessa sen parent-objektiin. Tämä on hyvin hyödyllinen ominaisuus varsinkin venyvissä rigeissä.



Kuvio 5. Esmerkkikuva segment scale compensation:in vaikutuksesta.

Lähes kaikki nykyaikaiset pelimoottorit tukevat epätasaista jointtien skaalausta, mutta esimerkiksi Unity-pelimoottori ei tue Mayan segment scale compensation-asetusta. Tämä ominaisuus kannattaa poistaa käytöstä, jos rigi on tarkoitus siirtää pelimoottoriin (Autodesk 2016g). Silloin täytyy huolehtia siitä, että jointit, joille ei halua periä skaalaa, eivät ole samassa hierarkiassa skaalautuvien jointtien kanssa.



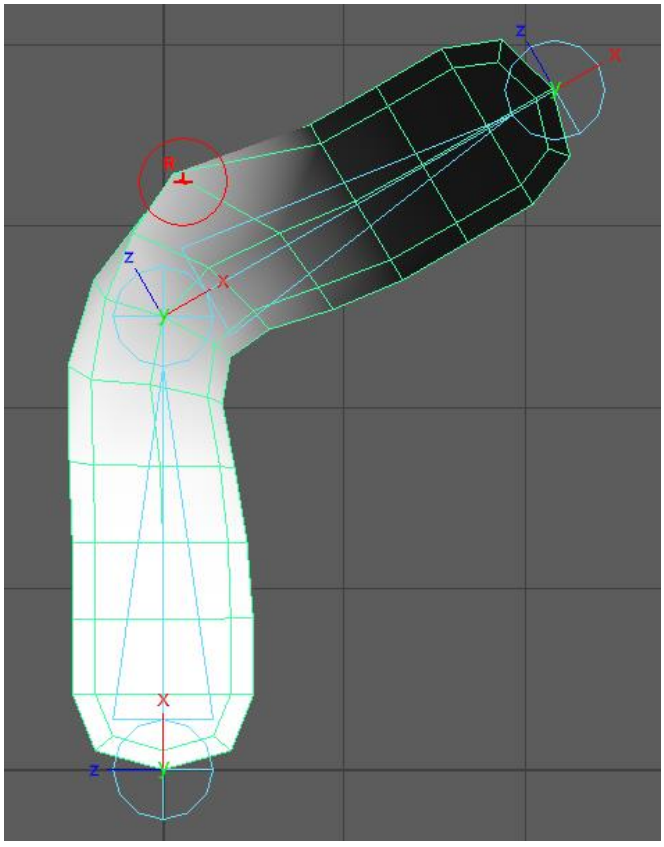
Kuvio 6. Skaalan kompensointi manuaalisesti tehtynä

Kuva 6:ssa skaalan kompensointi on tehty manuaalisesti antamalla skaalantuvalle jointille kompensoiva child-jointti, jolla on käänteinen skaala. Käänteinen skaala saadaan jakamalla 1.0 skaalautuvan jointin skaalalla. Tämä tapahtuu helposti yhdistämällä skaalautuvan jointin skaala multiplyDivide-noden input2-attribuuttiin ja multiplyDivide-noden output-attribuutti kompensoivan jointin skaalaan. MultipleDivide-noden operation-attribuutti täytyy myös muuttaa divide-moodiin. Tästä on liitteenä skenetiedosto (liite 29).

Tämä kompensointimetodi toimii pelimoottoreissa, jotka tukevat epätasaista skaalausta. Se myös kulkee animaatiotiedoston mukana pelimoottoriin. Käänteinen skaala on myös suhteellisen helppo ohjelmoida itse pelimoottorissa, jos rigin toiminnallisuus ei saa olla riippuvainen animaatiodatasta.

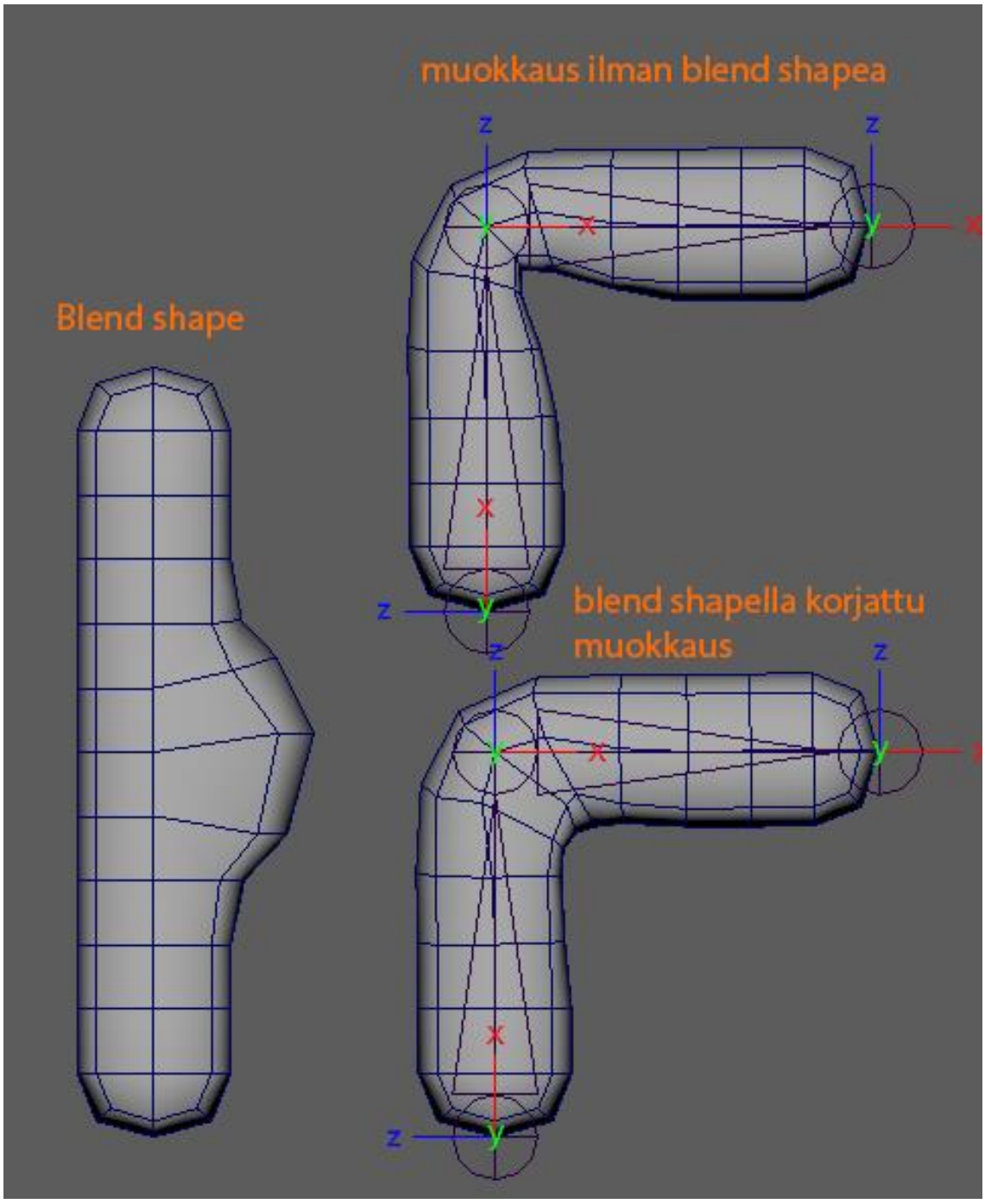
3.2 Skinnaus

Skinnauksella tarkoitetaan metodia, jolla polygonigeometria saadaan seuraamaan muokkaajia halutulla tavalla. Yleisin pelimoottoreissa käytetty järjestelmä on smooth skinnausalgoritmi. Smooth-skinnauksessa jokainen verteksi saa weight-arvon 1 jaettuna halutuille jointeille. Pelimoottoreissa yleisesti maksimi vaikuttajien määrä on 4, eli yksi verteksi voi ottaa vaikutusta neljältä eri jointilta (Autodesk 2016h).



Kuvio 7. Weightien maalausprosessi. Punainen ympyrä on maalaustyökalun osoitin. Valkoinen väri näyttää vaikutusalueen valitulle jointille, joka tässä tapauksessa on joint-ketjun ensimmäinen.

On muitakin polygonien muokkaamiseen tarkoitettuja metodeja, jotka toimivat moderneissa pelimoottoreissa. Yksi näistä metodeista on blend shape, jossa muokattavasta mallista tehdään duplikaatti. Tämän duplikaatin muotoa muokkaamalla alkuperäisen mallin muoto muuttuu (Autodesk, 2016i). Tästä on myös esimerkkivideo ja skenetiedosto liitteenä (liite 5, liite 20).



Kuvio 8. Korjaus-blend shape

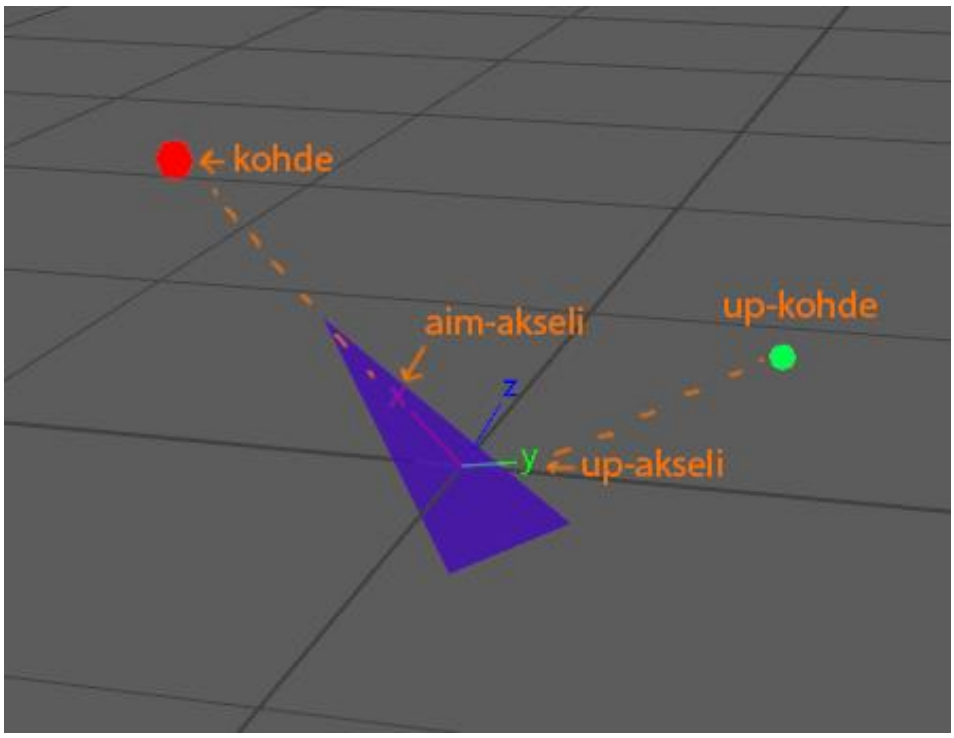
Kolmas muokkausmetodi, joka toimii monissa pelimoottoreissa, on verteksianimaatio. Tämä joudutaan lähes kokonaan ohjelmoimaan itse pelimoottorissa algoritmilla. Modernit pelimoottorit sisältävät myös reaaliaikaisen vaatteiden fysiikkasimulaatiomoottorin, joka mahdollisesti helpottaa animaattorin ja rigaajan työtä.

3.3 Constraintit

Constraintit ovat nodeja, joilla transform-objekti saadaan seuraamaan toista transform-objektia manipuloimalla niiden transform-attribuutteja eri tavoilla. Constrainteilla saadaan objetit seuraamaan toisia objekteja maailman avaruudessa riippumatta objektien sijainnista hierarkiassa. (Autodesk, 2016j.)

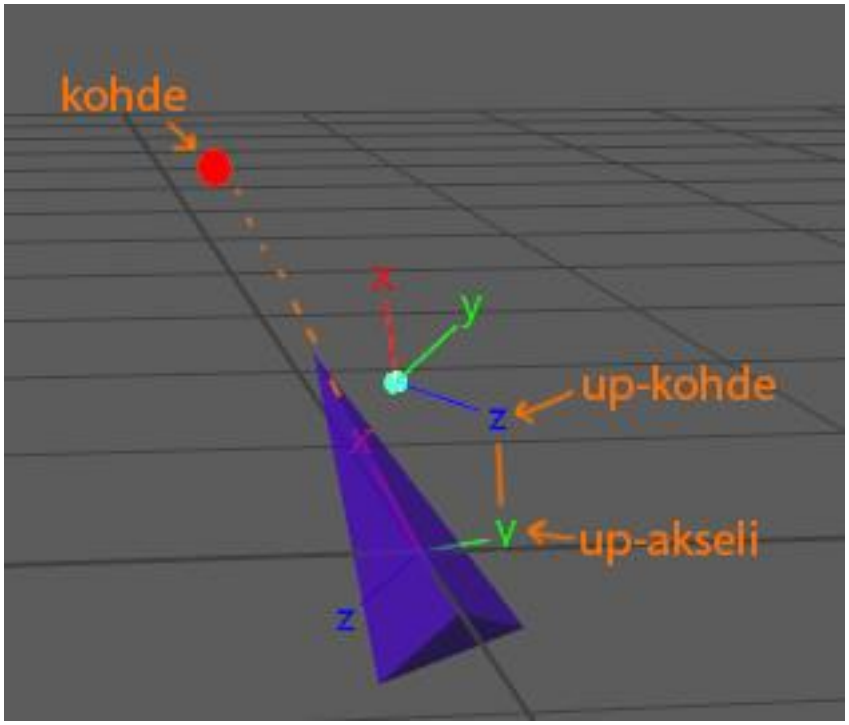
3.3.1 Aim-constraint

Aim-constraintilla saadaan objekti tähtäämään toista objektia. Jotta aim-constraintin saa toimimaan halutulla tavalla, sille pitää määrittää aim-akseli, up-akseli ja millä tavalla up-akselin halutaan toimivan. (liite 16.)



Kuvio 9. Aim-constraintin toiminta

Up-akselin asetukset ovat vector, object-up, object-rotation-up ja world-up. Vector-moodissa käyttäjä voi itse valita suuntavektorin, joka voidaan laskea vaikka vektorimatematiikalla.Object-up-moodissa up-akseli yrittää tähdätä haluttuun kohteeseen. Object-rotation-up-moodissa up-akseli yrittää tähdätä samaan suuntaan kuin haluttu kohteen akseli. kuvassa 10 objektin Y-akseli yrittää osoittaa samaan suuntaan kuin up-kohteen Z-akseli. (liite 17.)



Kuvio 10. Esimerkki aim-constraintin object-rotation-up-asetuksesta.

3.3.2 Parent-constraint

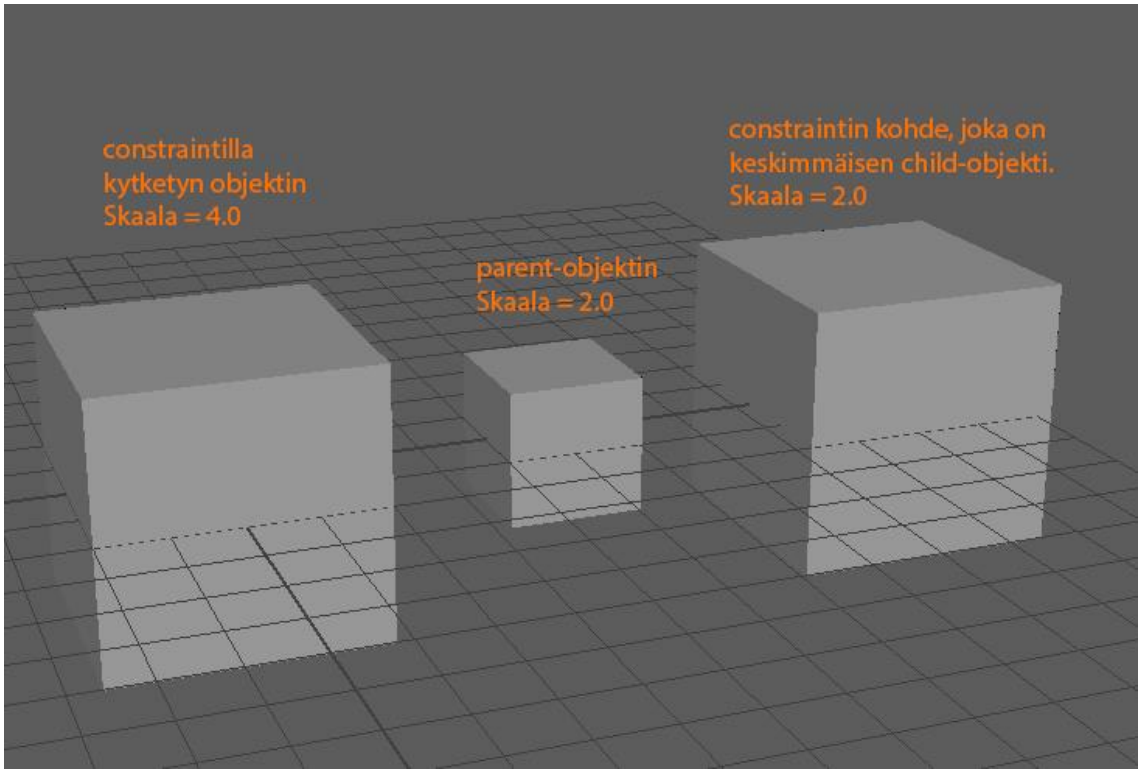
Parent-constraint simuloi varsinaista parentointia. Sillä saadaan objekti seuraamaan toista objektia rotaatiossa ja translaatiossa. Translaatioon vaikuttaa myös kohteen rotaatio, eli jos kohde ja constraintilla kytketty objekti ovat eri paikassa skenessä ja kohdetta rotatoidaan, niin constraintilla kytketty objekti liikkuu kohteen mukana aivan kuin se olisi kohteen child-objekti. Constraintilla kytketty objekti ei kuitenkaan peri skaalaa, mitä voisi constraintin nimestä päätellen ajatella.

3.3.3 Point-constraint

Point-constraintilla kytketty objekti seuraa yhtä tai usempaa kohdetta käyttäen vain positiota. Kohteen rotaatio ei tässä tapauksessa vaikuta constraintilla kytkettyyn objektiin millään tavalla (Liite 35).

3.3.4 Scale-constraint

Scale-constraintilla kytketty objekti perii kohteiden skaalan. Tähän skaalaukseen vaikuttaa kaikki hierarkian sisällä peritty skaalaus.



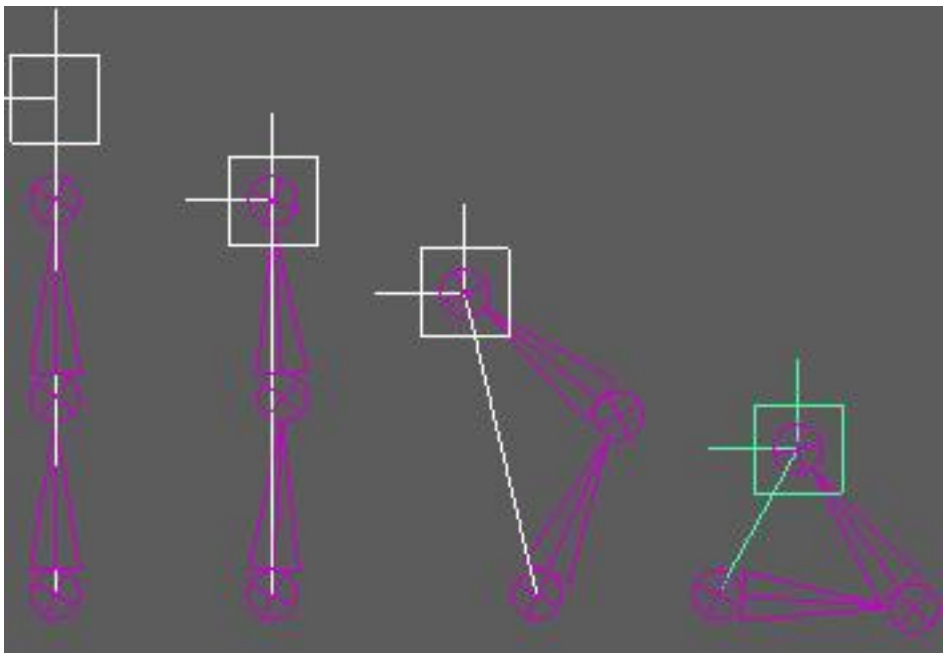
Kuvio 11. Esimerkki scale-constraintin periytyvästä skaalasta.

3.3.5 Orient-constraint

Orient-constraint kääntää objektin samaan rotaatioon kohteen kanssa. Jos kohteita on enemmän kuin yksi, constraintattavan objektin rotaatiot saattavat hypätä 180 astetta, kun kohdetta kääntää tarpeeksi paljon. Tätä varten orient-constraintissa on muutama erilainen asetus: no-flip, average, shortest ja longest, jotka saattavat korjata joitakin hankalia tapauksia (Liite 31).

3.3.6 Inverse Kinematics (IK)

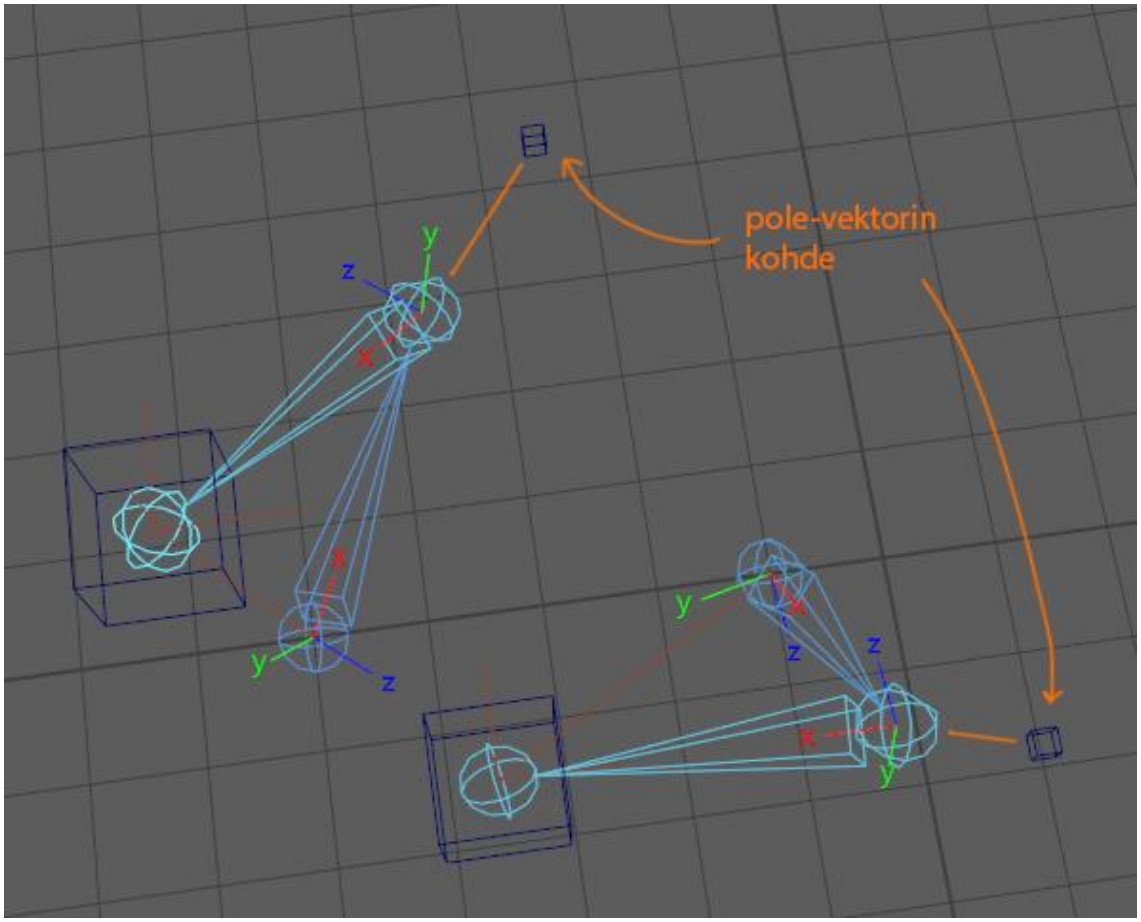
Animaatiossa yksi yleisimpiä tapoja manipuloida rigiä on inverse kinematics tai lyhennettynä IK. IK:ssa rotaatiot lasketaan siten, että jointtiketjun viimeinen jointti on siirtynyt vapaasti liikuteltavan kontrolliobjektin kohdalle. Jos kontrolliobjektin vie tarpeeksi kauas, jointtiketju vääntyy suoraksi yrittäen kurottaa niin lähelle kontrolliobjektia kuin mahdollista (Autodesk 2016k). Liitteinä myös video ja skenetiedosto (liite 4, liite 27).



Kuvio 12. Esimerkki IK:n toiminnasta.

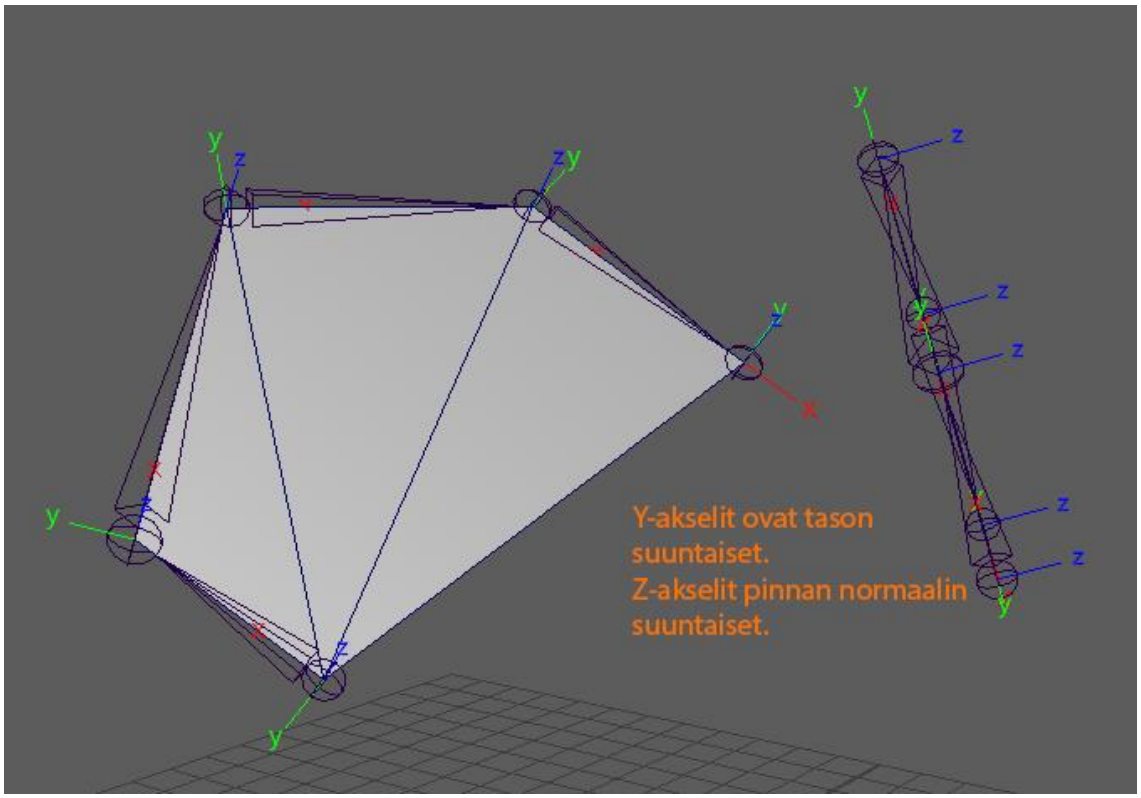
IK:sta on apua esimerkiksi jalkoja animoidessa kävelyanimaatioihin, tai kädet voi saada helposti pysymään kohteessa muun vartalon liikkuessa esimerkiksi kiipeämisessä tai nojatessa johonkin.

IK-solvereita on kahdenlaisia, rotate-plane-solver ja single-chain-solver. Rotate-plane-solverissa jointtiketjun taittumissuunta määräytyy pole-vektorin avulla, joka toimii täysin samalla tavalla kuin aim-constraintin up-vektori ja sisältää täysin samat asetukset. Single-chain-solverissa taittumissuunnan määrittää IK-handle-objektin rotaatio. Tästä on esimerkki olkapään twistin korjauksessa (Liite 40). Tähän syvennyn tarkemmin luvussa Rigisovellukset (4.2).



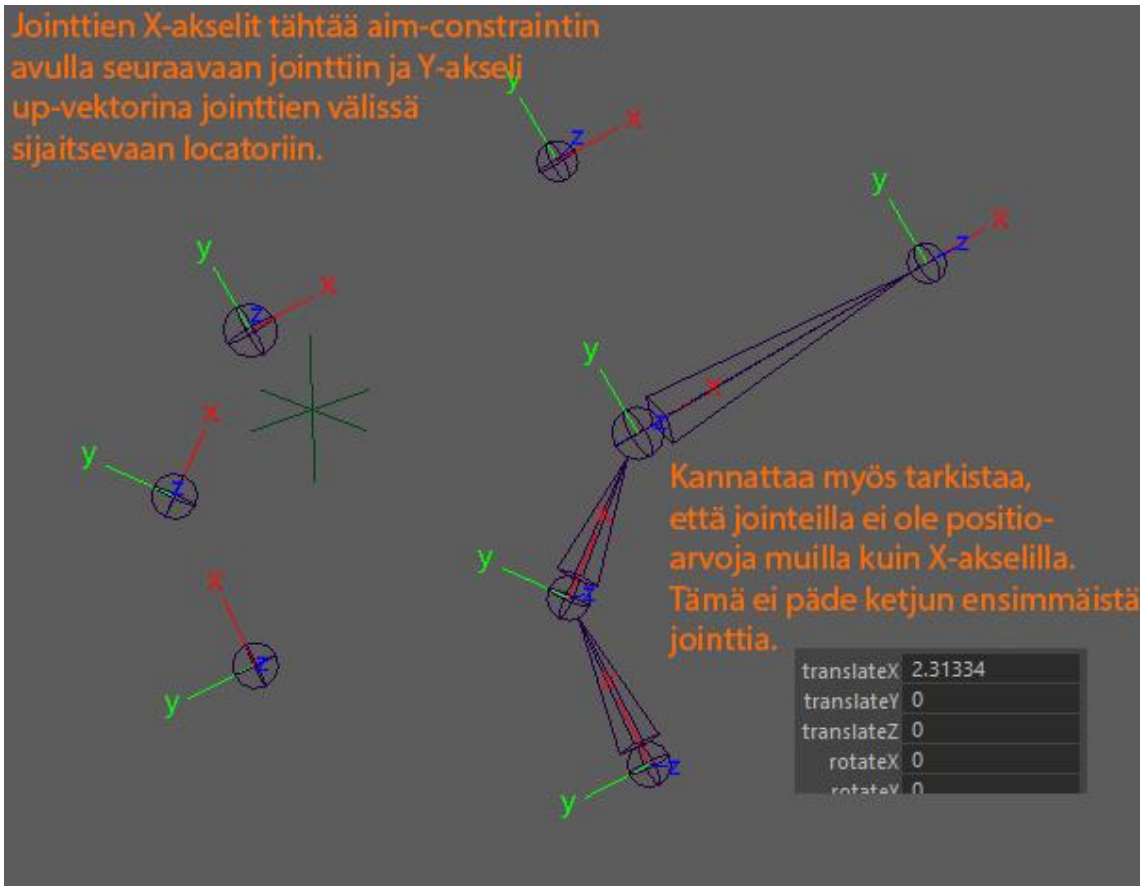
Kuvio 13. Esimerkki pole-vektorista.

Jotta IK-ketjun rotaatiot saadaan toimimaan järkevästi, aluksi kannattaa varmistaa, että ketjun jointit muodostavat tasaisen pinnan ja jointtien rotaatioakselit on kohdistettu johdonmukaisesti (kuva 14). Yksi akseli osoittaa seuraavaan jointtiin ja muut akselit jointtien välille muodostuvan tason mukaisesti (liite 43).



Kuvio 14. Jointtiketjun välille muodostuva taso

Jointtien akselien kohdistaminen onnistuu asettamalla jointtiketjun keskelle transform-objekti point-constraintin avulla. Sen jälkeen jointtien parentointi poistetaan ja jokainen jointti laitetaan tähtäämään seuraavaan jointtiin aim-constraintin avulla. Up-vektoriksi asetetaan jointti-ketjun keskellä oleva transform-objekti. Up-akseliksi voi valita joko Y- tai Z-akselin. Valitsin Y-akselin, koska se on yleisimmin käytetty rigeissä.

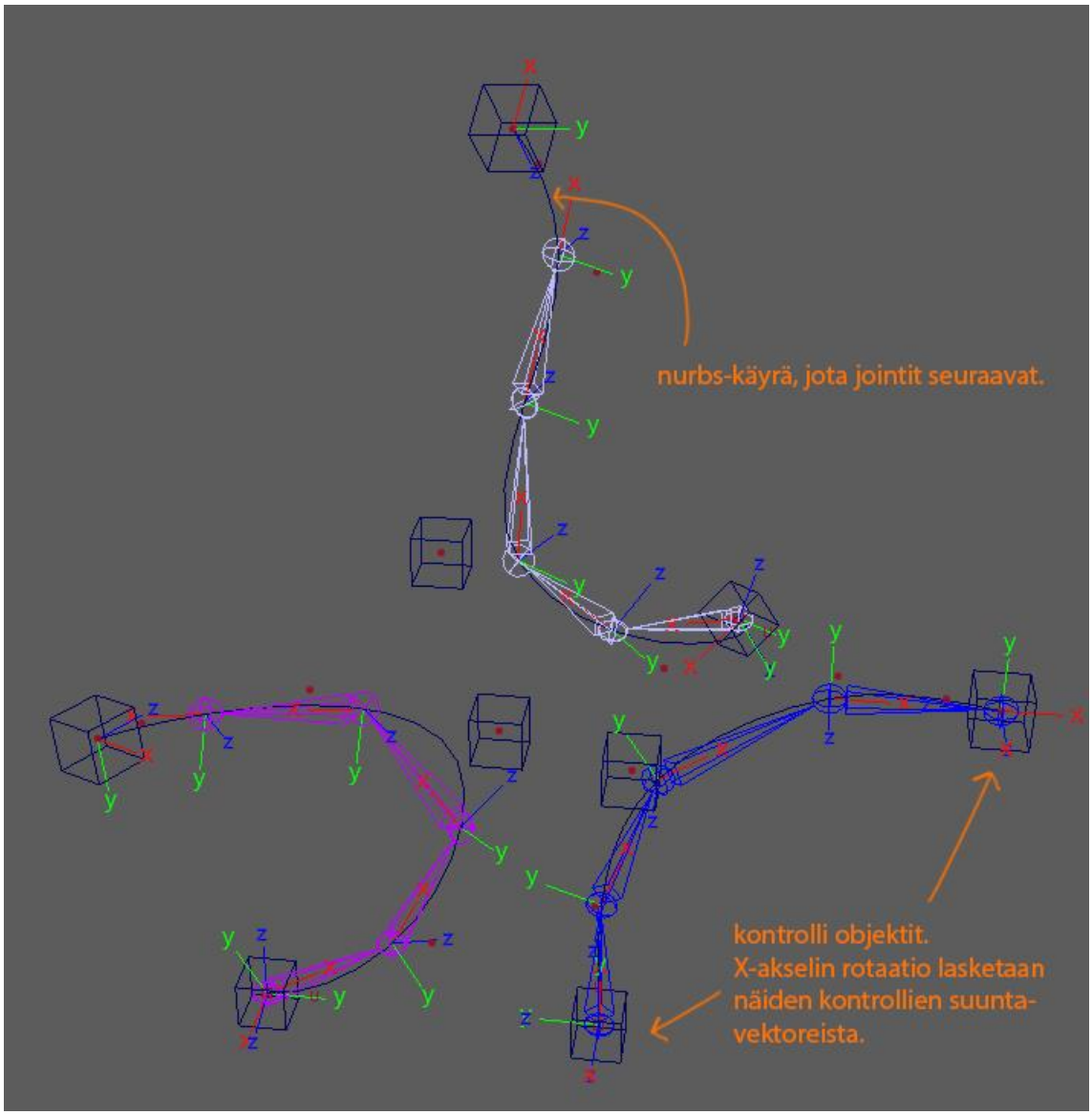


Kuvio 15. Jointtien akselien suuntaus

Tämän jälkeen constraintit voidaan poistaa, jointit uudelleen parentoida hierarkiaksi ja lopuksi jointeilta jäädyttää rotaatiot freeze transformation-toiminnolla, jolloin rotaatioarvot siirtyvät joint-orient-attribuuteille. Ketjun viimeisen jointin rotaatiot ja orientaatiot kannattaa nollata.

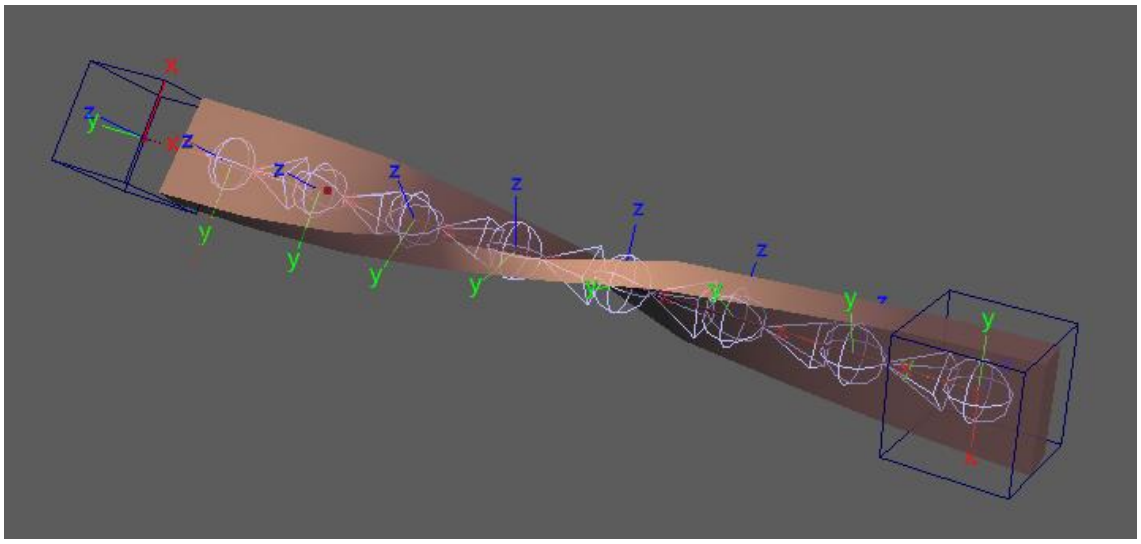
3.3.7 Spline-IK

Spline-IK on järjestelmä, jossa jointtiketju mukautuu nurbs-käyrän muotoon pitämällä alkuperäisen pituutensa (Autodesk 2016l). Liitteenä video ja skenetiedosto (liite 2, liite 42).



Kuvio 16. Esimerkki spline-ik:sta

Käyrän tangentti kykenee antamaan ainoastaan rajatun määrän informaatiota vastaavalla tavalla kuin aim-constraintin kohteesta saatavan suuntavektorin lisäksi tarvittiin upvektori. Tätä varten spline IK:lle tarvitaan twist-kontrolli, joka laskee jokaiselle jointille suunnan, johon Y- ja Z-akselit kuuluu osoittaa. Tähän löytyy IK-spline-solverista pari vaihtoehtoa. Ehkä kätevin vaihtoehto näistä on Advanced Twist Control (liite 15).

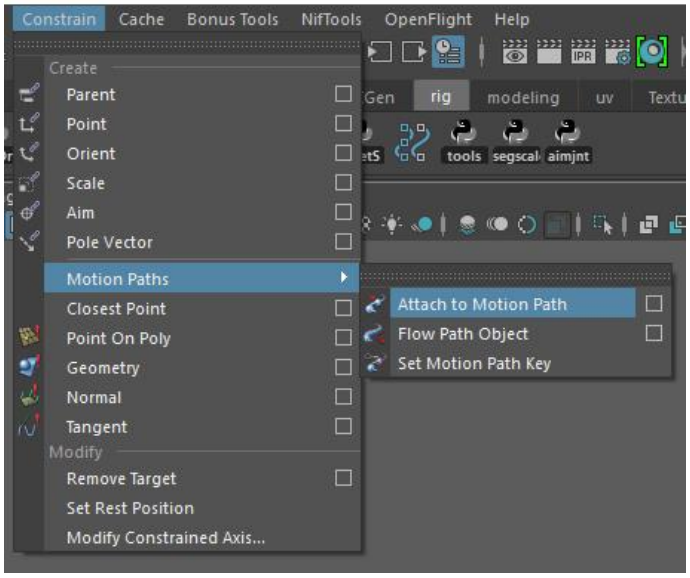


Kuvio 17. Esimerkki advanced twist controlista, jossa jointtien negatiivinen Y-akseli osoittaa kontrolliobjektien X-akselia. Y-akselien rotaatiot on interpoloitu näiden kahden kontrollien välille.

3.3.8 MotionPath-node

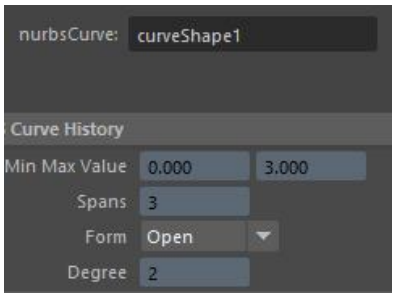
MotionPath-nodella voidaan kytkeä yksittäisiä transformeja tai jointteja käyrälle (Autodesk 2016m). Kytketyt objektit rotatoituvat käyrän tangentin mukaisesti ja seuraavat positiossa käyrää, jota voi säätää motionPath-noden uValue-attribuutilla. MotionPath-node tarvitsee myös up-vektorin samalla tavalla kuin muutkin constraintit ja spline-IK.

Jos motionPath-node luodaan attach to motion path-toiminnolla, uValue-attribuuttiin kytketään automaattisesti animCurve-node, joka liikuttaa constraintattua objektia skenen ajan mukana. Tämä kannattaa tässä tarkoituksessa kytkeä pois, jos tarkoituksena on käyttää motion path nodea rigissä.



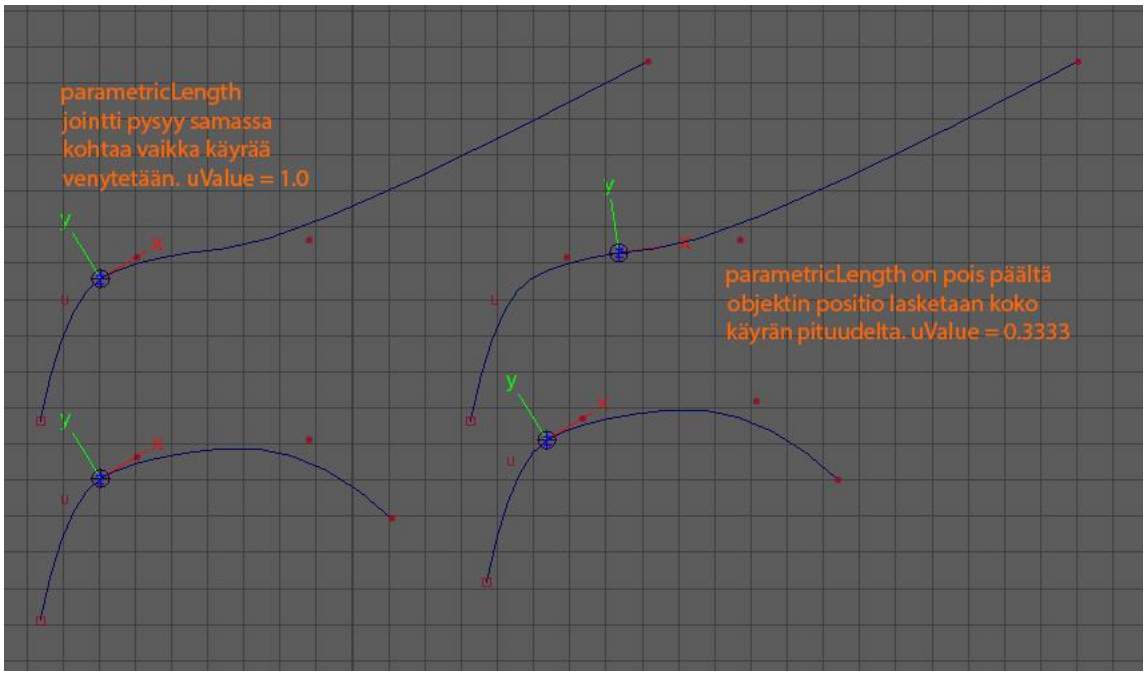
Kuvio 18. Attach to motion path toiminto.

MotionPath-nodessa on kaksi eri attribuuttia, jotka määrittävät, miten constraintilla kytketyt objektit seuraavat käyrää. Ensimmäinen asetus on follow-attribuutti, joka määrittää, halutaanko objektin rotaation seuraavan kurvin tangenttia. Toinen on parametricLength. Jos parametricLength on päällä, motion-pathin uValuen maksimiarvo on käyrän spans-attribuutin arvo (liite 34).



Kuvio 19. Spans-attribuutti löytyy käyrän shape-nodesta.

Tällä pystytään vaikuttamaan siihen, lasketaanko constraintattujen objektien positio käyrän koko pituudesta vai sen pituudesta jaettuna osiin. Jos objekti on kytketty käyrän alkupäähän ja käyrää venytetään viimeisestä control-pointista, objekti ei liiku.

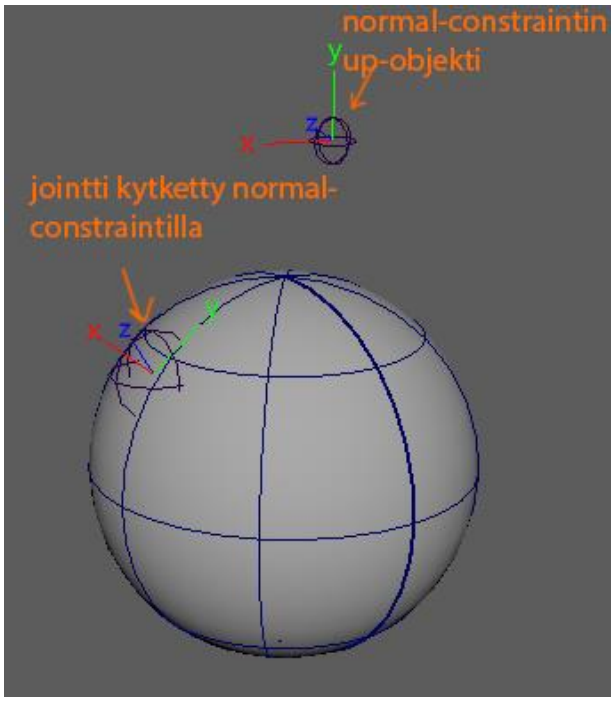


Kuvio 20. Esimerkki parametric length-attribuutista.

MotionPath-noden etu verrattuna spline IK:hon, on sen mahdollistama tarkempi kontrolli objektien positiossa. Heikkoutena on, että up-vektori joudutaan määrittämään jokaiselle yksittäiselle objektille erikseen, mikä tekee advanced twist controllin tapaista toimintoa erittäin hankalaa rakentaa. Sitä varten on kehitetty ratkaisuja, joista kerron lisää luvussa Ribbon-spine (4.1).

3.3.9 Normal-constraint

Normal-constraint pakottaa objektin seuraamaan geometrian pinnan normaalia halutulla akselilla. Tämä node tarvitsee myös up-vektorin kuten aikaisemmin mainitut nodet, ja asetukset ovat myös samoja (liite 30).



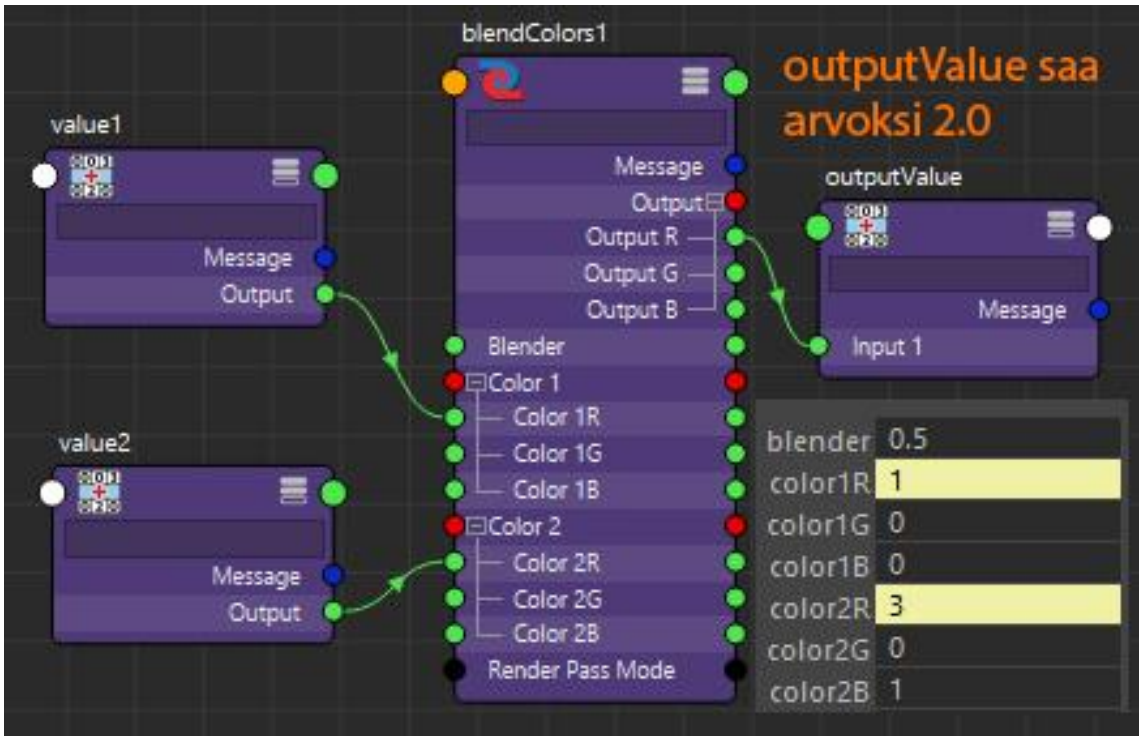
Kuvio 21. Esimerkki normal-constraintista

3.4 Utility-nodet

Mayassa on pitkä lista niin kutsuttuja utility-nodeja, joiden alkuperäinen tarkoitus on muokata tekstuureita. Tästä johtuen monet näistä nodeista on nimetty tarkoituksensa mukaisesti, kuten esimerkiksi blendColors. Moni näistä nodeista on kuitenkin erittäin hyödyllisiä rigauksessa. BlendColors-nodea voi käyttää esimerkiksi vaihtamaan monia rigien toimintoja pehmeästi. Mayassa näitä Utility nodeja on niin monia, että voin mainita näistä vain yleisimmät. (Autodesk 2016n.)

3.4.1 blendColors

BlendColors-node sekoittaa kahta arvoa keskenään weight-arvon suhteessa. Esimerkiksi jos blendColors-noden blender-attribuutin arvo on 0,5, color1R on 1,0 ja color2R on 3,0, niin output arvoksi muodostuu 2,0. koska se on tasan arvojen 1,0 ja 3,0 välissä.



Kuvio 22. Esimerkki blendColors-noden toiminnasta.

BlendColors-node pystyy ottamaan sisään myös vektoreita. Tämä tarkoittaa, että input-attribuuttiin voi kytkeä suoraan objektin translate-arvot ilman, että kaikki arvot joutuisi kytkemään erikseen. Tämä pätee moniin muihinkin utility-nodeihin.

3.4.2 multipyDivide

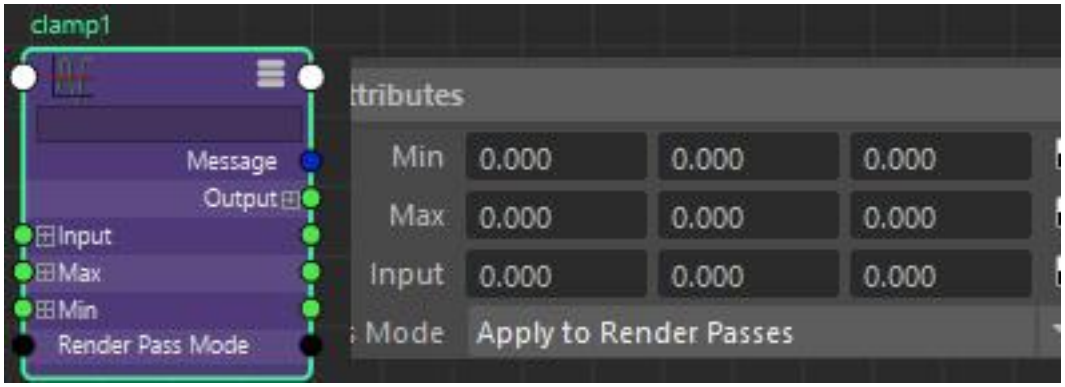
Tällä nodella kerrotaan, jaetaan tai nostetaan arvo potenssiin. MultiplyDivide myös pystyy ottamaan vastaan vektoreita. Se on erittäin hyödyllinen node monessa tarkoituksessa, joita käsittelen lisää luvussa Rigisovellukset (4).

3.4.3 addDoubleLinear

Tämä node yksinkertaisesti summaa kaksi lukua. AddDoubleLinear ottaa sisään vain yhden float-arvon. Jos halutaan käsitellä vektoreita, joutuu näitä nodeja tekemään yhden jokaista akselia varten.

3.4.4 clamp

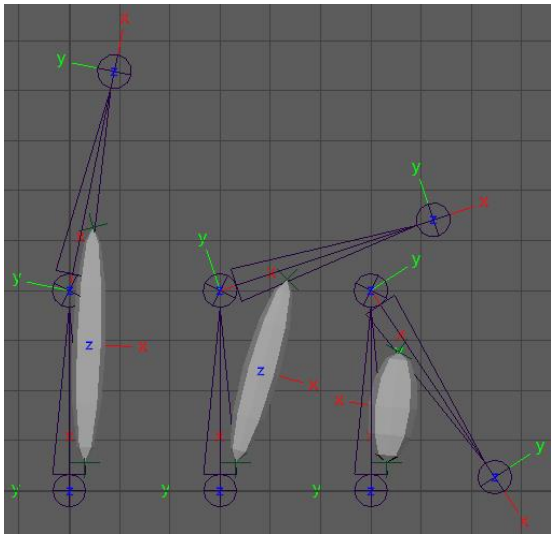
Jos arvolle halutaan asettaa ylä- tai alaraja, clamp nodella voi sen tehdä. Input-attribuuttiin kytketään arvo, jota halutaan rajoittaa. Min-arvo määrittää minimiarvon ja vastaavasti max-arvo maksimi-arvon. Tämä node ottaa myös vastaan vektoreita.



Kuvio 23. Clamp-noden atribuutit

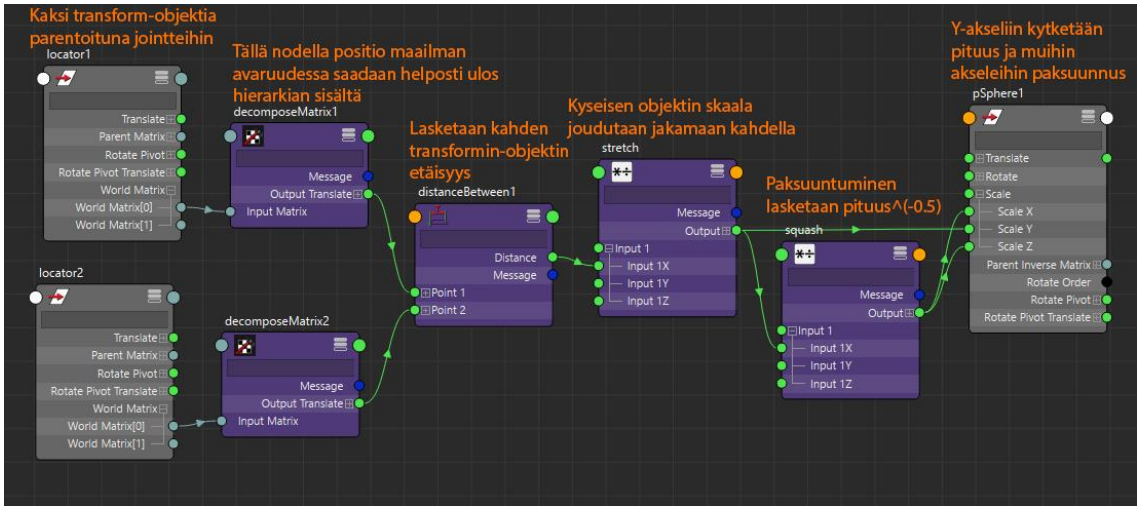
3.4.5 distanceBetween

Tämä node laskee etäisyyden kahden transformin välillä. Se ottaa vastaan vektorin ja antaa ulos yhden arvon. Tämä node on erittäin hyödyllinen monessa tapauksessa, jossa täytyy automatisoida jokin toiminto yhdellä arvolla. Esimerkiksi venyvissä rigeissä tätä voidaan käyttää laskemaan, kuinka paljon jointin kuuluu paksuuntua ja venyä riippuen kahden jointin etäisyydestä toisiinsa (Liite 22).



Kuvio 24. distanceBetween-node lihassimulaatiossa.

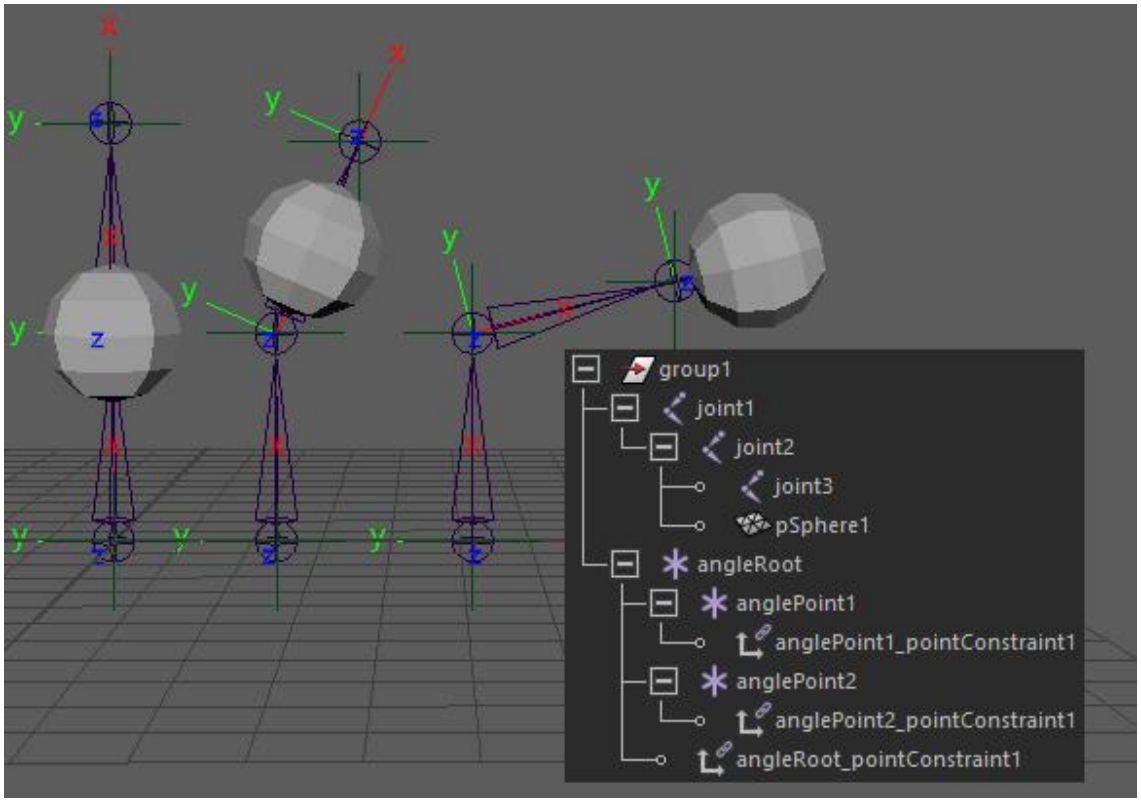
Ylemmässä esimerkissä (kuva 24) distanceBetween-nodea käytetään säätelemään polygonigeometrian pituutta ja paksuutta. Etäisyys lasketaan kahden locatorin välille, jotka ovat parentoitu jointteihin. Kun jointtia kääntää, locatorit lähestyvät toisiansa. Tällä etäisyyden muutoksella pystytään manipuloimaan objektin skaalaa. Kuvassa 25 näkyy nodepuu kyseisestä lihassimulaatiosta.



Kuvio 25. Esimerkki distanceBetween-noden käytöstä.

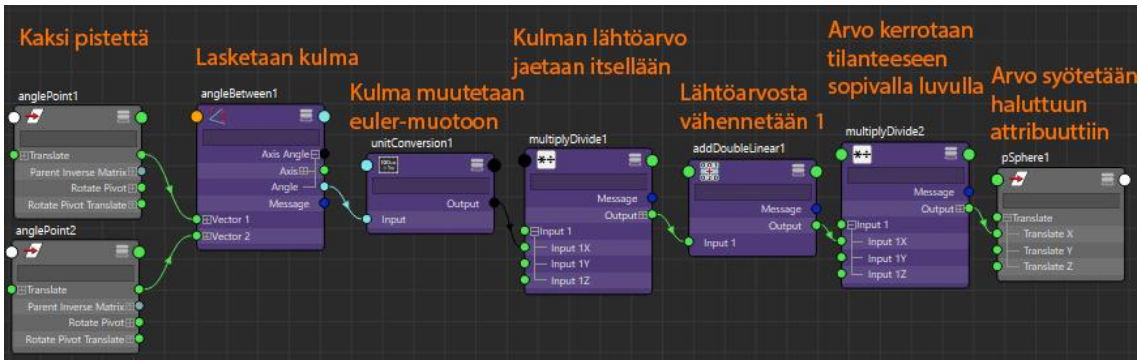
3.4.6 angleBetween

Tämä node toimii lähes samalla tavalla kuin distanceBetween-node, paitsi tässä lasketaan kahden pisteen kulma toistensa välillä. Kulman laskemiseen tarvitaan aina kolmio, joka muodostuu kolmesta pisteestä. AngleBetween ottaa sisään kuitenkin vain 2 kulmaa. Node olettaa, että tämä kolmas piste on aina nollavektori, joten tämän noden käytössä tarvitaan hieman enemmän huolellisuutta, jotta saadaan käytettäviä arvoja ulos. Tässä tapauksessa kulma kannattaa lähes aina laskea lokaalissa avaruudessa siten, että ne kaksi pistettä, joiden väliltä kulma lasketaan, ovat jonkun toisen transform-objektin child-objekteja. Tällä tavalla parent-transform-objektia siirtelemällä pystytään muuttamaan nollavektorin paikkaa, josta kulma lasketaan (liite 19).



Kuvio 26. Esimerkki angleBetween-noden käytöstä. Kun kulma pienenee, pallo siirtyy kauemmaksi alkuperäisestä paikasta.

Tässä vielä esimerkki nodepuusta, jossa angleBetween-nodea käytetään (kuva 27). Tämä kyseinen rigi ei välttämättä ole kaikkein hyödyllisin, mutta se toimii esimerkkinä, miten angleBetween-nodeen perustuvan järjestelmän voi rakentaa.

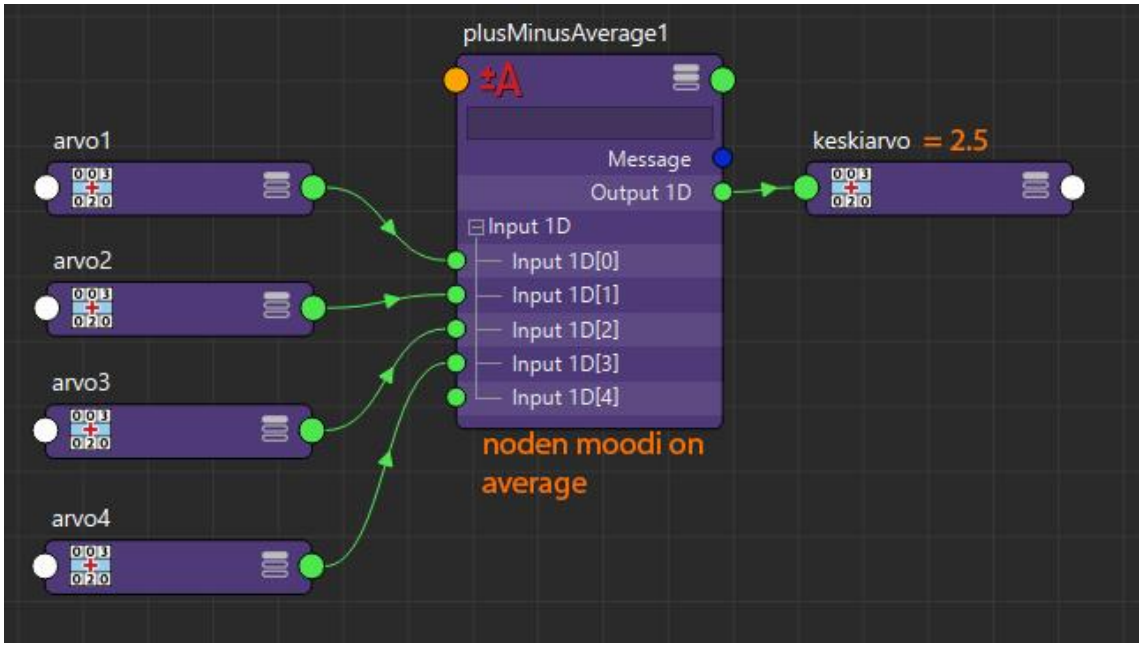


Kuvio 27. Nodepuu, jossa käytetään angleBetween nodea.

Tämä node on erittäin hyödyllinen rakentaessa korjausmekanismeja, esimerkiksi olkapäiden muokkaukseen tai jalkojen laajoihin liikeratoihin, joissa rotaatioita tapahtuu kaikilla akseleilla.

3.4.7 plusMinusAverage

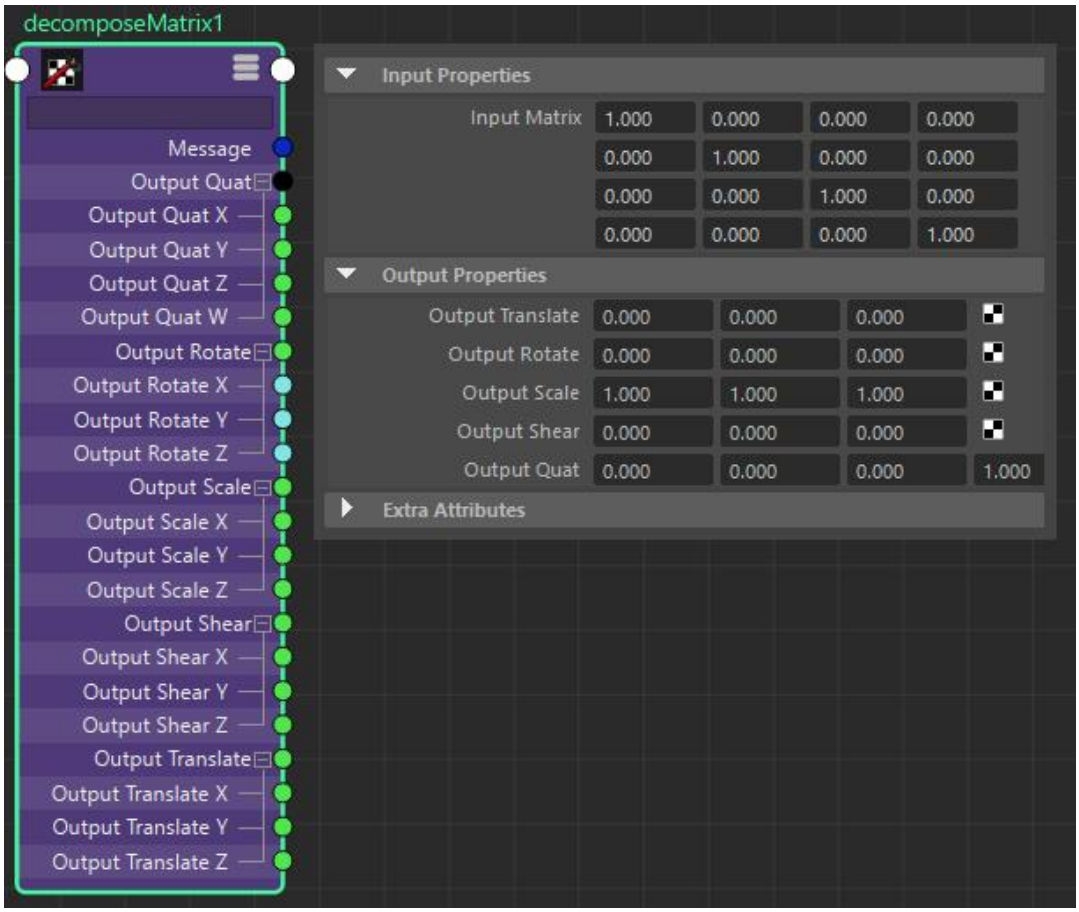
Tällä nodella voi summata, vähentää tai laskea arvojen keskiarvon. Tämä node ottaa sisään joko yksittäisiä arvoja, 2D-vektoreita tai 3D-vektoreita. Laskettavien arvojen määrällä ei ole ylärajaa. Tämä node on hyödyllinen, jos rigissä tarvitsee esimerkiksi summata monia eri arvoja yhteen. Tämä node vähentää tarvittavien nodejen määrää siten yksinkertaistaa nodepuuta.



Kuvio 28. Esimerkki plusMinusAverage-nodesta.

3.4.8 decomposeMatrix

Tämä node on erittäin hyödyllinen, jos transformin-atribuutit halutaan saada ulos maailman avaruudessa. Tämän noden monipuolisempi käyttö tosin vaatii syvällisempää tuntemusta matriisimatematiikasta ja erityisesti 3D-matematiikan transform-matriisien operoimisesta. Jokaisessa 3D-ohjelmassa saatetaan matriiseja käsitellä hieman eri tavoilla.



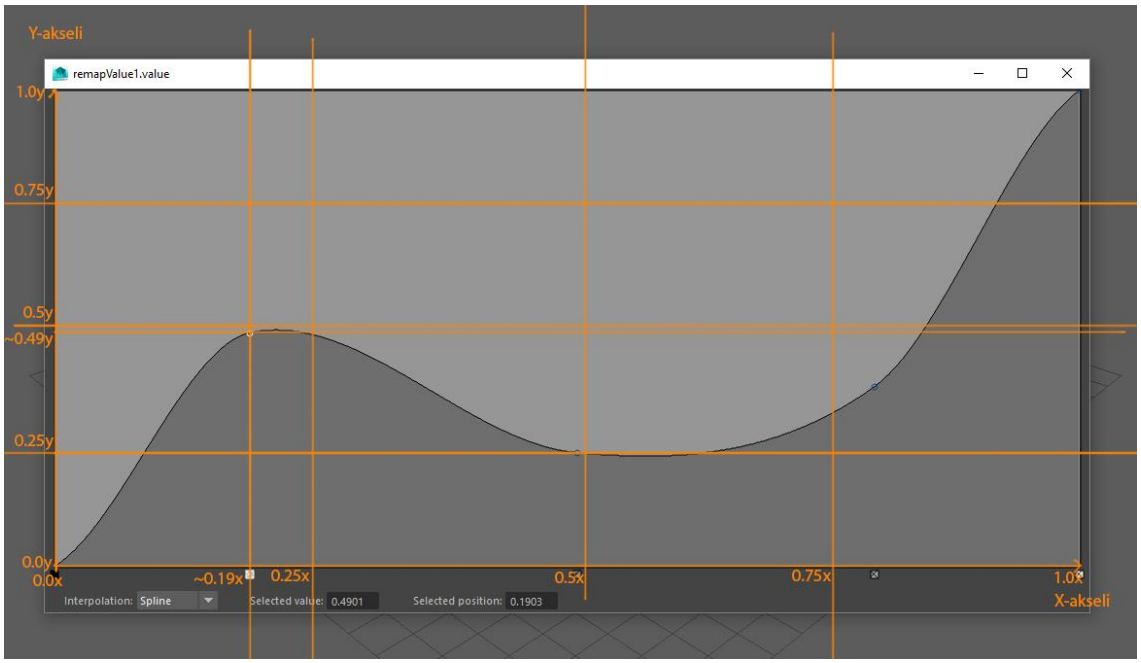
Kuvio 29. decomposeMatrix-noden atribuutit

3.4.9 reverse

Tällä nodella saadaan attribuutin arvo käänteiseksi matemaattisella kaavalla x = 1-y. Jos x:n arvo on 1, niin y:n arvo on 0. Jos y:n arvo on 1, niin x:n arvo on 0.

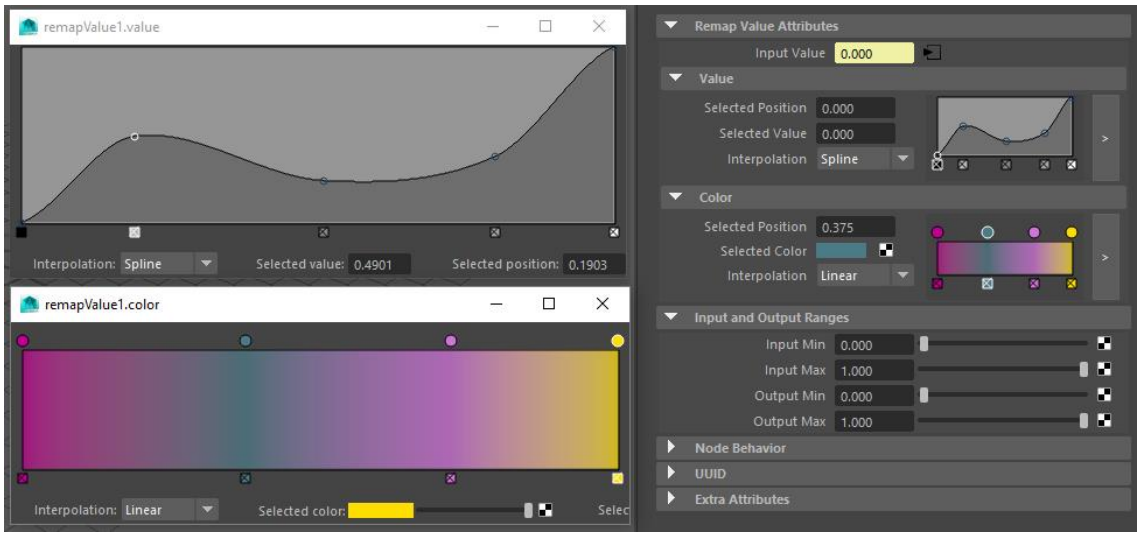
3.4.10 remapValue

remapValue-node toimii hieman samalla tavalla kuin set driven key, josta kerron lisää luvussa Set driven key (3.6). Tällä saadaan aikaiseksi epälineaarisia attribuuttien riippuvuus suhteita käyrän tai värigradientin avulla. RemapValue-noden toiminnan voi helpoiten havainnollistaa kuvalla. Kuvan X-akseli on remapValue-noden inputValue-attribuutti ja Y-akseli output-attribuutti (kuva 30).



Kuvio 30. remapValue-esimerkki

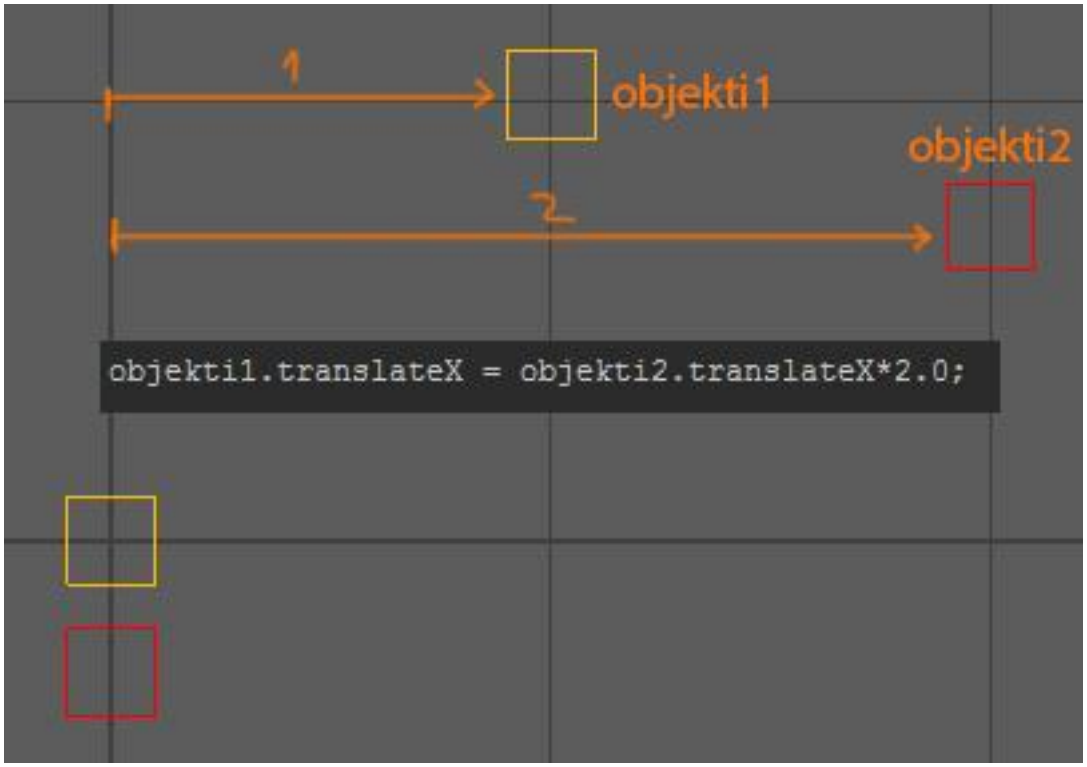
Esimerkkikuvassa inputValue-attribuutin arvolla 0,19 saadaan output-attribuutin arvoksi n. 0,49. Värigradientti toimii täysin samalla tavalla, paitsi jos kyseessä on kolmiulotteinen kartta, joka on helpoin havainnollistaa värigradientilla.



Kuvio 31. remapValue esimerkki

3.5 Ekspressiot

Expressio eli ekspressio on suoraan käännettynä matemaattinen lauseke, jolla voidaan tehdä epäsuoria riippuvuuksia objektien attribuuttien välille (liite 23). Esimerkiksi objekti1.translateX = objekti2.translateX*2.0; tarkoittaa samaa kuin matematiikassa x = 2y. Aina kun objekti1 liikkuu yhden mittayksikön verran X-akselilla, objekti2 liikkuu kaksi kertaa saman verran X-akselilla. (Autodesk 2016o.)



Kuvio 32. Esimerkki yksinkertaisesta ekspressiosta.

Ekspressiot mahdollistavat komplekseja riippuvuuksia objektien välille. Ne tukevat ehtoja ja vaativampia matemaattisia metodeja. Esimerkiksi jos halutaan toisen objektin seuraavan toista objektia epälineaarisesti ja rajoittaa liikettä siten, että tietyn pisteen jälkeen seuraaminen loppuu, tarvitsemme ensiksi epälineaarisen matemaattisen lausekkeen ja ehdon samaan ekspressioon.

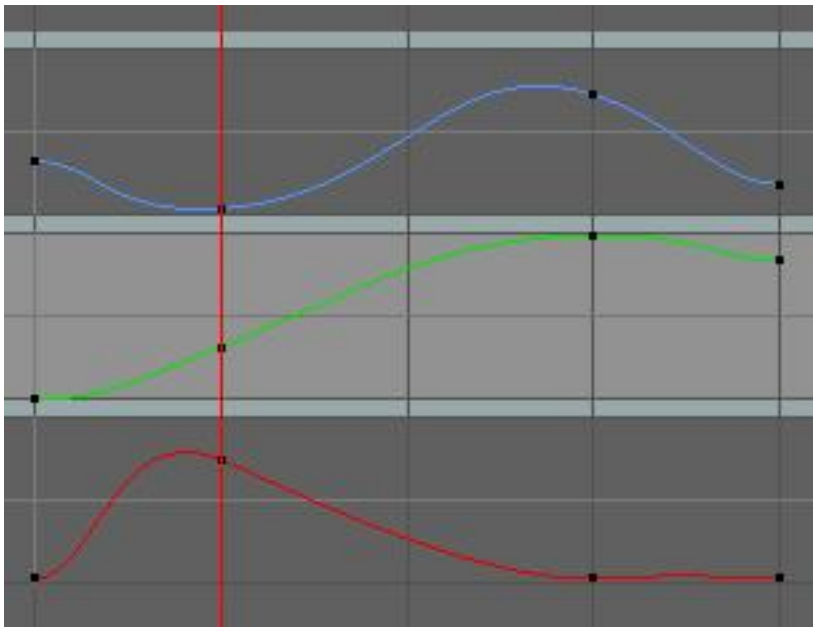
```
foot_ctl.softMinDist = foot_ctl.chainLength*foot_ctl.softness*0.1;  //softness minimun distance
foot_ctl.softDist = foot_ctl.chainLength-foot_ctl.softMinDist;       //softness distance

if(R_leg_distCtlToRoot.distance >= foot_ctl.softMinDist && foot_ctl.softDist != 0)
{
    foot_ctl.softFactor = foot_ctl.softDist*(1-exp((-(R_leg_distCtlToRoot.distance-foot_ctl.softMinDist))/foot_ctl.softDist))+foot_ctl.softMinDist;

    R_leg_stretchBlender.color2R = foot_ctl.softFactor;
    R_leg_stretchBlender.color2G = 1;
    R_leg_stretchBlender.color1R = R_leg_distCtlToRoot.distance;
    R_leg_stretchBlender.color1G = R_leg_distCtlToRoot.distance/foot_ctl.softFactor;
}
else
{
    R_leg_stretchBlender.color1R = R_leg_distCtlToRoot.distance;
    R_leg_stretchBlender.color1G = 1;
    R_leg_stretchBlender.color2R = R_leg_distCtlToRoot.distance;
    R_leg_stretchBlender.color2G = 1;
}
```

Kuvio 33. Esimerkki kompleksista ekspressiosta.

Tämä ekspressio antaa soft-IK-ominaisuuden IK-rigille. Kerron siitä rigien sovelluksissa lisää (liite 9, liite 41). Jos tämän haluisi luoda utility-nodejen avulla, se vaatisi turhan laajan nodeverkoston, jota olisi hankala tarvittaessa muokata. Ekspressioiden heikkoutena kuitenkin on, että ne evaluoidaan aina, vaikka attribuuteissa ei tapahtuisi muutosta, toisin kuin nodepuulla rakennetut järjestelmät. Suurimmissa osissa tapauksista tällä ei kuitenkaan ole merkitystä, koska nämä ekspressiot automatisoivat rigien toimintoja, joita animaatiot tarvitsevat lähes jatkuvasti. Toisin sanoen animaatiot muuttavat attribuutteja, jotka kutsuvat uudelleen evaluoinnin jatkuvasti joka tapauksessa.

3.6 Set driven key

Set driven key on käytännössä sama asia kuin animaatio, paitsi se käyttää jotain muuta attribuuttia ohjaajana kuin aikaa (liite 39). Tätä käytetään esimerkiksi korjaus-blend-shapejen ohjaajana tai minkä tahansa muun attribuutin, jonka liike olisi muuten liian monimutkaista saada tarkaksi ekspressioilla tai nodepuulla. (Autodesk 2016p.)



Kuvio 34. Esimerkki set driven keyllä tehdystä animaatiokäyrästä.

3.7 Dependency graph (DG)

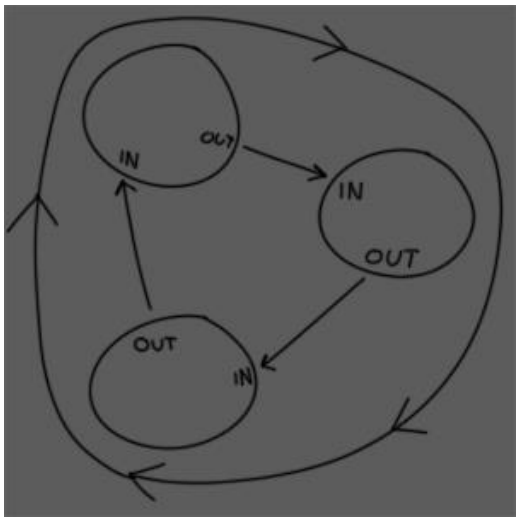
Dependency graph tai DG on ketju nodejen kytköksiä, jotka määräävät, missä järjestyksessä attribuutit muuttuvat skenessä. Jokaisella nodella on inputeja ja outputeja. output-arvot riippuvat täysin mitä arvoja inputeihin syötetään ja mitä nodet matemaattisesti tekevät näille arvoille. Sana ”dependency” tulee tästä riippuvaisuussuhteesta ja ”graph” kokonaisesta nodejen inputien, outputien ja niiden kytköksien verkostosta. (Autodesk 2016q.)

Maya on optimoitu laskemaan arvoja vain silloin kun tarvitaan. Kun jonkun noden input-arvo muuttuu, saman noden output merkataan likaiseksi (dirty) ja kaikki tähän outputtiin kytketyt nodet merkataan myös likaisiksi. Tämä jatkuu, kunnes koko nodepuu on käyty läpi.
Kun Maya kutsuu uudelleen evaluoinnin, esimerkiksi skenen piirron yhteydessä, se katsoo, onko noden output merkattu likaiseksi. Jos on, kyseiselle nodelle kutsutaan uudelleen evaluointi. Samalla katsotaan, onko tällä uudelleen evaluoitavalla nodella likaisia inputeja. Jos on, sama tehdään kaikille likaisille inputeille, kunnes koko nodepuu on evaluoitu ja merkattu puhtaaksi. (Autodesk 2016r.)

Joskus monimutkaisissa rigeissä voi syntyä dependency-luuppi. Maya tunnistaa tämän ja hälyttää, että sykli (cycle) tapahtuu joidenkin nodejen kytköksissä eikä evaluointi välttämättä toimi niin kuin pitäisi (liite 21).

Kuvio 35. Esimerkki cycle-hälytyksestä.

Tämä johtuu siitä, että jonkin noden output riippuu samassa puussa seuraavan noden inputista, jolloin evaluointi alkaa kiertää samaan tapaan kuin mikrofonin kaappaama ääni kaiuttimen vieressä. Maya onneksi tunnistaa tämän ja varoittaa eikä kaada koko ohjelmaa. Joskus dependency-luuppien syitä on vaikea tunnistaa. Usein paras tapa on avata nodeeditori ja yrittää etsiä mahdollisia syitä tai rakentaa kyseinen järjestelmä toisella tavalla.



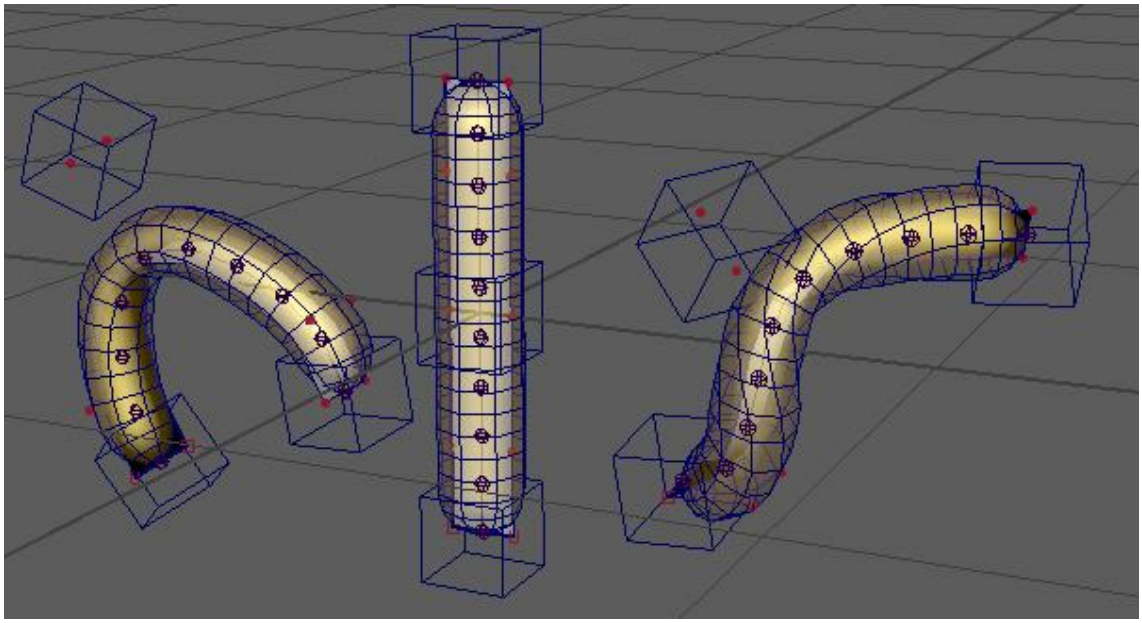
Kuvio 36. Havainnollistus dependency-luupista

4 Rigisovellukset

Tässä luvussa käsittelen rigisovelluksia, joita yllä mainituilla sisäänrakennetuilla työkaluilla voidaan saada aikaiseksi. Suurin osa näistä on suhteellisen yleisiä systeemeitä, joita kuitenkaan ei lähes missään ohjelmassa ole sisäänrakennettuna. Osista näistä on hyvin vaikea löytää tietoa, ja sekin tieto on usein hyvin suppeaa tai maksumuurin takana.

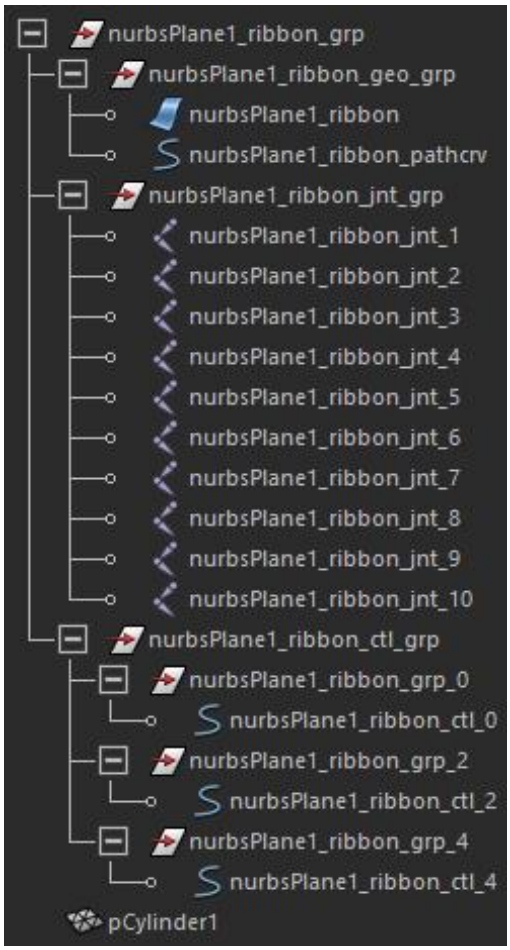
4.1 Ribbon-spine

Ribbon-spine on spline-IK:n tapainen rigi, jossa jointit seuraavat käyrän sijasta pitkän muotoista nurbs-pintaa (Martin 2010). Liitteenä video ja skenetiedosto (liite 8, liite 38)



Kuvio 37. Ribbon-spine esimerkki.

Koska nurbs-pinta on moniulotteisempi kuin käyrä, sillä on mahdollista saada tarkempi twistin kontrollointi jointeille. Jointtien twistaus voidaan laskea nurbs-pinnan normaalista. Ribbon-spinellä on muitakin etuja, esimerkiksi tarkempi kontrolloitimahdollisuus jointti-ketjun pituudelle eikä jointtien tarvitse olla samassa hierarkiassa.

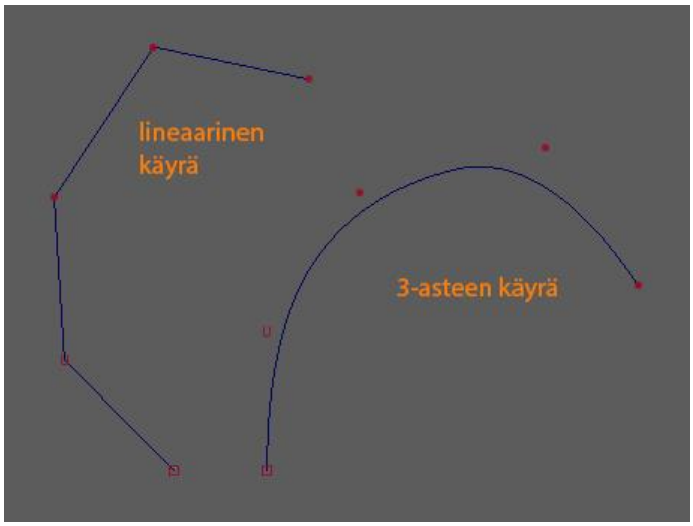


Kuvio 38. Ribbon-spinen hierarkia

Jointtien skaalan voi laskea yksilöllisesti, ja paikan määrittämiseen nurbs-pinnan pinnalla on enemmän mahdollisuuksia. Jos jointtiketju halutaan pysymään ehdottomasti saman pituisena, silloin spline IK on kuitenkin paras vaihtoehto.

4.1.1 Ribbon eli nurbs-pinta

Ribbon on yksinkertaisesti nurbs-pinta, jota kutsutaan ribboniksi vain tämän kaltaisen rigin yhteydessä. Nurbs-pinta muodostuu U- ja V-suuntaisista nurbs-käyristä, joiden väliin lasketaan matemaattisesti pinta (Autodesk 2016b). Näille käyrille voi määritellä aste-arvon (degree), joka määrittää niiden matemaattisen kompleksisuuden. 1-aste tarkoittaa, että käyrä on lineaarinen, eli käyrä kulkee suorana viivana kontrollipisteiden välillä.



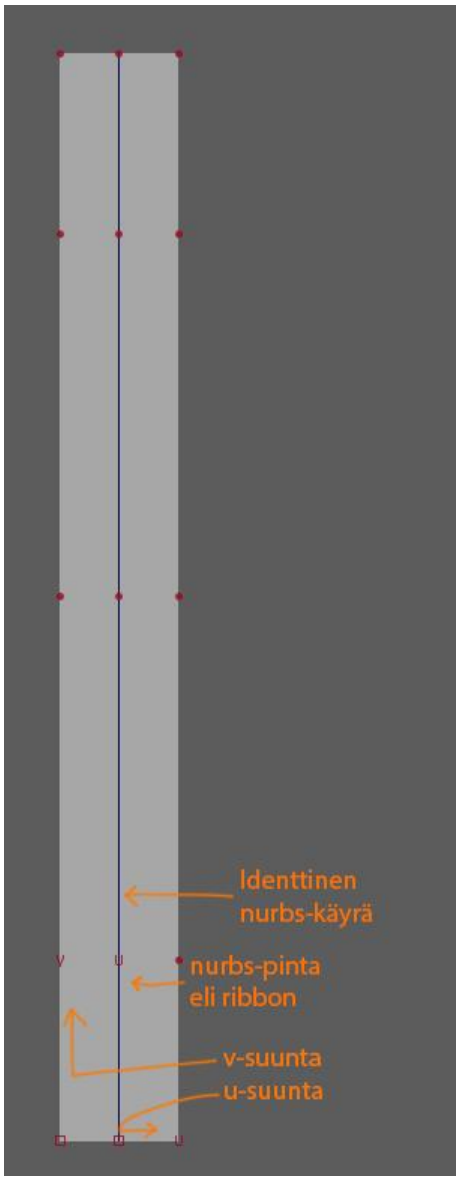
Kuvio 39. Käyrän asteet

Ribbon-spineen tarvitaan nurbs-pinta, joka on 1-asteinen U-suuntaan ja 2- tai 3-asteinen V-suuntaan.

4.1.2 Jointtien kytkentä ribboniin

Monet lähteet neuvovat jointtien kytkennän ribboniin Mayan hiussimulaatioon tarkoitetuilla follicle-nodeilla. Itse en suosittele tätä metodia, koska sillä saa aikaan turhan paljon ylimääräisiä nodeja, kompleksisuutta ja se on myös hieman rajoittuneempi kuin itse kehittämäni järjestelmä.

Oma metodini käyttää ribbonia vain jointtien twistin laskemiseen sen pinnan normaaleista ja jointit kytketään motion-path-nodella ribbonin keskellä kulkevaan käyrään. Tämän käyrän täytyy olla täysin samanasteinen ja sisältää yhtä monta kontrollipistettä, jotta se saadaan muokkautumaan täysin identtisesti ribbon-pinnan kanssa.

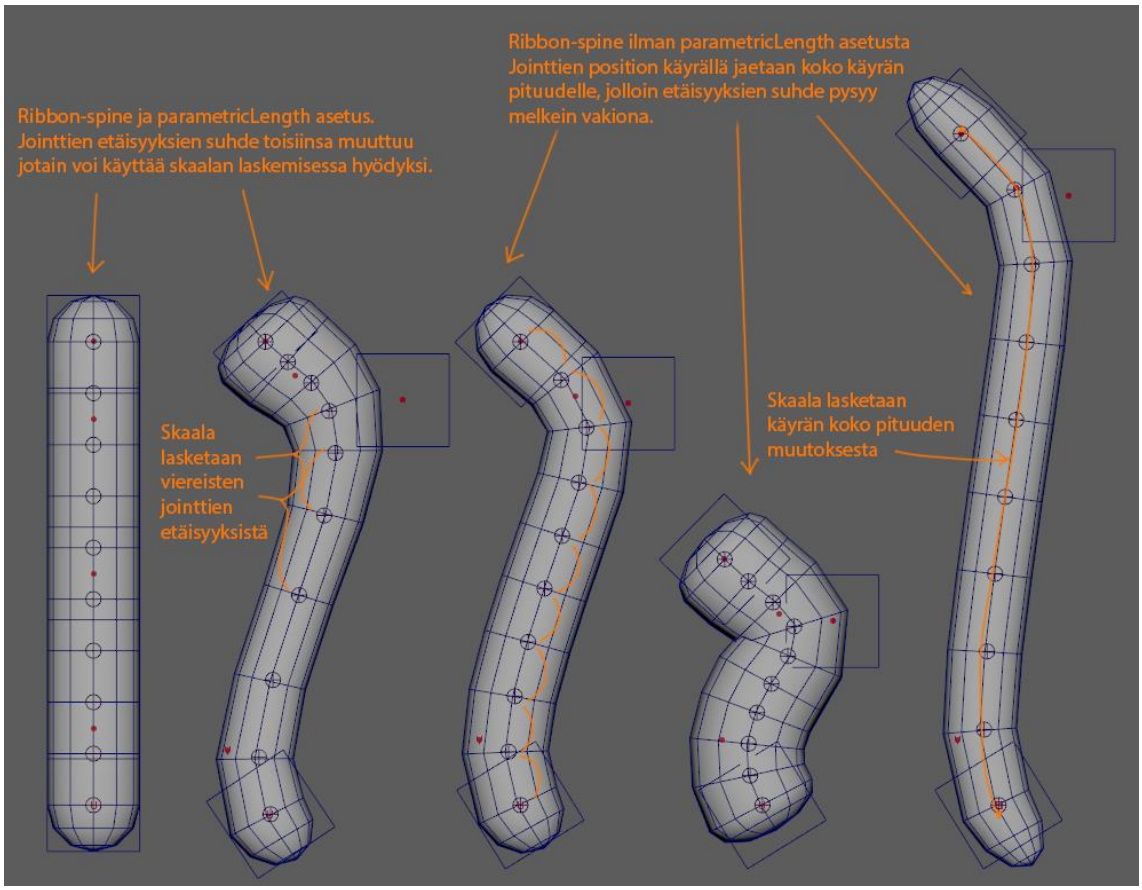


Kuvio 40. Nurbs-pinta (ribbon) ja sen keskellä identtinen käyrä

Jointtien twistin saa laskettua ribbonista helposti käyttämällä normal-constrainttia. Normal-constraintilla kytketty objekti seuraa geometrian pintaa käyttäjän määritellyllä akselilla. Constrainttiin voi myös asettaa up-vektorin samalla tavalla kuin esimerkiksi aim-constrainttiin. Ribbon-spinen tapauksessa motion-path-node huolehtii up-vektorista. Kannattaa huomioida, että normal-constraintin aim-akseli ja motion-path-noden front-akseli käyttävät eri akseleita. Itse tavallisesti laitan normal-constraintille aim-akseliksi Y-akselin ja motion pathille front-akseliksi X-akselin, koska näiden akselien käyttö kyseisissä tarkoituksissa on yleisesti standardisoitunut. Motion-path-noden uValue määrittää jointtien position käyrällä.

4.1.3 Parametric-length-skaalauksen rakennus

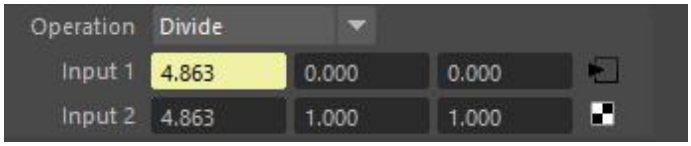
Ribbon-spine toimii erinomaisesti varsinkin rigeissä, joissa tarvitaan venymisominaisuutta. Ribbon-spinelle, jossa on käytetty motionPath-nodea, voidaan käyttää kahta erilaista metodia, jotka riippuvat siitä, onko motionPath-noden parametricLength-attribuutti päällä vai ei. Kummallakin asetuksella on omat etunsa (liite 7, 32, 33).



Kuvio 41. Ribbon spline parametricLength-asetuksella ja ilman.

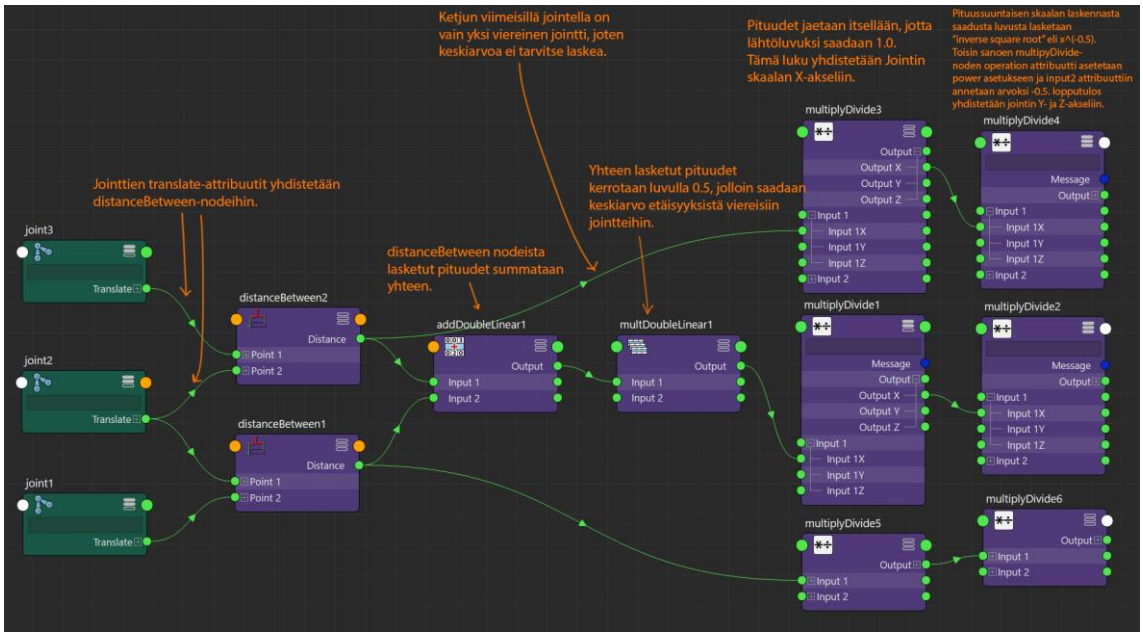
Jos parametricLength on päällä, kannattaa jointtien skaala laskea jokaisen jointin etäisyydestä viereisiin jointteihin. Tällöin skaalaa kontrolloidaan paikallisesti eri kohdissa käyrän pituutta. Tämä myös vaatii huomattavasti suuremman määrän nodeja, joka myös syö enemmän tietokoneen laskentatehoa.

Ensimmäisenä lasketaan jokaiselle jointille etäisyydet viereisestä jointista yhdistämällä translate-arvot distanceBetween-nodeen. Tämän jälkeen etäisyyksistä lasketaan keskiarvo laskemalla viereiset etäisyydet yhteen ja kertomalla luku 0,5-luvulla, joka tarkoittaa samaa kuin luku jaettaisiin kahdella. Tähän voi käyttää addDoubleLinear- ja multDoubleLinear-nodeja. Keskiarvolaskusta saatu tulos jaetaan itsellään, jotta lähtökohdaksi saadaan 1,0.



Kuvio 42. Etäisyyden keskiarvo jaetaan itsellään multiplyDivide-nodella.

Tässä vaiheessa pituussuuntaisen skaalan laskenta on valmis ja tämä luku voidaan yhdistää jointin scaleX-attribuuttiin. Muut akselit saadaan laskemalla tästä luvusta käänteinen neliöjuuri (Schleifer 2006, 72), eli multiplyDivide-noden operation-atribuutti asetetaan power-asetus, ja input2-atribuuttiin syötetään luku -0,5. Kun objekti venyy, sen muut skaala-akselit muuttuvat siten, että objektin tilavuus säilyisi suurin piirtein samana.



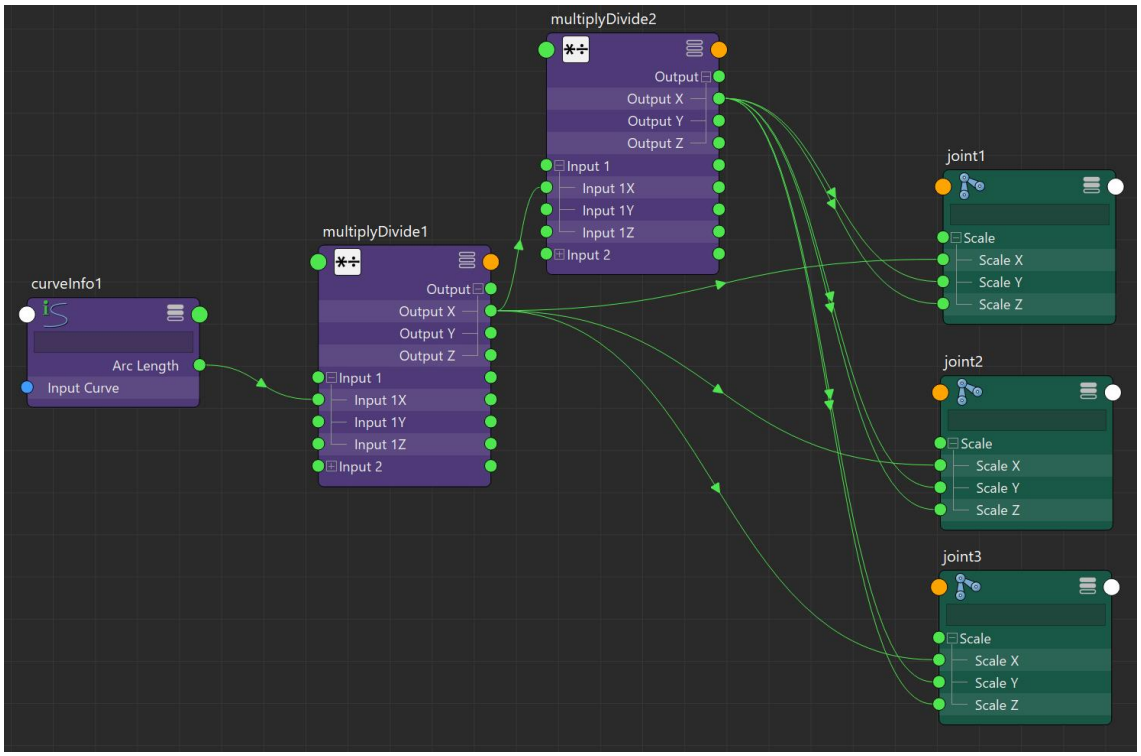
Kuvio 43. parametricLength-skaalalaskurin nodeverkko.

4.1.4 Skaalan laskenta ilman parametricLength-asetusta.

Ilman parametricLength-asetusta jointtien paikat pysyvät vakiona suhteessa käyrän koko pituuteen. Tämä tarkoittaa, että skaalan laskenta viereisten jointtien etäisyyksistä ei kannata. Tässä tapauksessa on helpompi laskea skaala käyrän koko pituuden muutoksesta. Käyrältä ei saa pituusattribuuttia ilman sille tarkoitettua nodea nimeltä curveInfo. Tämä node kannattaa luoda kirjoittamalla komento ”arclen –ch 1” Mayan alaosassa sijaitsevaan komentoriviin eli command line.

Kuvio 44. ”arcLen –ch 1”-komento komentorivissä.

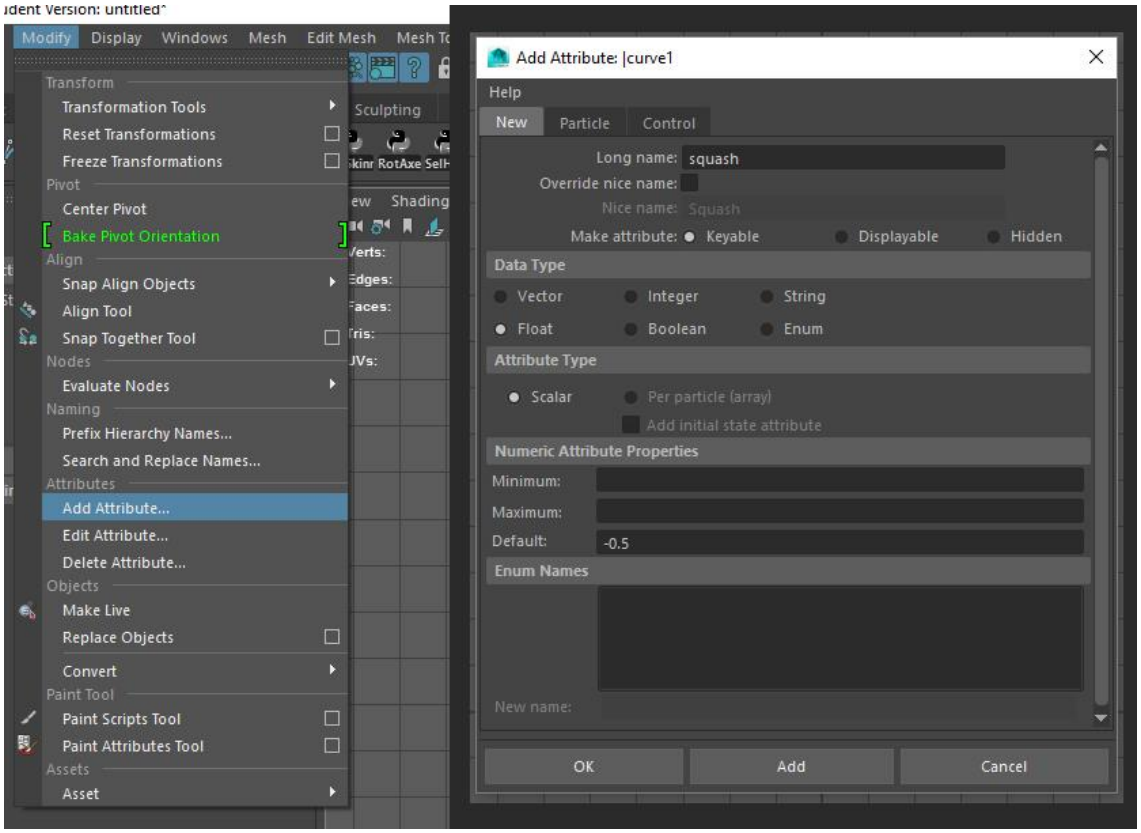
Tässä nodessa on attribuutti nimeltä arcLength, joka kertoo käyrän pituuden. ArcLength-attribuutista saatu pituus jaetaan itsellään multiplyDivide-nodella, jolloin saadaan lähtöluvuksi 1,0. Tämä luku yhdistetään kaikkien ketjun jointtien pituussuuntaiseen skaalaan. Muut akselit saadaan samalla tavalla kuin tehtiin edellisessä skaalan laskennassa, jossa parametricLength oli aktivoituna. Pituussuuntaiselle skaalalle lasketaan käänteinen neliöjuuri multiplyDivide-nodella. Noden operation-atribuutti asetetaan power-asetukseen, ja input2X-attribuutille annetaan arvo -0,5. Tämän jälkeen outputX-attribuutti kytketään jointin scaleY- ja scaleZ-atribuutteihin.



Kuvio 45. Skaalan laskenta käyrän pituudesta.

4.1.5 Kontrolli-attribuutit skaaloille

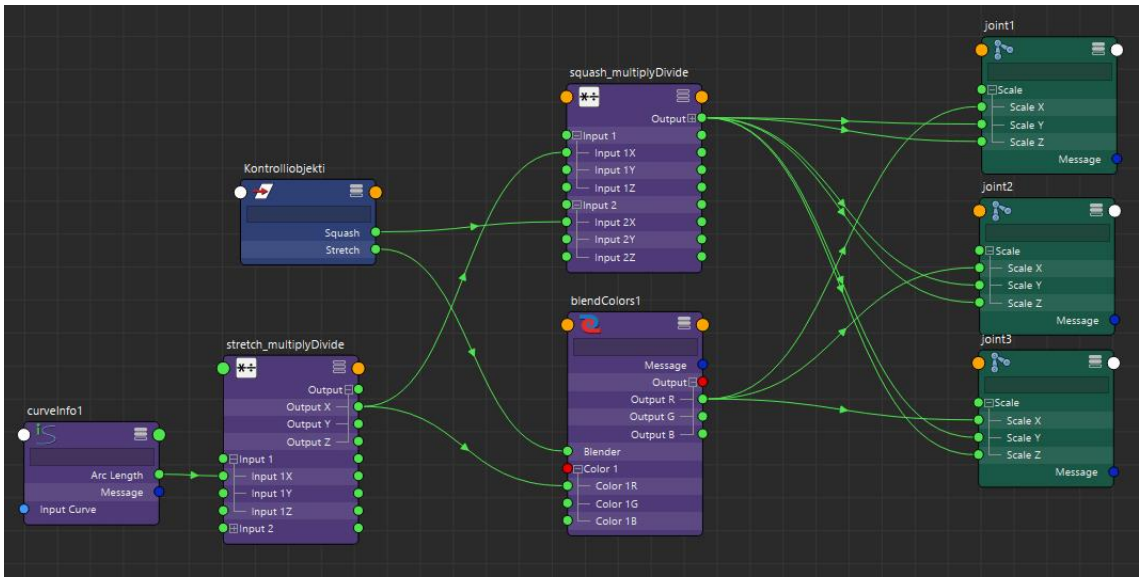
Monessa tilanteessa skaalan automatiikka halutaan pois päältä ja säätää skaaloja manuaalisesti. Kontrolliobjektille voidaan tehä squash- ja stretch-attribuutit. Omia attribuutteja tehdään Mayassa modify-valikosta add attribute-työkalulla.



Kuvio 46. add attribute-valikko

Squash-attribuutille annetaan lähtöarvoksi -0,5. MultiplyDivide-noden input2X-arvo, jolla laskettiin käänteinen neliöjuuri, kytketään kontrolliobjektin squash-attribuuttiin. Tätä attribuuttia säätelemällä pystytään säätämään, kuinka paljon objekti paksuuntuu, kun sen pituus muuttuu.

Stretch-attribuutin kytkentä on hieman monimutkaisempi. Tähän tarvitaan blendColors-nodea, jolla sekoitetaan pituussuuntaista skaala-arvoa 1,0-luvun välillä. Kontrolliobjektin stretch-attribuutti kytketään blendColors-noden blender-attribuuttiin, MultiplyDivide-noden outputX-attribuutti kytketään blendColors-noden input1R-attribuuttiin ja input2R-attribuutille annetaan arvo 1,0.



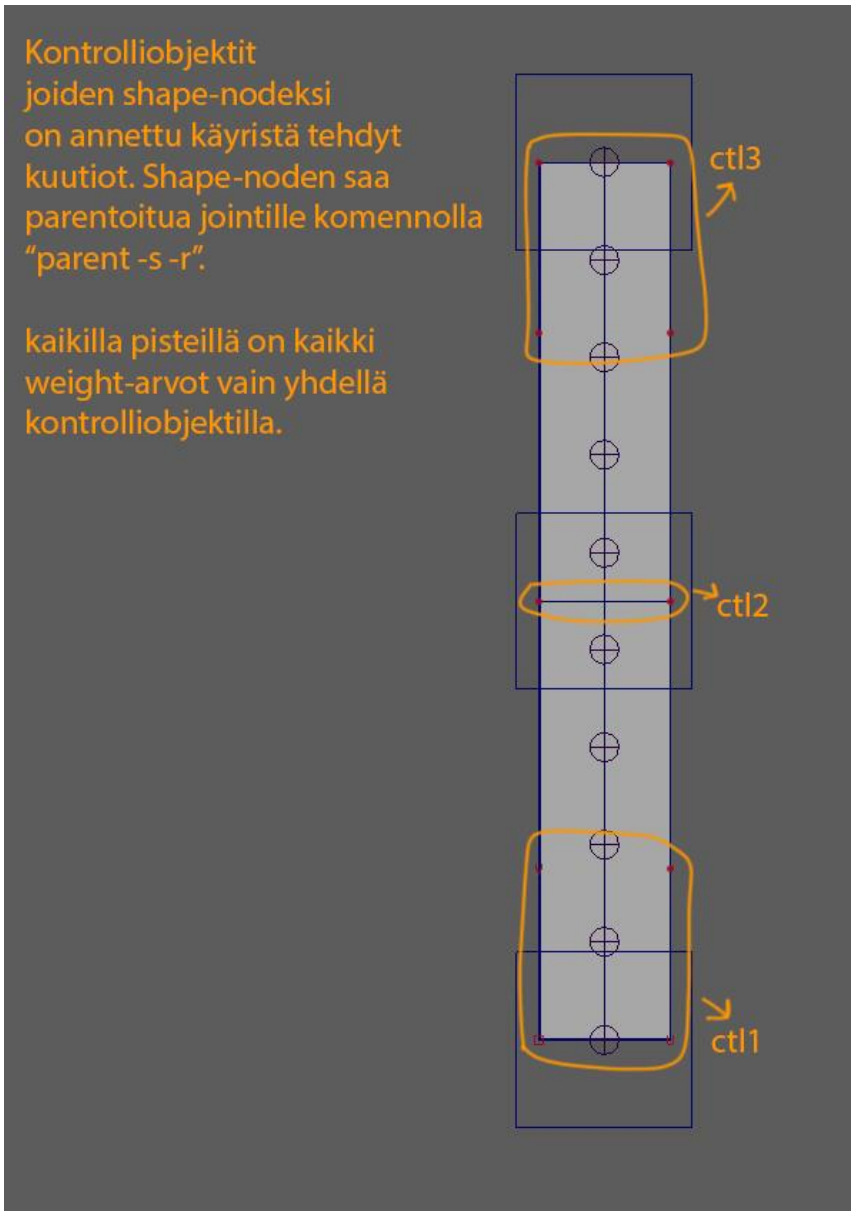
Kuvio 47. Kontrollit venymiselle ja paksuuntumiselle.

Jos squash-attribuutin arvo pienenee, paksuuntumisefekti kasvaa. Kun arvo on nollassa, silloin paksuuntumista ei tapahdu ollenkaan. Positiivinen arvo kääntää efektin päinvastaiseksi, jolloin objekti paksuuntuu, kun sitä venytetään pidemmäksi. Stretch-attribuutin arvon ollessa 1,0 automaattinen venyminen on käytössä. Jos arvo on 0,0 jointti saa aina pituussuuntaisen skaalan arvoksi 1,0.

Tämä skaalakontrolli kannattaa rakentaa kaikille jointeille erikseen tarkoittaen sitä, että jokaista jointtia kohden tarvitaan omat squash- ja stretch-attribuutit ja blendColors-nodet.

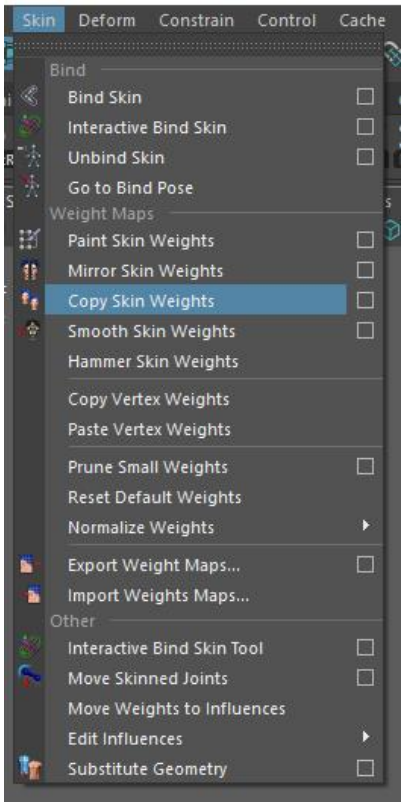
4.1.6 Kontrolliobjektien rakennus

Kun jointeille on asetettu constraintit, on aika tehdä kontrolliobjektit itse ribbon-pinnalle ja käyrille. Käyrän ja pinnan saa kytkettyä kontrolliobjekteihin vähimmällä vaivalla käyttämällä smooth-skinnausta. Smooth-skinnausta käytettäessä joudutaan käyttämään jointteja kontrolliobjekteina. Jointille kannattaa tehdä parent-objektiksi transform-objekti, jonka avulla translate-atribuutit saadaan nollattua. Kontrolliobjektit kannattaa liikuttaa ja rotatoida kohdalle liikuttamalla transform-objektia.



Kuvio 48. Esimerkki kontrolliobjekteista.

Ensimmäiseksi kannattaa bindata pelkkä ribbon-pinta ja maalata siihen weightit. Sen jälkeen käyrään voidaan projisoida weightit käyttämällä copy skin weights-työkalua, jotta weightit ovat varmasti identtiset. Näin käyrä pysyy täysin ribbon-pinta keskellä, kun kontrolliobjekteja liikutetaan.



Kuvio 49. Copy skin weights-työkalu löytyy skin weights-valikosta.

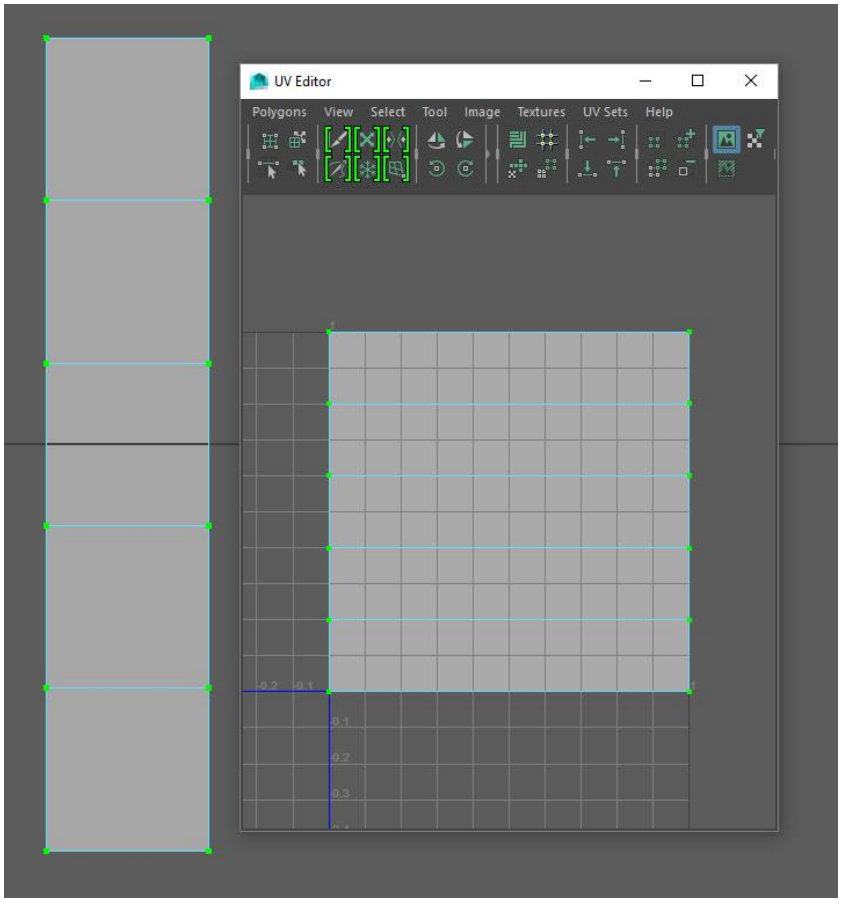
Kontrolliobjektien parent-objektina olevan transform-objektin voi parentoida tai kytkeä constraintilla mihin tahansa varsinaiseen rigiin ja ribbon-spine seuraa mukana. Myös ribboniin kytketyt jointit voidaan aika vapaasti siirtää sisälle hierarkioihin. Kaikkia mahdollisuuksia en ole tutkinut.

Ribbon-spinestä olen ohjelmoinut skriptin, joka rakentaa koko rigin alusta loppuun automaattisesti. Se sisältää samat ominaisuudet kuin tässä käsitellyssä ribbon-spinessä. Skripti on mukana tämän opinnäytetyön liitteenä (liite 12). Mukana on myös skripti, jolla voi nollata jointtien skaalat, jos ribbon-spineä on muokattu rakentamisen jälkeen (liite 13). Ribbon-spinestä on myös toinen versio, joka käyttää kahta käyrää nurbs-pinnan sijasta. Tähän on myös skripti liitteenä (liite 11).

4.1.7 Polygonipinnalle rakennettu ribbon-spine

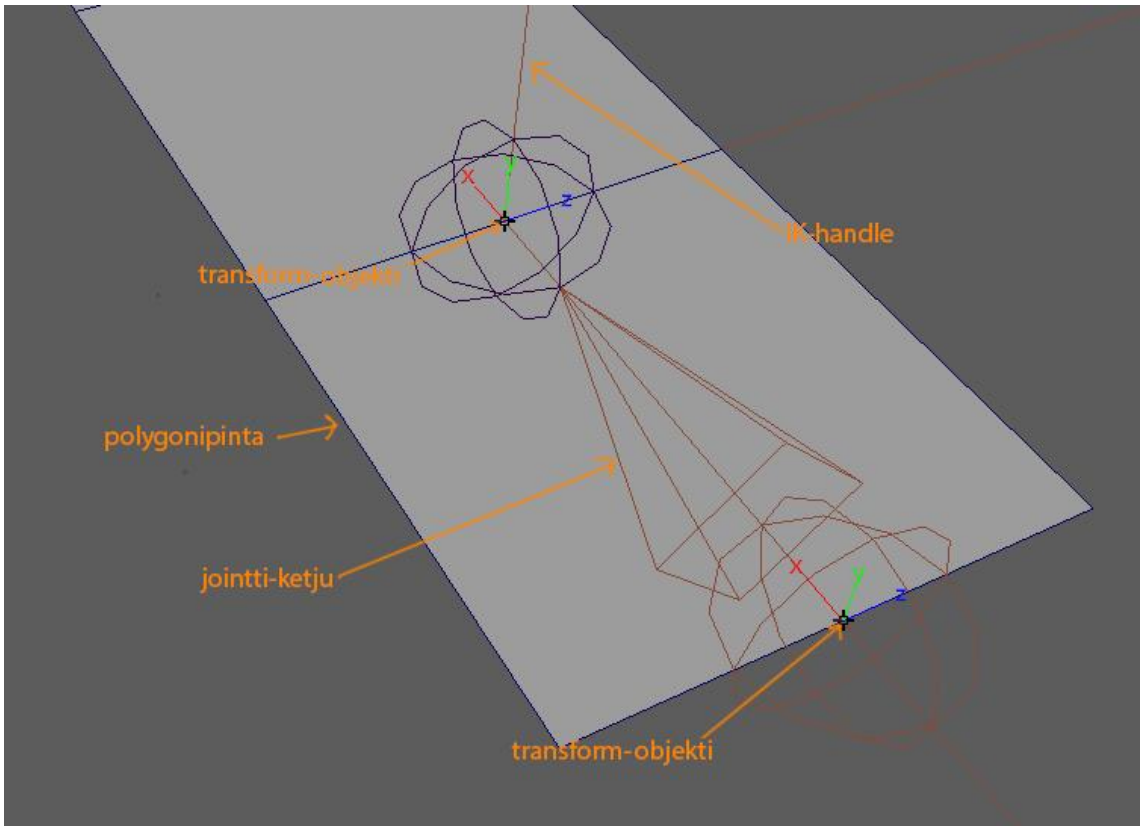
Ribbon-spinen voi rakentaa myös polygonigeometriaan (liite 37). Tässä metodissa transform-objekteja kytketään polygonipinnalle pointOnPoly-constraintilla. Transform-objektit seuraavat polygonipintaa sen UV-kartan avulla (Autodesk, 2016s). Polygonipinnan UV:t

kannattaa järjestää kuvanmukaisesti niin, että UV-koordinaatit täyttävät koko UV-kartan tilan (Kuva 50).



Kuvio 50. Polygonipinnan UV:t

Sen jälkeen tehdään haluttu määrä kahden jointin ketjuja, jotka ohjataan single-chain-IK:lla. Single-chain-IK:n ensimmäinen jointti kiinnitetään point-constraintilla polygonipintaan kiinnitiettyyn transform-objektiin, ja IK-handle-objektin voi parentoida seuraavaan polygonipintaan kiinnitettyyn transform-objektiin kuvan 51 mukaisesti.



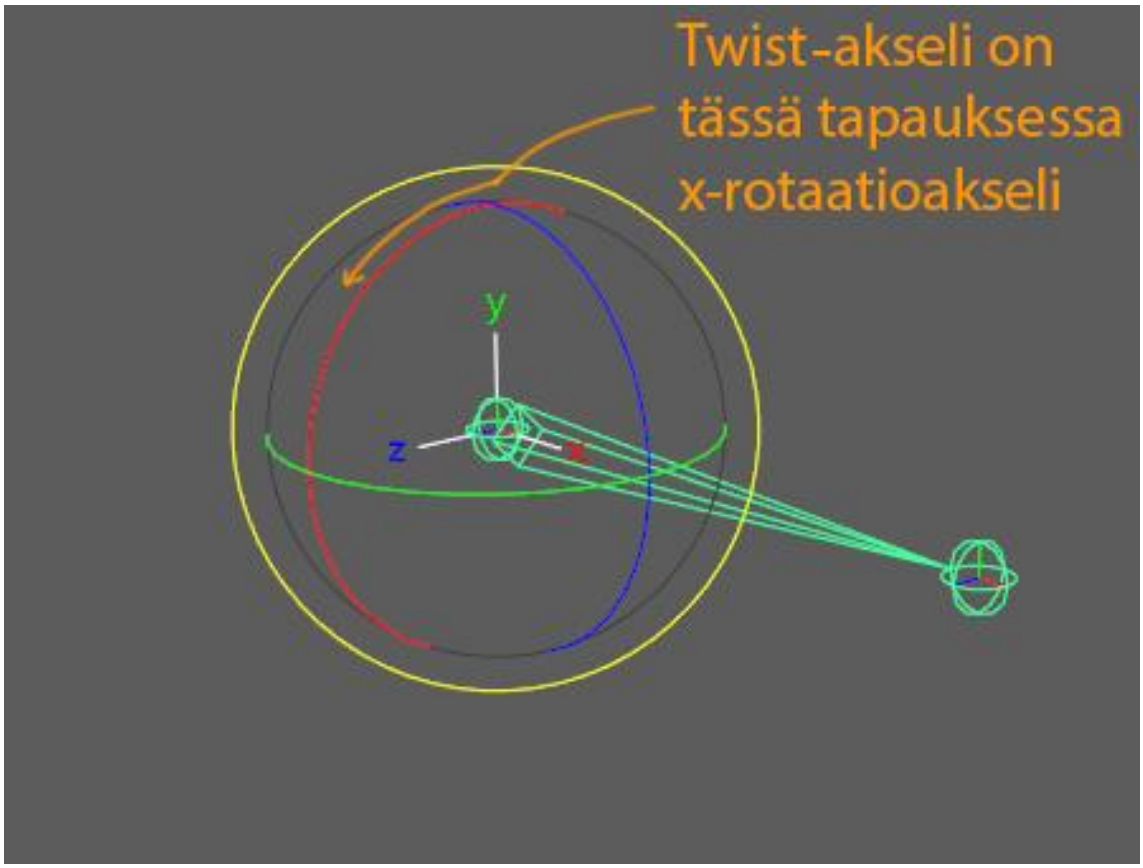
Kuvio 51. single-chain-IK-ketjun kiinnitys polygonipinnalle

Tämän jälkeen IK-handlen rotaatiot kannattaa kiinnittää edelliseen transform-objektiin Orient-constraintilla, jotta jointin twisti saadaan myötäilemään polygonipintaa oikeassa kohdassa.

Tämän jälkeen jointtien venymisen kontrollointi voidaan rakentaa samalla tavalla kuin nurbs-pinta käyttävän ribbon-spinen, jossa käytetään parametricLength-asetusta motionPath-nodeissa.

4.2 Twist-joint

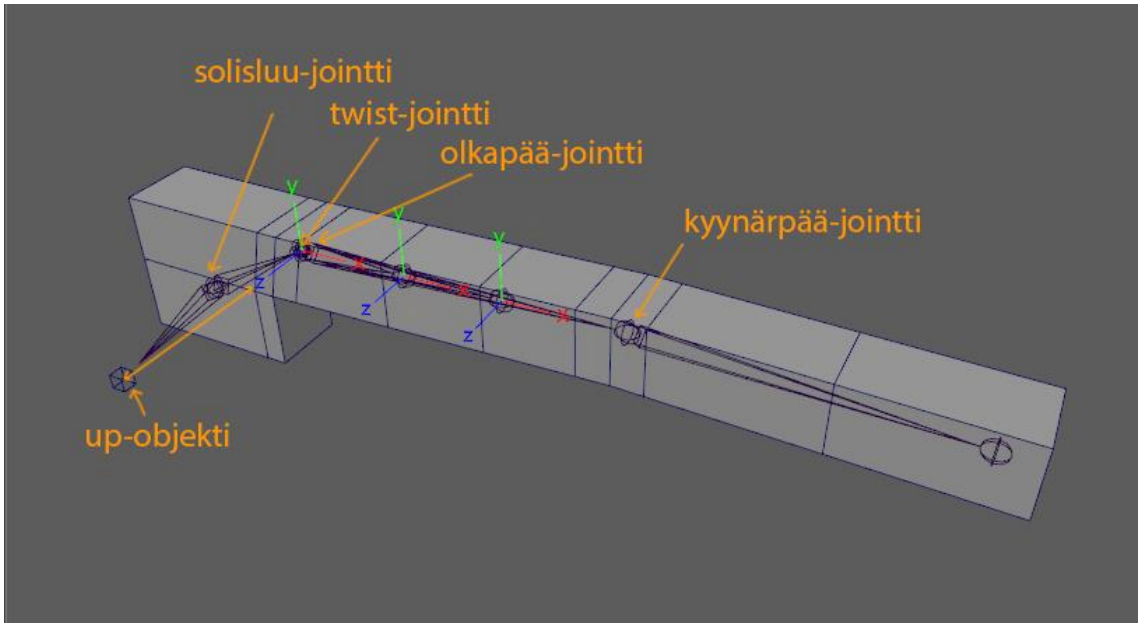
Twist jointeilla pystytään korjaamaan muokkausvirheitä, joita syntyy hyvin laajoista nivelten liikkeista (Amin 2014). Esimerkiksi olkapään, ranteen ja reisiluun liikeradat ovat yleisesti isoimmat ongelmat. Otan tähän esimerkiksi olkapään, koska sen liikerata on kaikkein laajin, joka tekee siitä myös kaikkein hankalimman tapauksen (liite 6, liite 44). Tässä tapauksessa twistillä tarkoitetaan kuvan 52 mukaista akselia.



Kuvio 52. Twist-akseli on jointtien x-rotaatioakseli.

4.2.1 Twist-kontrolli aim-constraintilla

Jotta olkapää saataisiin muokkautumaan oikein, olkapäässä ei saa tapahtua twistausta eli X-akselilla rotaatiota. Tähän tarvitaan ylimääräinen jointti, jonka twistaus on poistettu aim-constraintin avulla. Twist-jointti tähtää kyynärpääjointtiin ja up-objektiksi tehdään transform-objekti, joka parentoidaan solisluujointtiin. Up-objekti kannattaa asettaa täydellisen tarkasti olkapääjointin eteen siten, että olkapääjointin z-akseli osoittaa suoraan up-objektia. Twist-jointti kannattaa parentoida olkapääjointtiin ja nollata kaikki translaatiot, rotaatiot ja orientaatiot ennen constraint-kytkentää. Tällä tavalla aim-constraintin ja jointin arvot pysyvät nollassa constraint-kytkennän jälkeen.



Kuvio 53. Olkapään jointit

4.2.2 Twist-kontrolli single-chain-IK:lla

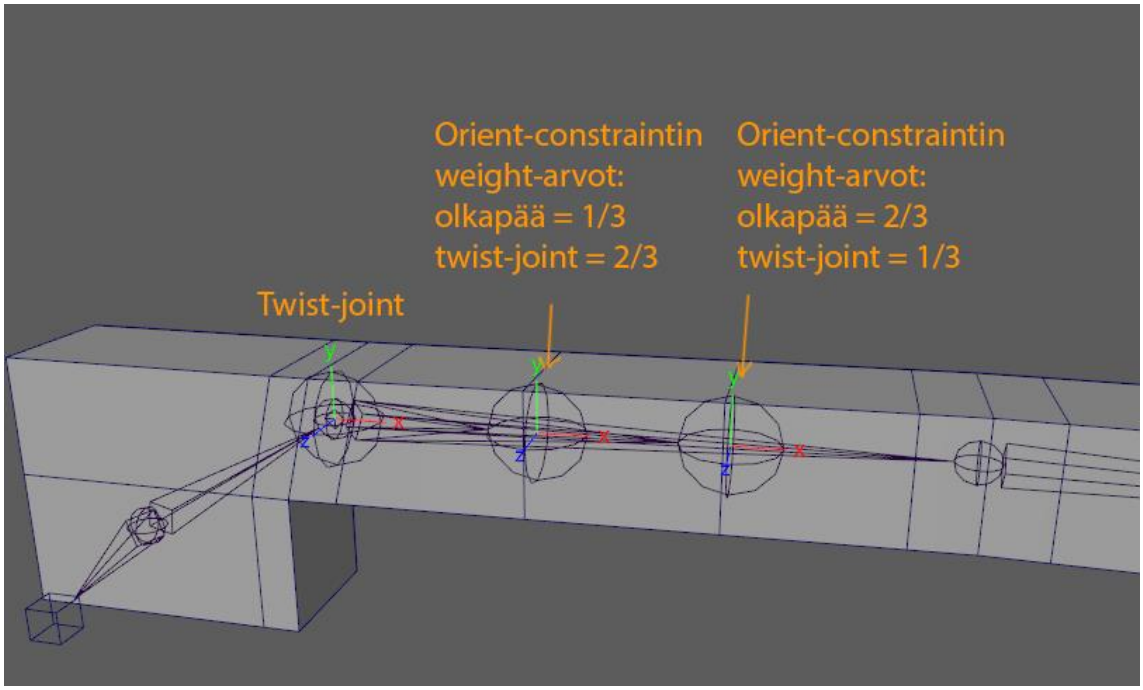
Twist-kontrollit voidaan tehdä myös single-chain-IK:n avulla, joka osoittautui itseasiassa vielä paremmaksi metodiksi kuin aim-constraintilla rakennettu järjestelmä (Liite 40). Tämä on yksinkertaisempi rakentaa ja mahdollistaa laajemman liikeradan ilman up-vektorin manuaalista säätämistä. Aim-constraintilla rakennetussa twist-kontrollissa saattaa kuitenkin olla joissain tapauksissa hyötyä, jos jointtien twistiä halutaan kontrolloida käsin up-vektorin avulla.

Twist-jointista tehdään kahden jointin ketju, jolle tehdään single-chain-IK. Tämä jointtiketju voidaan parentoida suoraan olkapääjointtiin ja IK-handle kytketään kyynärpäänjointtiin point-constraintilla. Sen jälkeen IK-handle kytketään orient-constraintilla solisluuhun. Orient-constraintin interpType-attribuuttiin kannattaa asettaa varmuuden vuoksi shortest-arvo.

4.2.3 Ylimääräiset korjausjointit

Käsivarren muokkausta on vielä lisää paranneltu lisäämällä kaksi ylimääräistä twist-jointtia, jotka ovat kytketty orient-constraintilla kumpaankin olkapääjointtiin ja ensimmäiseen

twist-jointtiin. Orient-constraintista on säädetty weight-attribuutit siten, että ensimmäisessä twist-jointti saa 2/3 arvon ja olkapää-jointti saa 1/3 arvon. Toinen twist-jointin orient-constraint saa weight-arvot twist-jointtiin 1/3 ja olkapää jointtiin 2/3. Orient-constraintille asetetaan interpolointi tyypiksi shortest. Tämä tarkoittaa quaternion-interpolointia, joka laskee lyhimmän reitin tavoitekulmaan. Tämä toimii suurimmissa osissa tapauksista erinomaisesti.

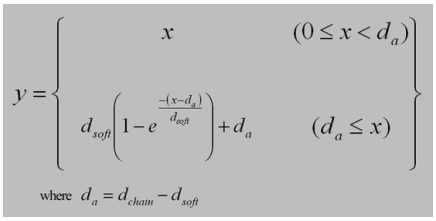


Kuvio 54. Orient constrainttien arvot

Olkapään twistin korjaus on valmis. Tämä on hyvä lähtökohta paremmalle muokkaukselle. Jos tämän lisäksi tarvitsee vielä lisää korjausta, tämän järjestelmän päälle voi rakentaa blendshape-korjailua tai lisätä jointteihin perustuvia korjausmekanismeja.

4.3 Soft-IK

Kun IK-ketjun kääntää suoraksi, jointit naksahtavat suoraksi kiihtyvällä nopeudella. Animoidessa esimerkiksi jalkoja polvien naksahtelujen korjaaminen saattaa aiheuttaa huomattavasti lisätyötä ilman jonkinlaista automaatiota. Tähän on kehitetty matemaattinen kaava, jolla jointit saadaan kääntymään pehmeästi suoraksi automaattisesti. Sitä kutsutaan soft-IK:ksi (Nicholas 2006).



Kuvio 55. Soft-IK:ssa käytetty kaava (Nicholas 2006)

Tästä kaavasta voidaan tehdä ekspressio, jolla IK-handle-objekti saadaan jäämään hieman jälkeen tietystä pisteestä eteenpäin, joka estää IK-jointtien kiihtyvän suoristumisen. Tätä samaa kaavaa voidaan käyttää myös jointtien skaalaan perustuvaan korjaukseen. Näitä kahta metodia voidaan sekoittaa blendColors-nodella keskenään (liite 41).

Päätin tehdä koko järjestelmän yhteen ekspressioon, koska nodeista rakennettuna nodeverkosto olisi liian hankalasti muokattava ja sekava.

```
foot_ctl.softMinDist = foot_ctl.chainLength*foot_ctl.softness*0.1;  //softness minimun distance
foot_ctl.softDist = foot_ctl.chainLength-foot_ctl.softMinDist;       //softness distance

if(R_leg_distCtlToRoot.distance >= foot_ctl.softMinDist && foot_ctl.softDist != 0)
{
    foot_ctl.softFactor = foot_ctl.softDist*(1-exp((-(R_leg_distCtlToRoot.distance-foot_ctl.softMinDist))/foot_ctl.softDist))+foot_ctl.softMinDist;

    R_leg_stretchBlender.color2R = foot_ctl.softFactor;
    R_leg_stretchBlender.color2G = 1;
    R_leg_stretchBlender.color1R = R_leg_distCtlToRoot.distance;
    R_leg_stretchBlender.color1G = R_leg_distCtlToRoot.distance/foot_ctl.softFactor;
}
else
{
    R_leg_stretchBlender.color1R = R_leg_distCtlToRoot.distance;
    R_leg_stretchBlender.color1G = 1;
    R_leg_stretchBlender.color2R = R_leg_distCtlToRoot.distance;
    R_leg_stretchBlender.color2G = 1;
}
```

Kuvio 56. Soft-IK-ekspressio

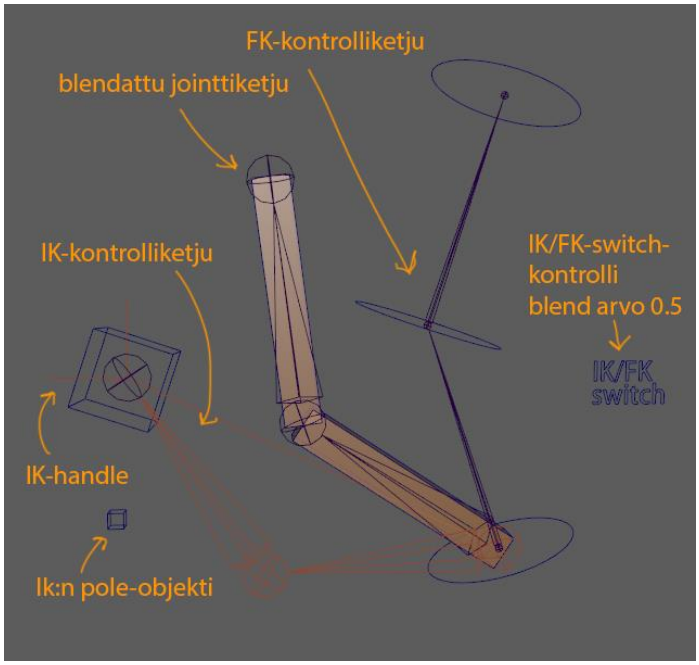
Kontrolliobjektilla on attribuutti nimeltä softness, jonka arvoa voi säätää 0,0-10,0:n välillä. Ekspressio käyttää tätä attribuuttia kertoimena, jolla saadaan arvoja jointtiketjun pituuden ja nollan väliltä. Jos softness-attribuutin arvo on 0,0, IK-handle alkaa jäädä jälkeen heti jointtiketjun juuresta eteenpäin. Jos arvo on 10,0, soft-IK on käytännössä poissa käytöstä.

Kontrolliobjektilla on toinenkin attribuutti nimeltä stretchy, jolla sekoitetaan skaalautuvan soft-IK:n välillä. Skaalaan perustuvassa metodissa käytetään täysin samaa kaavaa, paitsi IK-handlen position manipuloimisen sijasta manipuloidaan jointtien skaalaa.

Olen ohjelmoinut Python-skriptin, joka rakentaa koko soft-IK-järjestelmän automaattisesti valmiille IK-rigille. Kyseinen skripti on liitteenä mukana (liite 14).

4.4 IK/FK blend

Lähes kaikista hyvin varustelluista rigeistä löytyy IK/FK-blend ominaisuus (Baskin 2014). Mayan IK-solverista löytyy sisäänrakennettu ikBlend-attribuutti, mutta sen toiminta on hyvin rajottunutta. Sen sijaan kannattaa tehdä IK- ja FK-jointtiketjut erikseen. Tämä rigi koostuu loppujen lopuksi kolmesta jointtiketjusta: IK-ketju, FK-ketju sekä varsinainen jointti-ketju, joka seuraa kumpaakin jointtiketjua. Seuraamiskohdetta voidaan vaihtaa pehmeästi ik_fk_blend-attribuutilla IK/FK-switch-kontrolliobjektista (liite 10, liite 25).



Kuvio 57. Kuva rigistä, jossa on IK/FK-blend ominaisuus. (liite 25)

Esimerkkikuvassa (kuva 57) keskimmäisen jointtiketjun jointit on kytketty orient- ja scale-constraintilla kumpaankin, FK- ja IK-jointti-ketjuun. Orient constraintin interpType-attribuutti on muutettu asetukseen. Tämä tarkoittaa, että orient-constraint käyttää quaternion-interpolointia ja valitsee lyhimmän mahdollisen kulman. Tämä myös tarkoittaa, että jos kulmien ero muuttuu suuremmaksi kuin 180 astetta, interpolointialgoritmi löytää pienemmän asteen toista reittiä ja rotaatio hyppää 180-astetta vastakkaiseen suuntaan. Tässä tapauksessa virheen voi korjata välttämällä lähellä 180 asteen kulmaeroja, silloin kun IK/FK-blend arvo on muuta kuin 0,0 tai 1,0.

Koska constrainteilla on tässä tapauksessa kaksi kohdetta, IK-jointit ja FK-jointit, niistä myös löytyy kaksi weight-arvoa kumpaakin kohdetta varten. Constraintien weight-arvot ovat suhdelukuja. Jos kummatkin arvot ovat 1,0, niin objekti seuraa kumpaakin kohdetta yhtä suurella painolla. Jos toisella on weight arvo 0,0 ja toisella 1,0, niin toista kohdetta seurataan täysin.

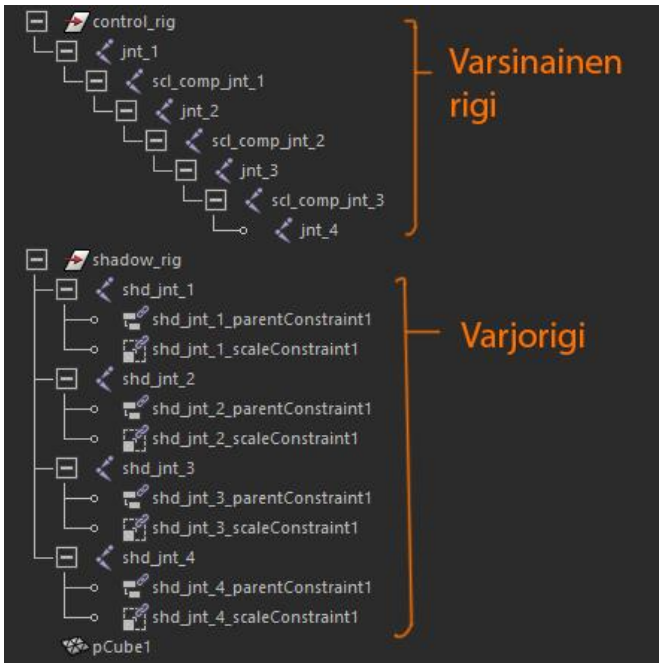
Tämä saadaan helposti aikaiseksi reverse-nodella. IK/FK-kontrollin ik_fk_blend-attribuutti kytketään reverse-noden inputX-attribuuttiin, ja reverse-noden outputX-attribuutti kytketään FK-weight-attribuuttiin. Sen jälkeen IK/FK-kontrollin ik_fk_blend-attribuutti kytketään suoraan IK-weight-attribuuttiin. Tällöin kummatkin constraintien kohteet saavat vastakkaiset weight-arvot muuttamalla IK/FK-kontrollin ik_fk_blend-attribuuttia.

Tähän voi myös käyttää blendColors-nodea suoraan, ilman constrainteja. rotaatiot ja skaalat kytketään suoraan blendColors-noden input-attribuutteihin ja ik_fk_blend-attribuutti blendColors-noden blender-attribuuttiin.

IK/FK-blend-kontrollien lisäksi usein rigien mukana tulee skripti, jolla IK:n ja FK:n välillä pystytään vaihtamaan saumattomasti. Tällaista skriptiä kutsutaan IK/FK-switchiksi.

4.5 Shadow-rig eli varjorigi

Varjorigiksi kutsutaan jointtiketjua, joka on yhteensopiva pelimoottorien kanssa ja on täysin erillään varsinaisesta kontrollirigistä. Useimmiten tämä tarkoittaa hierarkiaa, josta on riisuttu kaikki ylimääräiset objektit kuten IK-effektorit ja transform-objektit. Jos varjorigin jointit kytketään varsinaiseen rigiin pose- ja scale-constrainteilla, niin varjorigin hierarkian ei tarvitse olla identtinen. Hierarkia voi olla myös rakenteeltaan täysin erilainen (Liite 45).



Kuvio 58. varjorigin hierarkia

Jotkut pelimoottorit tukevat constrainteja itse pelimoottorin sisällä, jolloin varjorigi voi olla huomattavasti kehittyneempi. Animaatiodataakin syntyy huomattavasti vähemmän, kun osa jointeista liikkuu constrainttien avulla eikä liikettä tarvitse lukea animaatiokäyristä. Tämä myös mahdollistaa automaattisten korjausmuokkaajien toiminnan, vaikka pelihahmoa animoitaisiin proseduraalisesti pelimoottorissa esimerkiksi ragdollfysiikalla tai pelimoottorin sisäisillä IK-solvereilla.

4.6 Autorigaajat ja plug-init

Mayalle ja muille ohjelmille on nykypäivänä paljon erilaisia autorigaussovelluksia ja skriptejä, jotka voivat olla hyödyllisia kiireellisissä projekteissa. Autorigaajilla pystytään rakentamaan kokonaisia hahmorigejä alusta loppuun täysin valmiiksi animoitavaksi. Nämä rigit usein sisältävät suurimman osan yleisistästä ominaisuuksista, kuten soft-IK, IK/FK-blend ja twist-jointit. Jokaisessa autorigaajassa on kuitenkin omat puutteensa, jolloin omien rigien rakentelutaidosta on paljon hyötyä. Toinen autorigaajien mahdollinen ongelma on yhteensopivuus pelimoottorien kanssa ja eri peliprojektit saattavat vaatia hyvin spesialisoituja rigijärjestelmiä, varsinkin jos pelimoottori tukee constrainteja ja muita automatisaatioita. Automaattisesti rakennettuja rigejä voi olla myös erittäin vaikea muokata, varsinkin kun käyttäjä ei ole rakentanut niitä itse. Autorigaaja saattavat myös käyttää erillisiä plug-ineja, jotka ovat riippuvaisia ohjelman versiosta ja vaativat ylläpitoa,

jotta ne varmasti toimivat seuraavissakin versioissa. Muiden tekemiä rigejä kannattaa kuitenkin tutkia. Se saattaa avata uusia näkökulmia jotka saattavat auttaa ongelmien ratkaisuissa.

5 Pohdintaa ja yhteenveto

Erilaisia tapoja rigata on monia, ja jokainen rigaaja oppii oman tapansa ratkaista ongelmia. Rigaajana ei kannata yrittää välttää teknisiä ongelmia vaan yrittää ratkaista niitä. Yksi ratkaistu ongelma voi olla osaratkaisu jo monelle tulevalle ongelmalle, mikä tarkoittaa mahdollisesti paljon suurempaa harppausta kehittymisessä rigaajana.

Tähän opinnäytetyöhön ei valitettavasti mahtunut yhtä tärkeimmistä rigaukseen liittyvistä aiheista, skriptien ohjelmointi. Skriptaus on nykypäivänä lähes välttämätön työkalu rigaajalle. Monimutkainen rigi, joka ei toimi kuin yhdessä tilanteessa, on erittäin kallis mille tahansa tuotannolle. Lyhytelokuvissa, joissa ei välttämättä tarvita kuin pari tai yksi hahmo, pystytään välttämään skriptaus, mutta niissäkin tapauksissa siitä on suuri hyöty. Esimerkiksi jos yhden hahmon rigissä tarvittaisiin useassa tilanteessa ribbon-spineä, sen rakentaminen useaan kertaan alusta loppuun manuaalisesti veisi paljon aikaa. Tämän takia melkein mikä tahansa uudelleen käytettävä järjestelmä kannattaa miettiä siitä näkökulmasta, miten sen voisi hyötykäyttää jatkossa automatisoimalla sen uudelleen rakennus skriptien avulla.

Mielestäni kuitenkin pääsin tavoitteeseen tässä opinnäytetyössä siinä mielessä, että sain käsiteltyä hyvin kattavasti yleisimmät rigauksessa käytetyt Mayan työkalut. Onnistuin mielestäni suhteellisen selkeästi selvittämään rigisovelluksien toiminnan ja miten niitä voidaan rakentaa. Ehkä isoimpia ongelmia oli tarpeeksi laaja pohjustus rigauksen peruskäsitteistä, jotta aloittelevat rigaajat ymmärtäisivät asian mahdollisimman kivuttomasti.

Lähteet

Autodesk, 2016a. Nodes and attributes, Autodesk Maya 2016 Help. http://help.autodesk.com/view/MAYAUL/2016/ENU/?guid=GUID-D53B9E3D-E6E3-4CC3-A38F-3AA3A09205E5 luettu: 3.4.2016

Autodesk, 2016b. About NURBS: Uses and Limitations, Autodesk Maya 2016 Help. http://help.autodesk.com/view/MAYAUL/2016/ENU/?guid=GUID-51096BC4-32B7-4391-BE39-21641B374745 luettu: 3.4.2016

Autodesk, 2016c. Transforming objects, Autodesk Maya 2016 Help. http://help.autodesk.com/view/MAYAUL/2016/ENU/?guid=GUID-59D26516-9183-4154-A5D9-1A581275D06A luettu: 6.4.2016

Autodesk, 2016d. Joints and bones, Autodesk Maya 2016 Help. http://help.autodesk.com/view/MAYAUL/2016/ENU/?guid=GUID-1B59334F-2605-44C3-B584-A55B239A2CBE luettu: 6.4.2016

Autodesk, 2016e. Animated rotation in Maya, Autodesk Maya 2016 Help. http://help.autodesk.com/view/MAYAUL/2016/ENU/?guid=GUID-CBD30A0A-1166-4076-A564-1ADC946A15F3 luettu: 6.4.2016

Autodesk, 2016f. Understanding the attributes of a joint, Autodesk Maya 2016 Help. http://help.autodesk.com/view/MAYAUL/2016/ENU/?guid=GUID-5A8BB621-96E0-4CE8-9792-085EC8F07164

Autodesk, 2016g. Turning Off Segment Scale Compensate in Maya, Autodesk Maya 2016 Help. http://help.autodesk.com/view/MAYAUL/2016/ENU/?caas=caas/simplecontent/content/turning-segment-scale-compensate-maya-how-to-make-maya-rigs-play-nice-unity.html luettu: 6.4.2016

Autodesk, 2016h. Skinning your character, Autodesk Maya 2016 Help. http://help.autodesk.com/view/MAYAUL/2016/ENU/?guid=GUID-EFE68C08-9ADA-4355-8203-5D1D109DCC82 luettu: 6.4.2016

Autodesk, 2016i. Blend Shape deformer, Autodesk Maya 2016 Help. http://help.autodesk.com/view/MAYAUL/2016/ENU/?guid=GUID-B8853C3F-2997-4DC2-95A0-7C43E45888E4 luettu: 6.4.2016

Autodesk, 2016j. Constraints, Autodesk Maya 2016 Help. http://help.autodesk.com/view/MAYAUL/2016/ENU/?guid=GUID-7665A291-FAA7-44C0-BDEB-A6C83482116C luettu: 2.4.2016

Autodesk, 2016k. Inverse Kinematics (IK), Autodesk Maya 2016 Help. http://help.autodesk.com/view/MAYAUL/2016/ENU/?guid=GUID-07C3BA47-32BB-477B-B6C5-1090E5C9B81C luettu: 6.4.2016

Autodesk, 2016l. Spline IK solver, Autodesk Maya 2016 Help. http://help.autodesk.com/view/MAYAUL/2016/ENU/?guid=GUID-B4EF8784-92D1-4D83-9CA5-A692D06607B8 luettu: 6.4.2016

Autodesk, 2016m. Path Animation, Autodesk Maya 2016 Help.

http://help.autodesk.com/view/MAYAUL/2016/ENU/?guid=GUID-341D458E-7324-4538-A736-72DD9A58542E luettu: 6.4.2016

Autodesk, 2016n. Utility nodes, Autodesk Maya 2016 Help. http://help.autodesk.com/view/MAYAUL/2016/ENU/?guid=GUID-DA9707D2-8A0D-4911-A010-8274C57D3FD3 luettu: 2.4.2016

Autodesk, 2016o. Animation expressions, Autodesk Maya 2016 Help http://help.autodesk.com/view/MAYAUL/2016/ENU/?guid=GUID-5F75516C-403A-4CC3-A118-AEDEFDE970B4 luettu: 2.4.2016

Autodesk, 2016p. Driven keys, Autodesk Maya 2016 Help. http://help.autodesk.com/view/MAYAUL/2016/ENU/?guid=GUID-2C048635-CDD2-4CF7-820D-A032204C8CE8 luettu: 2.4.2016

Autodesk, 2016q. Dependency graph, Autodesk Maya 2016 Help. http://help.autodesk.com/view/MAYAUL/2016/ENU/?guid=GUID-51096BC4-32B7-4391-BE39-21641B374745 luettu: 3.4.2016

Autodesk, 2016r. Dependency Graph (DG) nodes, Autodesk Maya 2016 Help. http://help.autodesk.com/view/MAYAUL/2016/ENU/?guid=__files_Dependency_graph_plugins_Dependency_Graph_DG_nodes_htm luettu 2.5.2016

Autodesk, 2016s. Mapping UVs, Autodesk Maya 2016 Help. http://help.autodesk.com/view/MAYAUL/2016/ENU/?guid=GUID-FDCD0C68-2496-4405-A785-3AA93E9A3B25 luettu: 24.4.2016

Amin, Jahirul 2014. Introduction to rigging in Maya - The shoulder and the arms, 3D-Total. http://www.3dtotal.com/tutorial/1846-introduction-to-rigging-in-maya-the-shoulder-and-arms-by-jahirul-amin-animation-body-neck luettu: 16.2.2016

ARLab. Baking Animation in Maya, linkAr technical documentation http://arlab.com/doc/arlab/product/3d_engine/exporters/maya/advanced/baking_animations Luettu 24.4.2016

Baskin, Jason 2014. Tutorial: Creating an FK/IK arm setup in Maya. https://www.youtube.com/watch?v=M6ViCN_sPVE luettu: 6.4.2016

Martin, Jose-Antonio 2010. Ribbon Spine Rig, CreativeCrash. https://www.creativecrash.com/maya/tutorials/character/c/ribbon-spine-rig luettu: 6.4.2016

Nicholas, Andy 2006. Soft IK in XSI, Softimage Blog. http://www.softimageblog.com/archives/108 luettu: 27.3.2016

Schleifer, Jason 2006. Instructor Notes – Animator Friendly Rigging Part II, Autodesk Master Class

Wikipedia 2016. Gimbal lock. https://en.wikipedia.org/wiki/Gimbal_lock luettu: 6.4.2016

Liitteet

Videot:

Liite 1. Gimbal.avi
Liite 2. SplineIk.avi
Liite 3. SetDrivenKey.avi
Liite 4. IK.avi
Liite 5. BlendShape.avi
Liite 6. ShoulderTwist.avi
Liite 7. ParametricLengthRibbon.avi
Liite 8. RibbonSpine.avi
Liite 9. SoftIK.avi
Liite 10. IKFKBlend.avi

Skriptit:

Adobe Acrobat Reader ei hyväksy py-päätteisten tiedostojen avaamista PDF-tiedostoista, joten tämä täytyy tehdä jollain toisella ohjelmalla, kuten esim. Mozilla Firefoxilla.

Liite 11. makeCrvRibbon_v09.py
Liite 12. makeRibbon_v075.py
Liite 13. ribbonResetStretch.py
Liite 14. stretchySoftIk.py

Maya skene-tiedostot:

Liite 15. advanced_twist_control.mb
Liite 16. aim_constraint.mb
Liite 17. aim_constraint2.mb
Liite 18. align_ rotation_axes.mb
Liite 19. angleBetweenExample.mb
Liite 20. blendshape.mb
Liite 21. dependencyLoop.mb
Liite 22. distanceBetweenExample.mb
Liite 23. expression_example.mb
Liite 24. Gimbal.mb
Liite 25. ik_fk_blend_example.mb
Liite 26. IK_problem.mb
Liite 27. inverse_kinematics.mb
Liite 28. invert_scale_solution.mb
Liite 29. manual_segment_scale_compensation.mb
Liite 30. normalConstraint.mb
Liite 31. orient_constraint.mb
Liite 32. parametricLengthComparison.mb
Liite 33. parametricLengthRibbonAnimation.mb
Liite 34. parametricMotionPath.mb

Liite 35. point_constraint.mb
Liite 36. pole_vector.mb
Liite 37. polyRibbonExample.mb
Liite 38. ribbonSpine.mb
Liite 39. setDrivenKey.mb
Liite 40. single_chain_twist_joint_example.mb
Liite 41. softIK_example.mb
Liite 42. splineIK_with_advanced_twist_control.mb
Liite 43. tason_suuntaiset_rotaatioakselit.mb
Liite 44. twist_joint_example.mb
Liite 45. shadowRigExample.mb