Osaamista
ja oivallusta
tulevaisuuden
tekemiseen

Elina Sievänen

Johdanto modulaariseen rigaukseen

3D-hahmojen kehittäminen osissa

Metropolia Ammattikorkeakoulu

Medianomi

Viestinnän koulutusohjelma

Opinnäytetyö

29.5.2020

Tekijä(t) Otsikko Sivumäärä Aika	Elina Sievänen Johdanto modulaariseen rigaukseen 38 sivua + 3 liitettä 29.5.20208.5 2020
Tutkinto	Medianomi
Tutkinto-ohjelma	Viestinnän tutkinto-ohjelma
Suuntautumisvaihtoehto	3D-animaatio ja -visualisointi
Ohjaaja(t)	Lehtori Jaro Lehtonen

Tämä opinnäytetyö käsittelee hahmojen rigausta modulaarisesti 3D-viihdeteollisuudessa. Rigauksella tarkoitetaan 3D-hahmon valmistelua animoitavaksi ja tässä kontektissa modulaarisuus tarkoittaa, että tuloksena syntynyt animaatioranka, eli rigi, sisältää vähintään yhden itsenäisen ja uudelleenkäytettävän osan.
Modulaarisuuden perimmäinen idea on tehostaa tuotantoa kopioimalla ja kierrättämällä moduuleja sekä mahdollistaa hahmomoduulien kehittämisen ja ylläpidon jakamisen useammalle tekijälle. Lisäksi modulaarisuus on yleensä osa suurempaa oman työn automatisoinnin jatkumoa. Tehokkaimmillaan moduulit ovat koodipohjaisia, jolloin moduuleihin pohjautuvat hahmot voidaan päivittää ja uudelleen rakentaa helposti. Ideaalitilanteessa syntyy uusia standardeja ja edistyneempiä rigejä, sillä aikaa käytetään enemmän manuaalisen toiston sijaan uusien ominaisuuksien kehittämiseen.
Opinnäytteen päätavoite oli tutkia missä kohtaa modulaarisuus alkaa maksaa itseään takaisin. Kaksi havainnollistavaa esimerkkiä ovat: suuri tuotanto, joka sisältää paljon hahmoja ja kokeneitä kehittäjiä, ja toisaalta pieni ja kiireellinen opiskelijatuotanto, jossa tekijöitä on vähän. Todellisuudessa näiden kahden välille mahtuu paljon erilaisia tilanteita ja modulaarisuutta voi lisätä asteittain. Tuotannossa voi esimerkiksi esiintyä aluksi vain yksi käsin siirrettävä prototyyppi, joka kirjoitetaan ajettavaksi skriptiksi vasta seuraavien tuotantojen aikana. On myös huomioitava, että onnistuneet modulaariset rigit pohjautuvat onnistuneelle esituotannolle ja huolella suunnitelluille ja tehdyille 3D-malleille.

Opinnäytteen lisäksi toteutettiin modulaarinen Skorpion-rigi, joka pohjautuu mGear-sovellyskehyksen sisältämiin moduuleihin. Tuloksena oli havainto siitä, että vaikka valmiit moduulit nopeuttavat prototyyppien kehittämistä, on omien moduulien kehittäminen yhdelle ihmiselle merkittävä ajallinen investointi. Lisäksi opinnäytteen kirjoittamisen aikaan työkalun ja sen sisältämien moduulien dokumentointi on toistaiseksi puutteellinen erityisesti edistyneiden ominaisuuksien kohdalla.
Lähteenä on käytetty oppikirjoja, verkkoluentoja, artikkeleita, keskustelufoorumeita sekä aiemmin mainittua modulaarista mGear-sovelluskehystä, sen sisältämää Shifter-automaattirigaajaa ja tämän moduuleja.
Tuloksena opinnäytetyö suosittelee lukijalle modulaariseen rigaukseen tutustumista ja tuomista asteittain omaan työnkulkuun saavuttaakseen lisää tehokkuutta ja tietotaitoa pienemmällä tuotannollisella riskillä.

Avainsanat	3D, animaatio, maya, mGear, modulaarinen, rigaus

Author(s) Title Number of Pages Date	Elina Sievänen Introduction to modular rigging 38 pages + 3 appendices 28th of May 2020
Degree	Bachelor of Culture and Arts
Degree Programme	Media
Specialisation option	3D Animation and Visualization
Instructor(s)	Jaro Lehtonen, Senior Lecturer

This thesis is an introduction to modular character rigging in the 3D entertainment industry. That is a process in which a character, is made animatable in a way that the resulted character rig has at least one independent part that can be shared and reused with other characters of a similar kind.
The main reasoning behind modularity is increased development speed achieved by reusing the modules as much as possible and splitting the development of these components among more developers. Commonly modularity is introduced as a part of more broad trend of automating mundane tasks, and is most effective when the components are based on code, that can be updated and re-run easily for each character. In the best-case scenario, this introduces new standards and more advanced rigs, as more time is spent on the development of new features instead of re-implementing similar rigs to a set of characters.

The primary goal for the thesis was to find a tipping point, after which the modularity turns from a burden to being useful. Two opposite cases for this would be a big production with a lot of characters and seasoned staff versus a small and fast production with a few inexperienced students. In reality, there are many cases between the two, and ways of a partial shift to modularity are possible. For example, characters can have only one module in common, that gets developed and copied manually and is developed to be script based in later productions. A note is made on the fact that modular character modeling and rigging are affecting each other and that great modular rigs are based on thorough preproduction and well executed models.
In addition to the thesis, a modular scorpion character rig was developed using modules provided with mGear Shifter, a modular rigging framework, and an autorigger. The experience highlighted the fact that while modules do speed up prototyping, developing them is a significant investment of time for one person, and at the time of writing, many advanced features of the tool itself are undocumented, as it is an ongoing open-source project.

The primary sources for the thesis are educational books, openly shared online lectures, web articles, forums related to the topic, and mGear framework made to facilitate modular rigging.
As a result the thesis recommends readers to search and try modular rigging solutions and advanced rigging him or herself and introduce modular rigging gradually into their workflows to achieve more efficiency and technical skills with less risk.

Keywords	3D, animation, maya, mGear, modular, rigging

Sisällys

Sanastoa

Listaan alla muutamia opinnäytetyölle keskeisiä termejä ja näiden selityksiä.

- Joint,
  rigin osa, joka mahdollistaa geometrian taivuttamisen.

- Vertex, Vertices, Vertiisi, Verteksi,
  pistemäinen geometrian atominen osa, jonka avulla muodostetaan geometrian näkyvät pinnat, keskimäärin kaikki näkyvä koostuu 3D:ssa vertekseistä.

- Rotation, Rotaatio,
  pyörivä liike, kierto

- Translation, translaatio,
  siirtyvä liike.

- Transform, transformaatio,
  yleisnimitys rotaatiolle, skaalaukselle ja translaatiolle

- Deformer,
  yleisnimitys ominaisuudelle, joka saa geometrian pinnan muuttamaan muotoaan.

- FK, Forward Kinematics, suora liikemalli, tapa laskea jointtien transformaatioita hierarkisessa järjestyksessä ylhäältä alaspäin. Esimerkiksi hahmon kyynärpää liikuttaa rannetta.

- IK, Inverse Kinematics, käänteinen liikemalli, tapa laskea transformaatio epähierarkisessa järjestyksessä. Esimerkiksi hahmon ranteen vetäminen liikuttaa kyynärpäätä.

- IK Solver; matemaattinen kaava ja apuväline, joka laskee ja toteuttaa IK-liikkeen.

- Constraint, järjestelmä, jolla rajoitetaan liikettä, esim. aim-constraint pakottaa kappaleen osoittamaan käyttäjän määrittämään pisteeseen.

- Skin Cluster deformer,
  deformer, jonka avulla geometria saadaan seuraamaan jointteja.

- Skinning, weight painting,
  työvaihe, jossa määritetään mitkä jointit liikuttavat mitä verteksejä ja millä vaikutusprosentilla. Yksi rigauksen työvaihe.

- Referencing, referenssointi,
  toiseen ulkopuoliseen tiedostoon viittaaminen tiedoston sisällä. Käytetään mm. tietokoneen levytilan säästämiseksi ja mahdollistaa isojen työkokonaisuuksien jakamisen useammalle henkilölle yhtä aikaa.

(Autodesk 2014; Autodesk 2018; Kilpeläinen 2001; Joensuu 2016; Plularsight 2014.)

Johdanto

Rigaus, englanniksi rigging, on monikerroksellista työtä, jossa yhdistyy parhaimmillaan vahva tekninen osaaminen, anatomian ja materiaalien tuntemus sekä visuaalinen ilmaisu (Assaf 2015, luku 3). 3D-viihdeteollisuudessa rigaus on hyvin yleinen työvaihe, jonka tavoitteena on tehdä 3D-mallista animoitava. Rigaaja luo annetun geometrian pohjalta animaatiorangan eli rigin, jota animaattori käyttää animoidakseen. Rigaukseen erikoistuneesta henkilöstä voidaan käyttää englanninkielisiä nimityksiä ”rigging artist” tai ”character technical director”. Suomalaiseen työkieleen ammattinimikkeeksi on vakiintunut rigaaja, mitä käytän myös tässä opinnäytetyössä.

Mitä realistisemmasta kohteesta on kyse, sitä monimutkaisemmaksi rigaus käy. Rigaajan luoman rigin on tuettava usein erilaisia järjestelmiä, kuten pelimoottoreita ja animaatiokirjastoja, sekä oltava animaattorille helppokäyttöinen. Pohjimmiltaan rigaajan tehtävä on luoda kolmiulotteinen ohjelma, jolle animaattori antaa erilaisia arvoja ja joiden pohjalta syntyy näkyvää liikettä. Saavuttaakseen halutut liikeradat ja muodot on rigaajan ymmärrettävä muun muassa lineaarialgebraa, fysiikkaa, anatomiaa ja visuaalista viestintää. Lisäksi on ymmärrettävä, miten rigaamiseen käytettävä ohjelma toimii ja miten animaattori työskentelee.
Hallitakseen suuremmissa projekteissa helposti syntyvää kaaosta ja välttääkseen inhimillisiä virheitä on rigaajan osattava automatisoida omaa työtään. Tämä johtaa skriptaamiseen eli komentokieliohjelmointiin, jossa ohjelma suoritetaan suoraan tekstistä ilman erillistä kääntämistä konekielelle. 3D-viihdeteollisuudessa käytetyin komentokieli on Python, jonka hyödyllisyyttä myös sivuan (Kasakov 2016, s.15). Lisäksi kerron käytännön työstä hyödyntäen framework-ohjelmia. Framework voitaisiin vapaasti kääntää sovelluskehykseksi, jonka perusidea on tukea ja automatisoida manuaalista työtä. Modulaarisessa rigauksessa rigi luodaan yhden tai useamman itsenäisen osan eli moduulin pohjalta. Tässä opinnäytetyössä käsittelen yksittäisen opiskelijan ja ison tuotannon näkökulmasta modulaarisen rigauksen hyötyjä ja haittoja verrattuna tuotantoihin tai projekteihin, joissa sitä ei hyödynnetä. Pohdin myös, mitä yhdistettävää näissä kahdessa työtavassa voisi olla. Lisäksi esittelen opinnäytteen loppupuolella modulaarista rigausta käytännössä hyödyntäen ilmaista mGear-sovelluskehystä, jonka toimintaa myös hieman avaan. Opinnäytteessäni oletan lukijan jo tuntevan rigauksen työnkulkua ja tavoitteeni on inspiroida lukijaa kokeilemaan skriptausta ja modulaarisuutta myös pienissä tuotannoissa tai vasta harjoitellessaan rigausta.

1 Mitä on modulaarisuus?

Modulaarisuus käsitteenä yhdistää sekä muotoilua että ohjelmistosuunnittelua. Sen perusideana on luoda palasia, joita yhdistelemällä voidaan luoda isoja kokonaisuuksia ja mahdollistaa variaatioita eri kokonaisuuksien välillä helposti ja nopeasti. Modulaarisuuden etuna pidetään erityisesti ajansäästöä, sillä se selkeyttää suunnittelua ja lyhentää tuotantoon käytettyä aikaa. Lisäksi hyviä moduuleja on helppo kierrättää uusissa käyttötarkoituksissa (Spacey 2016.)

Modulaarisesta järjestelmästä voidaan puhua, kun se sisältää osasia, jotka ovat itsenäisiä, mutta toimivat yhdessä. Yksittäinen moduuli voidaan periaatteellisesti määritellä siten, että osat sen sisällä ovat vahvasti sidoksissa keskenään, kun taas sidos moduulin ulkopuolisiin osiin on huomattavasti heikompi. Lisäksi modulaarisuuteen sisältyy ajatus pelkistämisestä piilottamalla tietoa käyttöliittymän tai rajapinnan taakse (Baldwin & White 2000, luku 3.)

Yksinkertainen esimerkki modulaarisesta järjestelmästä voisi olla kaikille tutut Lego-palikat. Jokaisella palikkatyypillä on jokin uniikki ominaisuus ja pinta, jonka avulla ne voidaan yhdistää muiden palikoiden kanssa.

Viereisessä kuvassa Piroshi-nimimerkillä toimiva Lego-taiteilija on yhdistellyt samankaltaisista moduuleista kaksi uniikkia hahmoa, kaksijalkaisen humanoidin ja nelijalkaisen olion (Piroshi 2020). Hahmoista on selkeästi havaittavissa toistuviksi moduuleiksi esimerkiksi raajat, jotka on saatu kiinnittymään erityisellä nivelpalikalla kiinni hahmojen runkoihin.

Kuva 1. Kaksi modulaarista Lego-hahmoa (Piroshi 1.3 2020.)

2 Rigaus ilman modulaarisia osia

2.1 Lähtökohtana pientuotanto

Pienimmissä tuotannoissa sama henkilö voi toimia sekä rigaajana, mallintajana että animaattorina. On selvää, että tällaisessa järjestelyssä yksittäisen rigin kohdalla resurssit ovat rajalliset ja tuloksena on usein hyvin geneerisiä, joskin hyväksi todettuja ratkaisuja.

2.2 Rigin suunnittelu ja työstäminen

Tavallisesti rigin suunnittelussa otetaan huomioon geometria ja animaattorin kanssa sovitut tarpeet. Erityisesti rigattavilla humanoideilla eli ihmismäisillä hahmoilla nämä tarpeet ovat yleensä hyvin samanlaisia, mistä seuraa myös hyvin samankaltaisia rigejä. Joskus ainoastaan raajojen suhde ja esimerkiksi vaatteiden tuomat vaatimukset saattavat vaihdella.

3D-malli tuodaan sisään ja uniikki rigi luodaan suoraan vastaamaan kohteen rakenteita. Tämän jälkeen animaattori testaa rigiä ja palautteen pohjalta rigiä editoidaan lisää. Tyypillisesti parannellaan rigin asettelua ja skinnausta sekä lisätään kontrolleja. Rigauksessa syntyneet innovaatiot tuovat tekijälle lisää kokemusta, mutta jäävät kiireessä helposti yksittäisiksi ratkaisuiksi.

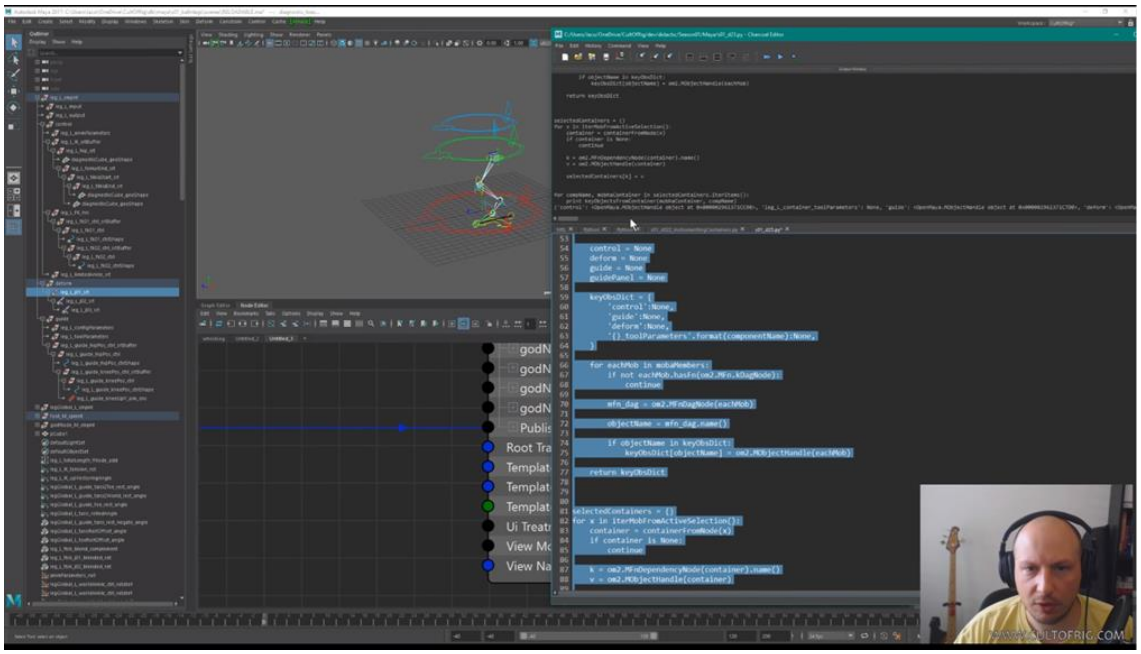
Rigi tallennetaan ja sitä referensoidaan tarvittaessa. Jos tietystä rigistä halutaan tehdä variaatio, rigistä tehdään kopio editoitavaksi. Jos rigaaja tekee innovaation, jonka haluaa lisätä kaikkiin rigeihinsä, on rigien päivitys tehtävä manuaalisesti.

Erityisesti aloittavan rigaajan on vaikea olla tarpeeksi kauaskatseinen välttääkseen usein myöhemmin tuotannossa esiintyvät sudenkuopat. Samasta syystä oman työskentelyn tehostaminenkin on aluksi vaikeaa. Todellisuus kuitenkin paljastuu viimeistään, kun ylläpidettäviä rigejä alkaa olla merkittävä määrä. (Crouzet 2014.)

3 Modulaarinen rigaus ja automatisointi

Valtaosa rigaajan ratkaistavista ongelmista toistuu jatkuvasti, erityisesti kaksijalkaisten kanssa. Tästä syystä modulaarisuus ja oman työnkulun suunnittelu vie pitkälle. Moduuli kannattaa suunnitella siten, että se palvelee tyypillisimpiä ongelmia ja on muokattavissa uusien edistyneempien moduulien pohjaksi. Moduuleja kannattaakin ajatella kerrytettävänä kirjastona, jonka pohjalta voi helposti tehdä huomattavasti siistitympää työtä. (Assaf 2015, luku 6.)

3.1 Omien työkalujen äärelle



Kuva 2. Edistyneen rigaajan työnäkymä. (Fragapane 2018, Cult Of Rig, season 01, Day 23)

Edistyneitä rigaajia yhdistää taito suunnitella työtään etukäteen sekä automatisoida usein toistuvia työvaiheita skripteiksi, kuten kokenut rigaaja Raffaele Fragapane demonstroi ylläolevassa kuvassa (Kuva 2). Voidaankin ajatella, että modulaariset rigit ovat vain jatkoa tälle. Skriptit ovat lyhyitä, yleensä hyvin erityistä käyttötarkoitusta varten nopeasti kirjoitettuja ja suoraan ajettuja ohjelmia, joilla on omat kielensä. Rigauksessa käytetyimmät kielet ovat Python ja MEL, sekä niiden kaksi yhdistelmää, PyMEL ja Maya Commands (Dombrova 2009). Tämän lisäksi edistyneimmät rigaajat käyttävät jonkin verran myös C++:aa (Fragapane 2018).

Kun jokin työvaihe alkaa toistua, voidaan sen pohjalta kehittää skripti ja käyttää sitä aihiona samankaltaisissa tilanteissa. Jos syntynyt skripti osoittautuu erittäin hyödylliseksi, voidaan sille kehitellä käyttöliittymä ja jakaa se myös laajempaan käyttöön. Voidaankin ajatella, että rigaus, skriptaus, modulaarisuus ja automaattiset työkalut ovat samaa jatkumoa. (Shotarov 2017.) Tätä ajatusta tukee myös havainto siitä, että monen opiskelijan lopputyö onkin usein jonkintyyppinen autorigaustyökalu (Puppeteer Lounge 2018).

3.2 Lähtökohtana suuri tuotanto

Kuva 3. Daniel Calvert kertoo Horizon Zero Dawn-pelin hahmotuotannosta. (Galvert 2018).

Kun tarkoituksena on luoda paljon hahmoja mahdollisimman tehokkaasti, modulaariseen järjestelmään siirtyminen on varsin ilmeinen ratkaisu. Esimerkiksi Horizon Zero Dawn -pelituotanto hyödynsi tekniikkaa menestykkäästi, tavalla jota aion käsitellä tarkemmin ”Design modulaaristen hahmojen apuna” -luvussa. (Calvert 2018.; Kuva 3)

Suuri tuotanto mahdollistaa rigaukseen erikoistumisen ja jopa useita rigaustiimejä. Kun yksittäistä rigiä editoi ja käyttää useampi ihminen, mahdollisesti jopa samaan aikaan, syntyy väkisin virheitä ja innovaatioita, joita ei muuten syntyisi. Seurauksena muodostuu työtapoja ja skripteja, jotka on kannattavaa jakaa työryhmän kesken. Isossa tuotannossa myös ylläpidettävä rigien määrä voi kasvaa hyvin merkittäväksi suhteessa työntekijöiden määrään. Tällaisessa tapauksessa yhden ihmisen kannattavaa käyttää esimerkiksi viikko monikäyttöiseen moduuliin, joka säästää jatkossa muiden rigaajien ja erityisesti animaattorien aikaa. Omat vaatimuksensa tuo kasvava määrä lisäjärjestelmiä kuten mocap-ohjelmistot, animaatiokirjastot ja reaaliaikamoottorit, joita rigien tulee tukea.

Kun moduulit luodaan vakioidusti, voidaan niiden käyttäytymistä ja sisältöä ennakoida huomattavasti tarkemmin. Kun yksittäiset osaset tukevat isoja järjestelmiä valmiiksi ja noudattavat sovittua hierarkiaa ja nimeyskäytäntöjä, on myös omien työkalujen kirjoittaminen helpompaa.

Kuva 4. Pieni osa Destinyn vihollisvariaatioista (Hunt & Sderlind 2015. kohta 3:41)

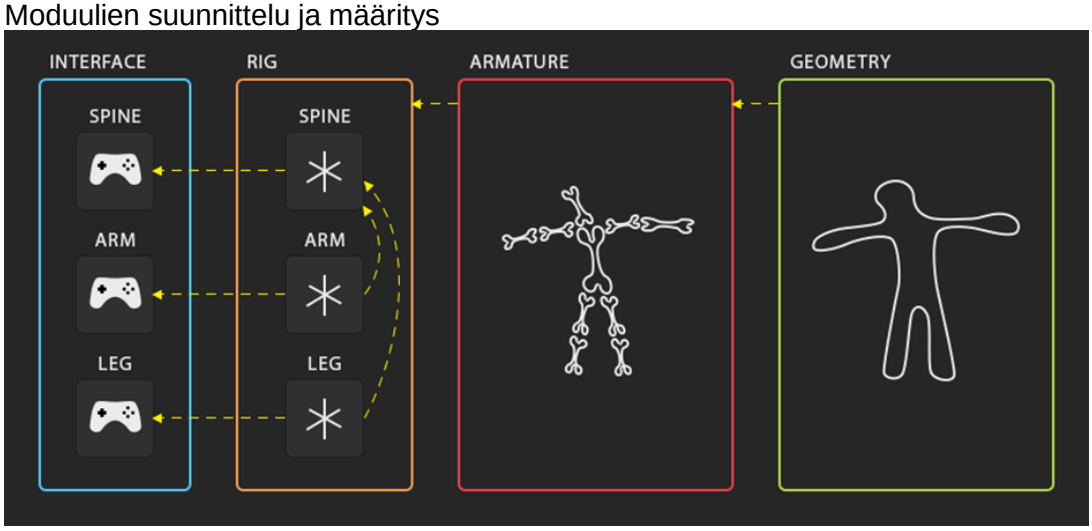
Toinen hyvä esimerkki isosta pelialan tuotannosta on Bungien scifi-räiskintäpeli Destiny. Peli sisälsi useita eri vihollisluokkia, pelaajavariaatioita, yksittäisiä uniikkeja hahmoja sekä merkittävän määrän ajoneuvoja ja aseita (Kuva 4). Lisähaasteena studio kehitti uutta pelimoottoria, jonka vaihtuvia vaatimuksia vanhojen ja tulevien rigien oli tuettava varmuudella. Jo esituotannon aikana tiedettiin, että rigauksen tuli olla nopeaa ja joustavaa. (Hunt & Sderlind 2015.) Game Developer Conference -tapahtumassa antamassaan luennossa Bungien rigaustiimi kertoo omakohtaisesti, miten moduulit haluttiin pitää pitää yksinkertaisina ja animaattoreille helposti muokattavina. Tämän lisäksi niiden piti aina tukea retargetointia eli animaationsiirtoa rigien välillä. Toisin sanoen osia piti kyetä vaihtamaan myös jo animoinnin jälkeen, jolloin animaatio siirretään vanhoista osista uusiin. Voittaakseen edellämainitut haasteet tiimi päätyi päivittämään jo olemassa olleen Xrig-järjestelmän MEL:ista PyMEL:iin, jonka avulla moduulit pystyttiin kehittämään oliopohjaisesti. (Hunt 2009;Hunt 2015.)

Kun uusia moduuleja lähdettiin kehittämään, päätettiin työ jakaa moduuleiksi usein toistuvien ketjujen mukaan, joita David Hunt esittelee alla olevassa kuvassa (Kuva 5).

Kuva 5. David Hunt esittelee Destiny-hahmorigien kehitystä (Hunt 2015).

Kun yksi moduuli oli saatu kehitettyä, saatettiin sitä käyttää pohjana toiselle. Esimerkiksi jalan Reverse Foot Component hyödyntää pohjanaan ensin kädelle suunniteltua FK/IK Componenttia (Kuva 6). Tämän päälle RFC:hen lisättiin jalkarigille tyypillisiä ominaisuuksia, kuten apupainopisteitä, yksinkertaiset FK-varpaat ja jalan rullausominaisuuden eli Foot Roll:in, jonka avulla hahmo voi esimerkiksi nousta varpailleen tai kyykistyä luonnollisesti. (Hunt 2015.) Yhteistä moduuleille on silminnähden esimerkiksi Pole Vector, joka käden tapauksessa ohjaa kyynärpään ja jalan tapauksessa polven taipumissuuntaa.

Kuva 6. FK-IK moduulin esittelyä (Hunt 2015).



Kuva 7. Kaavio yhdestä tavasta jaotella moduuleja (Crouzet 2014.)

Teoriassa mikä tahansa rigissä voi muodostaa moduulin: joint-ketjut, node-kokonaisuudet, skinnaus, control-objektit, utility-hierarkiat, rigin sisäiset scriptit ja erilaiset fysiikkakomponentit kuten vaatteiden liikkuvat osat. Kannattaa huomata, että englannin kielisissä lähteissä näistä puhutaan yleensä nimellä components.

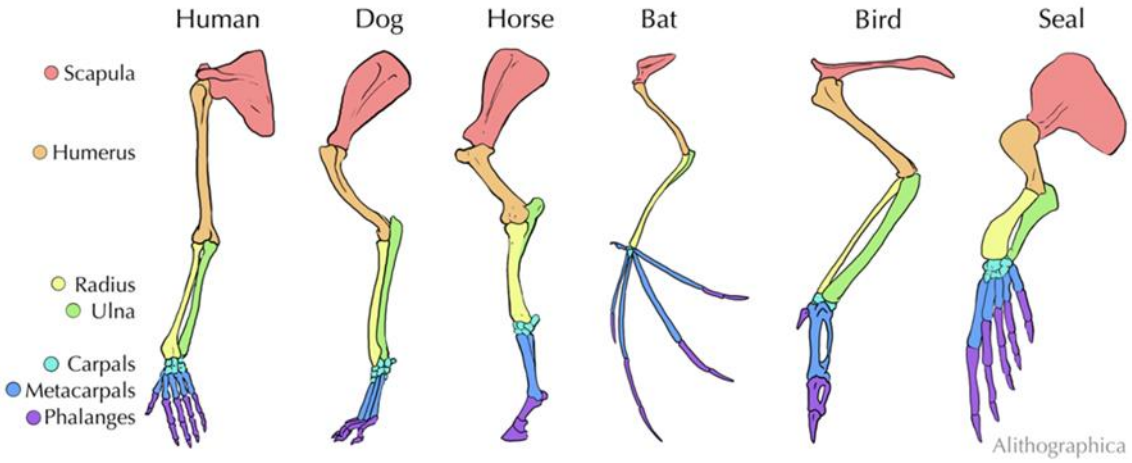
Modulaarisen rigauksen ei tarvitse käsittää pelkästään itse rigiin kuuluvia osia, vaan moduulikirjastoon voi kuulua myös modulaarista geometriaa, esim. päitä ja aseita. Rigausaiheisessa blogissaan Christopher Crouzet kuvailee miten lennosta hyvän moduulijaon keksiminen voi käydä hyvin sotkuiseksi ja työtä menee helposti hukkaan. Siksi on parempi tehdä vertailua jo tekemiensä rigien välillä ja keskittyä muutamaan varmasti toistuvaan isompaan kokonaisuuteen (Crouzet 2014). Moduulin varsinaiseen rigiin liittyvän osan lisäksi on otettava huomioon sen käyttöliittymä, esimerkiksi erilaiset asetukset, jotka voivat olla interaktiossa muiden moduulien kanssa. Tätä havainnoi myös Crouzetin yksinkertainen kaavio (Kuva 7).

Millainen sitten on tyypillinen moduuli? Moduulin perusidea on mahdollisuus yhdistellä luovasti, luoda ikään kuin rakennuspalikoita tuotannon käyttöön. Siksi on luontevaa, että moduulit ovat monikäyttöisiä kokonaisuuksia, esimerkiksi raajoja tai raajan osia, kuten sormia. Listaan ja avaan potentiaalisia moduuleja seuraavassa luvussa.

3.3 Esimerkkejä potentiaalisista moduuleista

Modulaarinen kehitys mahdollistaa keskittymisen yksittäiseen hahmon osaan ja työtä on mahdollista jakaa useammalle tekijälle rinnakkain. Havainnollistaakseni moduulikehityksen mahdollisuuksia esittelen alla muutamia potentiaalisia moduuleja ja näiden teknisiä ominaisuuksia, joista oman moduulikehittelyn voisi esimerkiksi aloittaa.

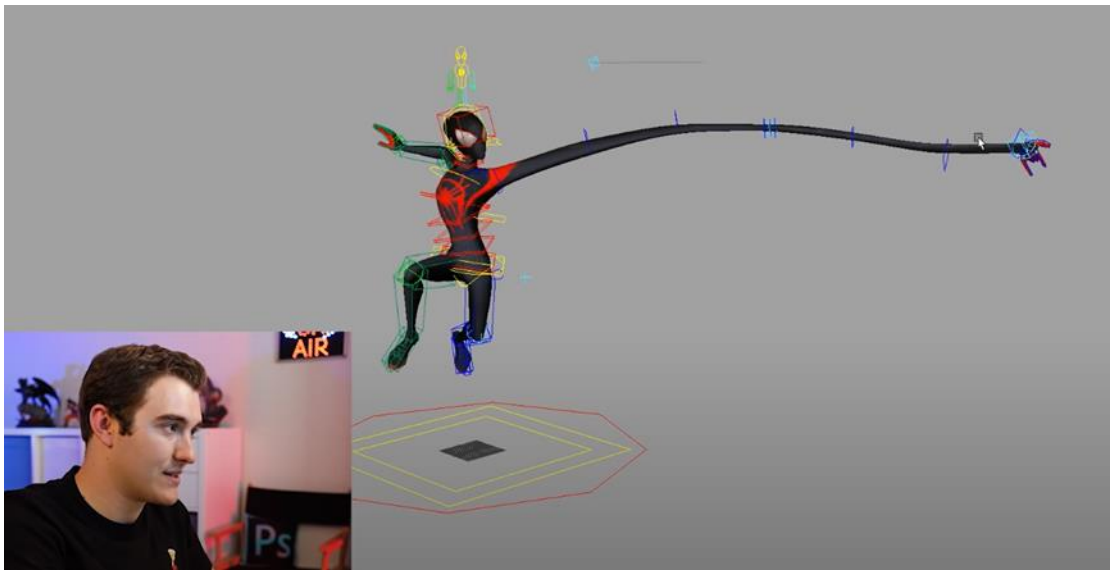
3.3.1 Raajat



Kuva 8. Havainnoiva kuva nisäkkäiden yläraajojen yhtenevistä piirteistä. (Alithographica 2020).

Aloitan esittelyn raajoista, sillä ne ovat hahmosta ja edustetusta lajista riippumatta tärkeitä hahmon toiminnan ja siluetin kannalta. Lisäksi raajojen kesken on paljon potentiaalia modulaarisuudelle, kuten ylläolevasta kuvasta näkyy (Kuva 8). Esimerkiksi lähes kaikkien nisäkkäiden raajoissa on paljon samankaltaisuutta, jolloin yhdestä raajamoduulista on mahdollista muunnella uusia variaatioita, kuten etu-, taka-, ylä- tai alaraajoja.

Rigauksesta oppikirjan kirjoittaneen Rob O'Neillyn mukaan raajalla tulisi vähintään olla ainakin luonnostaan syntyvän Forward Kinematic -ketjun lisäksi Inverse Kinematic Solver sekä ns. elbow pin. Näin raajalla voi ilmaista sekä vapaita kaarevia että kontaktissa syntyviä liikkeitä ja samalla ohjata, mihin suuntaan raaja taittuu. Lisäksi raajalla tulisi olla alku- ja loppupinta, joiden avulla se voidaan liittää ruumiseen ja sitä voidaan jatkaa lisänivelillä. (O'Neilly 2015.185.)



Kuva 9. Animaattori Sir Wade Neidstadt esittelee modulaarisen Spiderverse-rigin yläraajoja. (Neistadt 2020).

Jos halutaan olla mahdollisimman modulaarisia, voi mielestäni raajan, esimerkiksi ihmisen yläraajan, jakaa lapa-solisluu-käsivarsi-ranneketjuun, joka on muunneltavissa vaikka erikokoisille ihmisille tai eläinten eturaajan aihioksi. Syntynyttä moduulia voi sitten jatkaa rannetta tai nilkkaa mukailevalla nivelellä, jossa voi olla erilaisia painon siirtymistä kuvaavia toimintoja, kuten Foot Roll ja Bank eli jalan rullaus ja nilkan kallistus. Moduuli voisi tukea erilaisia sormia tai varpaita, joita voidaan kutsua yhteisnimellä digit. Digit-moduulissa voisi olla FK-kontrollien lisäksi Spread- ja Curl-toiminnot, joiden avulla voidaan esimerkiksi vetää sormet erilleen tai nyrkkiin kätevästi tai osoittaa painon siirtymistä varpaille. Oheisissa kuvissa ammattilaisanimaattori Sir Wade Neidstadt esittelee ilmaiseksi jaossa olevaa Spiderverse-rigiä, jossa on edistyneet ylä- ja alaraajonen ominaisuudet (Kuva 9;Kuva 10).

Kuva 10. Spiderverse-rigin Foot Roll -toiminnon esittelyä (Neistadt 2020).

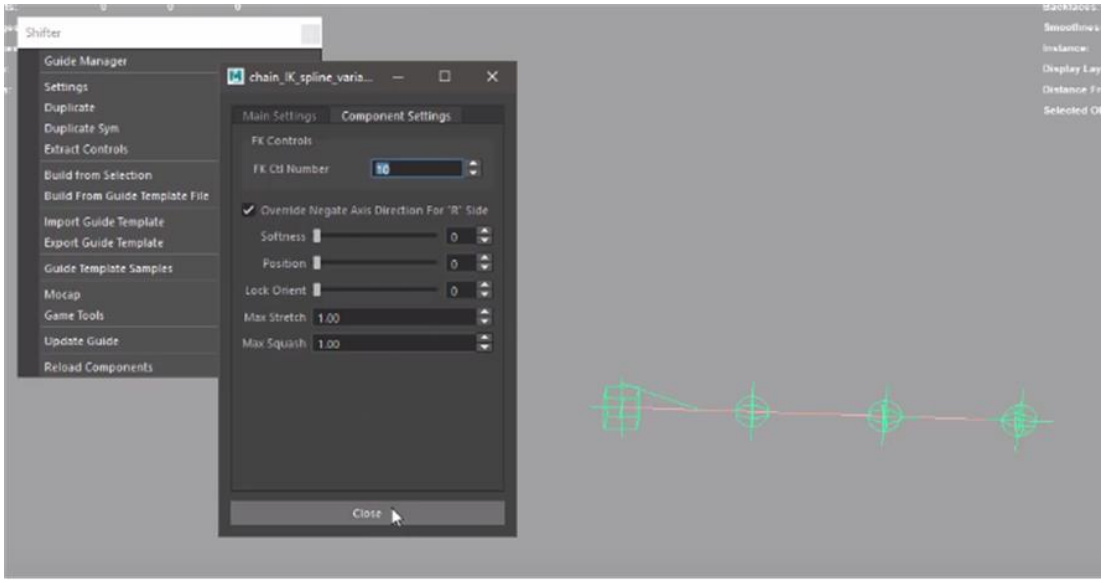
3.3.2 Selkäranka

O'neillyn mukaan yksinkertaisimmillaan selkärangalla voi olla useita tasoja; ensimmäinen taso olisi yksinkertainen FK-ketju, seuraava taso olisi Spline IK Set Up, jossa FK-ketjua ohjataan osittain kurveilla, ja viimeisenä olisi ns. Ribbon Spine. (O'neilly 2015. s.185.) Tässä tekniikassa NURBS-kurvin sijaan selkärangan kääntymistä säätelee NURBS-nauha, josta tekniikan nimikin tulee. (Ross 2015).

3.3.3 Pää ja kaularanka

O'neilly kuvailee kaularankaa yksinkertaiseksi ketjuksi, jossa on sekä FK-ketju että pään ohjaama IK-ketju sekä erilaisia Space Switch -vaihtoehtoja. Näin animaattori voi halutessaan animoida pään seuraamaan muun kehon liikkeitä tai pitämään kokonaan oman liikeratansa. (O'neilly 2015. s.185) Itse näen tästä erityisen hyvänä esimerkkinä linnut, joilla on hyvin ainutlaatuinen tapa pitää kaula vakaana rajuissakin liikkeissä. Tällaista olisi hyvin vaikea saavuttaa ilman Space Switch -vaihtoehtoa, jossa pää seuraa hartian sijaan esimerkiksi maailman sijaintia.

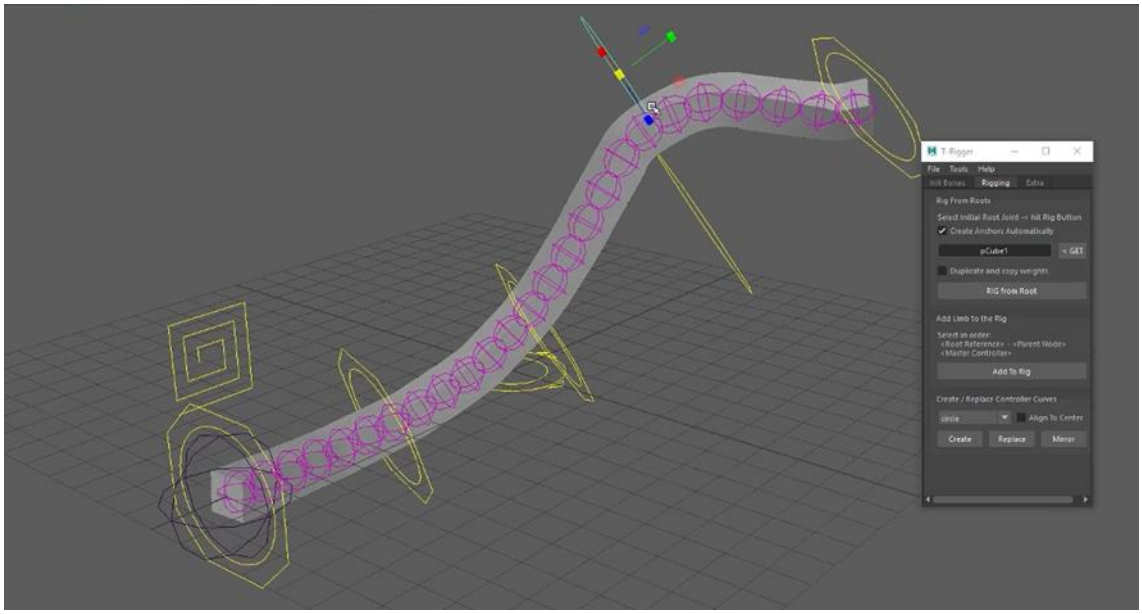
3.3.4 Häntä



Kuva 11. Kuvakaappaus Spline-FK-ratkaisusta, joka sopii esimerkiksi hännälle. (Campos 2019)

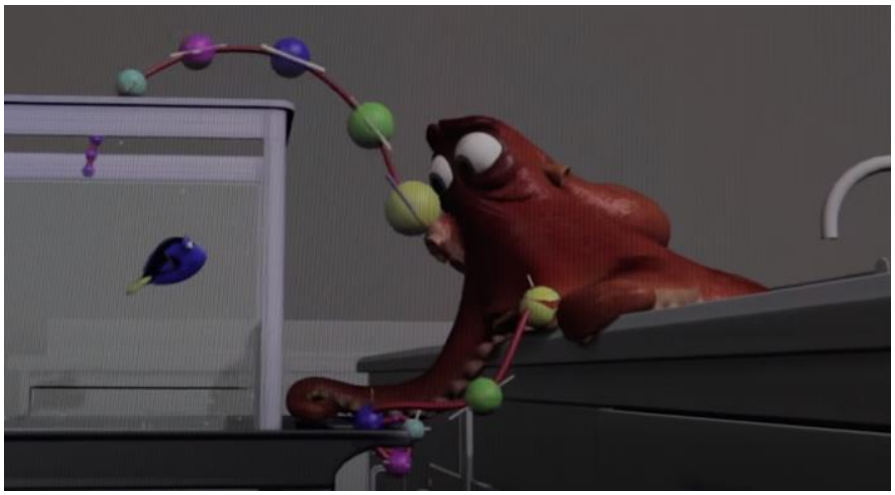
Kaikista aiemmin mainituista kohteista häntä vaikuttaa kaikista yksinkertaisimmalta rigata. Yksinkertaisilla hahmoilla usein pelkkä FK-ketju riittää, jos halutaan tehdä luontevia kaaria. O'neilly on tehnyt järkevän havainnon, että riippuen hännän pituudesta ja hienojakoisuudesta, suora FK-ketju voi olla liian sekava animoida. Tässä tapauksessa järkevämpää on tehdä NURBS-pohjainen spline-ratkaisu, jossa pienempi määrä kontrolleja ohjaa isompaa kokonaisuutta NURBS-nauhan avulla. (O'neilly 2015. s.185) Tällaisia moduuleja on saatavilla valmiina esimerkiksi myöhemmin käsittelemässäni Shifterissä. Tämä mahdollistaa sen, että sekä ylätason kontrolleja että alatason jointteja voi generoida automaattisesti ja kokeilla nopeasti, mikä olisi kontekstiin sopiva määrä. (Kuva 11; Campos 2019).

3.3.5 Lonkero



Kuva 12. Suunnittelija esittelee lonkeromoduulia. (Kutlu 2018).

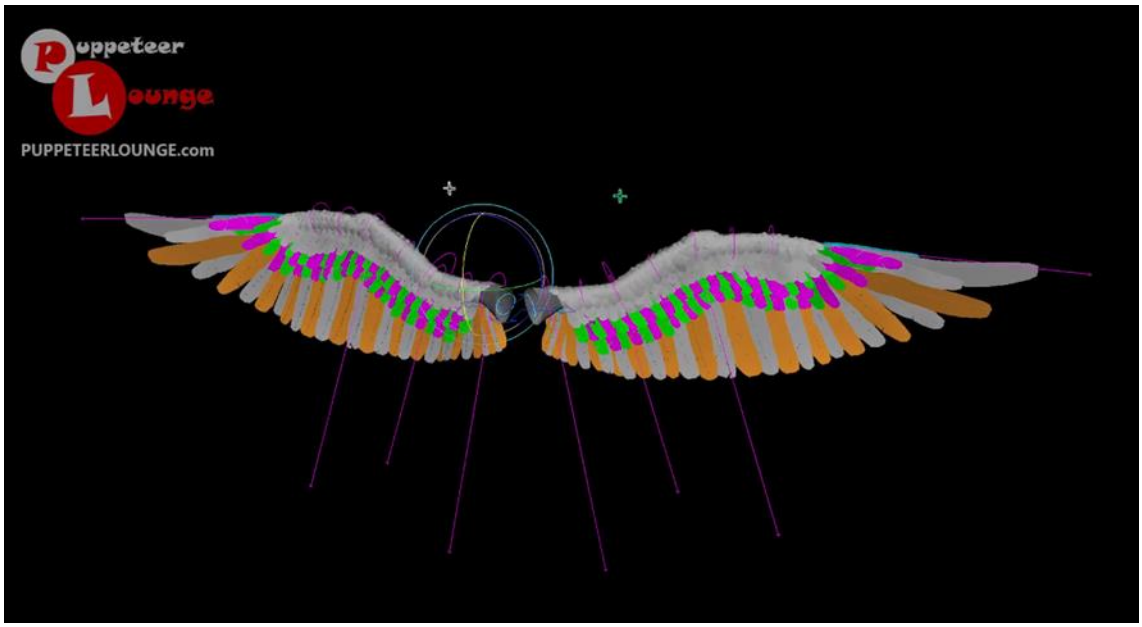
Miellän erilaiset lonkerot kasvorigien rinnalla yhdeksi monimutkaisimmista ongelmista, joita taitavimmat rigaajat pyrkivät ratkaisemaan uudelleen ja uudelleen, kuten esimerkiksi Arda Kultu on tehnyt omassa T-rigger-järjestelmässään (Kuva 12). O'neilly tekee kirjassaan huomion, että suurin ero aiemmin mainittuun häntään on se, että lonkeron on usein tuettava painonsiirtoa. Tästä huippuesimerkki on *Pixarin Doria etsimässä* -elokuvan Hank, jonka lonkeroiden kehittämiseen meni yhtiön mukaan noin 6 kuukautta (Kuva 13; Talbot 2016).



Kuva 13. Hank-hahmon lonkero-moduuli. Videon kohdasta 2:30 (Talbot 2016).

3.3.6 Siipi

Siipiä on hyvin monenlaisia. O'neilly korostaa niiden yhteiseksi nimittäjiksi kyvyt vetäytyä suppuun, taipua ja läpyttää. (O'neilly 2015.185.) Näen itse, että yksinkertainenkin siipimoduuli voisi olla erityisen hyödyllinen esimerkiksi fantasiatuotannoissa. Tällainen moduuli voisi olla luonnollinen yhdistelmä raajamoduulia ja erityisesti tähän suunniteltuja digittejä. Alla ammattimaista rigausta edustavan Puppeteer Lounge -koulun esittelyä siipien rigaamisesta (Kuva 14).



Kuva 14. Edistynyttä siipien rigausta. (Puppeteer Lounge 2018, kohta 2:50)

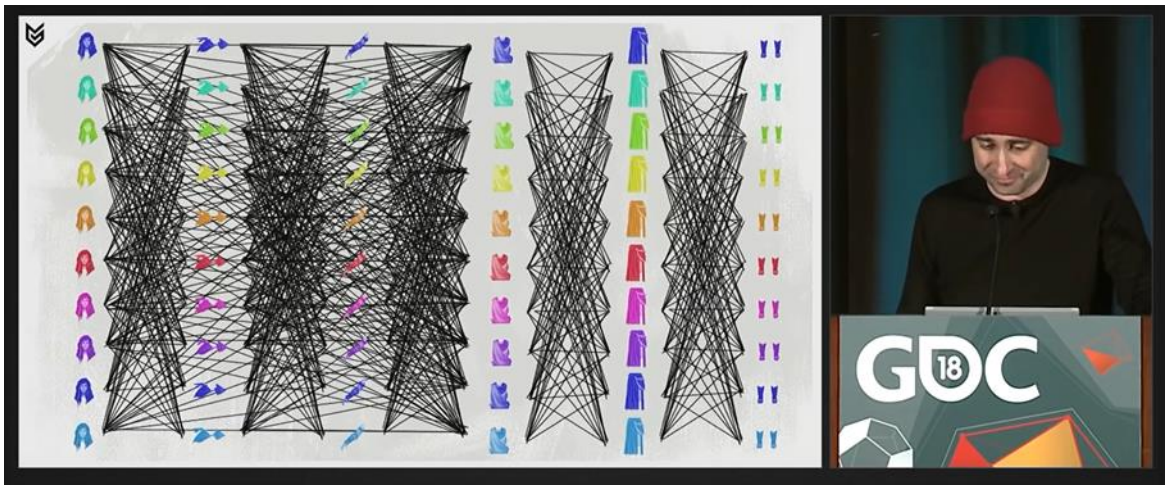
4 Design modulaaristen hahmojen apuna

Hahmojen rigaus ja mallinnus kulkevat käsi kädessä. Rigin asettelu on kokonaan sidoksissa malliin ja malleja korjataan, jotta ne tukisivat erilaisia animaation aiheuttamia muotoja mahdollisimman hyvin. Siksi modulaarinen rigaus on samaa jatkumoa modulaarisen mallintamisen ja animaation kanssa. Peleissä näkeekin paljon näiden kolmen yhdistelmiä, joissa hahmot koostuvat modulaarista rigeistä ja malleista ja joissa animaatiota käytetään erillisissä tasoissa. Näin samaa dataa voidaan käyttää useammissa yhteyksissä. Seuraavaksi käsittelen Horizon Zero Dawn -pelin modulaarisia hahmoja, sillä niiden suunnitteluprosessi, ongelmat ja ratkaisut kuvastavat mielestäni hyvin modulaarisuuden luonnetta ja sitä, miten modulaarinen rigaus ja mallinnus vaikuttavat toisiinsa.

Kuva 15. Horizon Zero Dawnin moduuleja. (Calvert 2018).

Vuonna 2017 ilmestynyt toimintaroolipeli Horizon Zero Dawn (HZD) tuli tunnetuksi näyttävistä robottihahmoistaan ja näitä metsästäneistä kansoista, joiden edustajia löytyy pelistä huomattavan paljon. Pelin julkaisun jälkeen pelistudio Guerilla Games esitti Game De-veloper Conference -tapahtumassa useita luentoja, joista yksi sivusi sitä, miten juuri modulaarisuus ja tarkoin harkittu design auttoivat suuren ja elävän hahmorepertuaarin suh-teellisen pienellä pelitiimillä.

HZD:n taiteellinen johtaja Dan Calvert kertoi, että paperilla peliä varten suunnitellut moduulit toimivat yhdistettynä loistavasti, mutta kun alustavia moduuleja lähdettiin mallintamaan, tuli vastaan merkittäviä ongelmia. Hahmoilla oli suunniteltu paljon vaatekerroksia, jotka aiheuttivat paljon päällekkäisyyttä (Kuva 15). Moduulien yhdistelemisestä tuli vaikeaa, kun osaset eivät saa liikkeessä mennä toisistaan läpi, mikä aiheuttaa jatkuvaa lisätyötä rigausvaiheessa. Suunnittelussa olikin jatkuvasti tehtävä kompromisseja, jotta osaset toimisivat mahdollisimman hyvin yhteen. Hyvässä suunnittelussa on kuitenkin Calvertin mukaan aina pieni tila kompromisseille. Hänen mukaansa modulaarisen järjestelmän haaste on se, että jokainen uusi osa lisää vuorovaikutusta ja ongelmia osien välillä. Mitä enemmän vuorovaikutusta, sitä enemmän tarvitaan kompromisseja. (Calvert 2018.)



Kuva 16. Dan Calvert kuvaa moduulien vaikutuspiiriä toisiinsa. (Calvert 2018).

Ongelmaa olisi voinut ratkaista asettamalla moduuleille tarkat rajat, jolloin erilaisten ongelmien määrä pienenee, mutta tällöin myös moduulit alkavat näyttää liikaa samalta. Vaarana oli myös, että ongelmia vältellessä uhrataan paljon yksityiskohtia, jotka tekevät hahmoista uniikkeja. Lähtötilanteen moduulien vaikutuspiiriä Calvert kuvaa erityisesti yllä olevassa kaaviossa, joka havainnoi hienosti mahdollisten ongelmien eksponentiaalista kasvua (Kuva 16). Horizon Zero Dawnin tapauksessa suunnittelua helpotettiin jakamalla moduulit kahteen kategoriaan sen mukaan, kuinka paljon uniikkiutta ne tuovat peliin (Calvert 2018.)

Kuva 17. Karsinnan ja uudelleen suunnittelun tuloksia. (Calvert 2018).

Lopulta palasten määrää vähennettiin yhdistämällä palasia toisiinsa, jotta tuloksena syntyneet isot palat olisivat mahdollisimman uniikkeja. Osia, joiden variointi ei ollut niin oleel-

lista, tuotettiin vähemmän (Kuva 17). Kun ongelmallinen vuorovaikutus itsenäisten palasten välillä minimointiin, saatiin hahmoista mahdollisimman uniikkeja ja niiden yhdisteleminen helpottui merkittävästi (Calvert 2018.)

5 Rigaus ja ohjelmointi

Kolme kokenutta ammattilaisrigaajaa striimasi 2018 keskustelun, jossa vertailtiin rigausta ja perinteisempää ohjelmistokehitystä toisiinsa. Esille nousivat mm. selkeästi määriteltävä input ja output, kehitysprosessihallinta, sekä yhteensopivuuteen ja käyttäjäystävällisyyteen pyrkiminen. Tämän takia voitiin perusteella, että rigaus itsessään on sovelluksenkehitystä ja ohjelmointia riippumatta siitä onko koodia kirjoitettu käsin, vai syntyykö se käyttöliittymän avulla (Anzovin, Campos, Fragapane 2018.) Siksi mielestäni kaikenlaiseen rigaamiseen ja erityisesti moduulin kehittämiseen voidaan myös soveltaa kahta modernin ohjelmoinnin perusperiaatetta: KISS ja DRY (Peters 2012).

5.1 KISS ja DRY

Alun perin ohjelmistonkehittäjien viljelemä mantra KISS eli ”Keep it Simple, Stupid” pätee mielestäni myös rigauksessa yleisesti. Käytännössä moduuleissa tulisi aina pyrkiä yksinkertaisuuteen, sillä yksinkertaisen moduulin käyttäytymistä on helpompi ennakoida, korjata ja myös opettaa muille käyttäjille.

Toinen tunnettu periaate on DRY eli Don’t Repeat Yourself (Peters 2012). Perimmäinen syy moduulien luonnille on välttää samankaltaisten ja puuduttavien työvaiheiden toistamista, jolloin aikaa jää mekaanisen ja toistuvan työn sijaan luovemmille ja haastavimmille ongelmille. Moduulien käytön opettaminen on myös nopeampaa, kuin sellaisen luomisen opettaminen alusta pitäen. Lisäksi inhimillisten virheiden määrä vähenee, kun mekaanista työtä tehdään käsin vähemmän.

5.2 Tapoja moduulien rakentamiseen

Kirjassaan ”Digital Character Development: Theory and Practice” Rob O’Neilly sivuaa nopeasti muutamia erilaisia tapoja lähestyä moduulien rakentamista. Karkeasti jaoteltuna modulaarisia rigejä voi koostaa neljällä tavalla alkaen manuaalisesta työstä jatkaen kohti puoliautomaattista työtapaa:

a) kopioimalla moduuleja käsin, jolloin uudet kopiot eivät ole sidoksissa alkuperäisiin. Tämä on erittäin nopea, mutta tuhoisa tapa luoda uusia moduuleja, sillä mahdolliset päivitykset on tehtävä suoraan syntyneisiin kopioihin. Sopii mielestäni tilanteeseen, jossa kaivataan nopeita prototyyppeja, joita ei jatketa sen pidemmälle.

b) referoimalla eng. referencing, eli luomalla kopion, joka viittaa alkuperäiseen moduuliin. Jos alkuperäistä moduulia tämän jälkeen muokataan, siirtyy muokkaus automaattisesti myös sitä referoiviin kopioihin. Haittapuolena on se, että aina päivitysten vaikutusta ei voi ennakoida ja voi syntyä tilanteita, joissa muutoksen ei haluta koskevan kaikkia referoivia kohteita (O’Neilly 2015, luku 20.3.)

c) kehittämällä moduulin kaksivaiheisesti, tekemällä ensin prototyypin ja sitten kirjoittamalla sen auki koodiksi, joka voidaan suoran kopioimisen sijaan ajaa. Koodia on helpompi muokata ja varioida siististi ja nopeasti ja päivitykset voidaan vain haluttuihin kohteisiin.

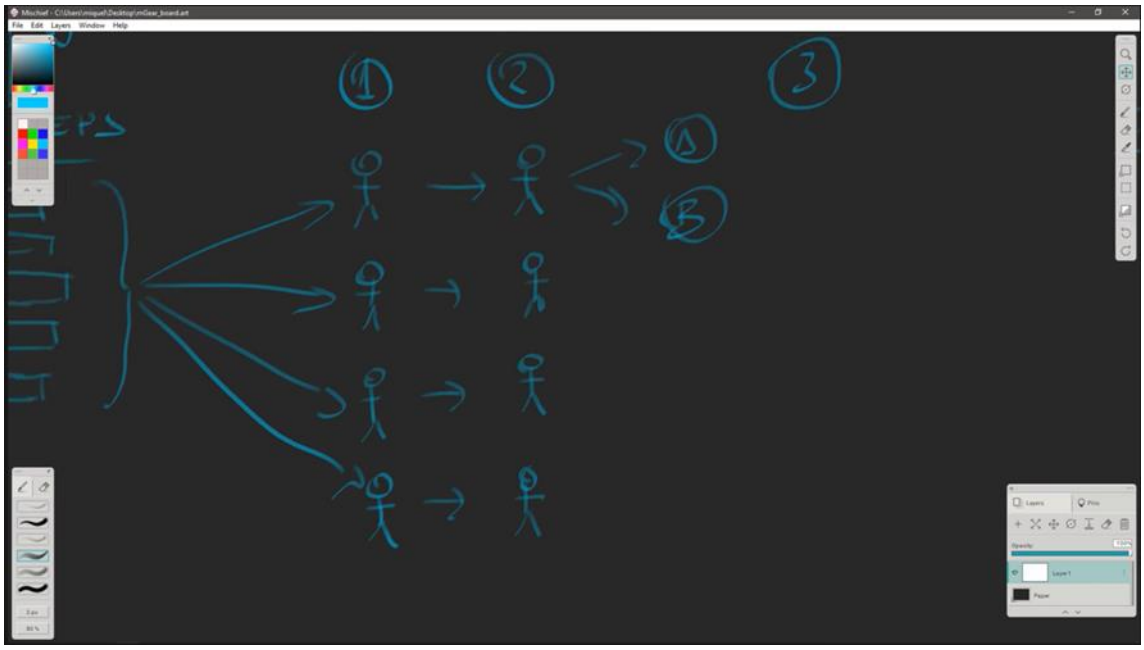
d) muuntamalla moduulit luettaviksi resepteiksi, joiden pohjalta automatiikka sitten uudelleen rakentaa halutun moduulin. Prosessia kutsutaan serialisoinniksi eng, data serialization, jossa kaoottinen data typistetään ja järjestellään koneelle tai ihmiselle luettavaan ja tallennettavaan muotoon, jonka pohjalta alkuperäinen tuotos voidaan toisintaa. Kopiointiin verrattuna data siis muuttaa muotoaan reseptiksi ja takaisin. Näin päivitykset voidaan tehdä halutessa itse rigien sijasta resepteihin, joita on helppo lukea ja muokata. (O’Neilly 2015, luku 20.4.)

Reseptien hyvä puoli on se, että niitä voi muokata koskematta itse koodiin, joka suorittaa itse rigin rakentamisen, näin itse rigit ja ne rakentava ohjelma on eroteltu toisistaan, jolloin teoriassa myös rakentavaa ohjelmaa voi muuttaa tai jopa vaihtaa dramaattisesti, kunhan myös uusi ohjelma osaa lukea samoja reseptejä. Tällaista

kehitystyyliä kutsutaan datakeskeiseksi, eng. Data Centric, joka käytännössä tarkoittaa tapaa, jossa itse tuotetta, tässä tapauksessa rigin osia ja sen reseptiä pidetään pysyvämpänä ja arvokkaampana kuin sitä tuottavaa ohjelmaa. Toisin sanoen rigin ominaisuudet määrittävät mitä ohjelman pitäisi osata lukea, eikä toisinpäin. Lisäksi rigi on jaettu pienempiin osiin, jotta ohjelma voi yhdistellä niitä annetun reseptin mukaisesti. (Campos 2018; McComb 2016.)

Kehysohjelman luominen ei kuitenkaan ole kovin yksinkertaista ja siksi käsittelen jo olemassaolevia ohjelmia vielä erikseen seuraavassa luvussa.

5.3 Kaaos haltuun sovelluskehyksellä

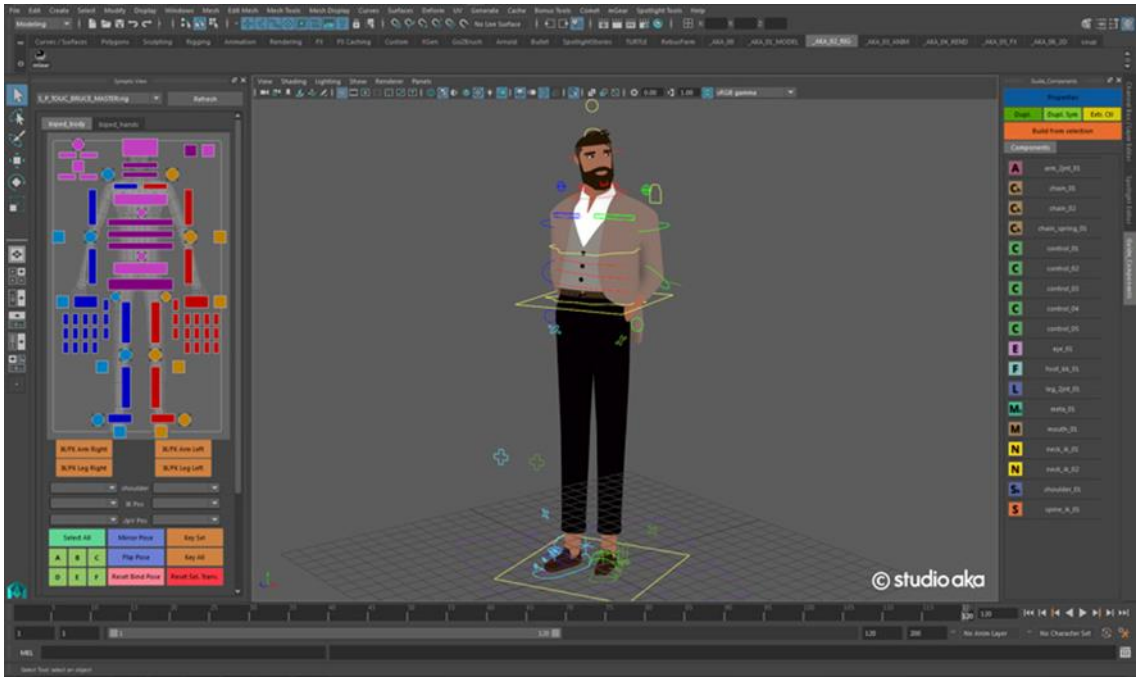


Kuva 18. Miguel Campos havannoi dataan perustuvia hahmoja (Campos 2018).

Rigien jakaminen pienempiin osiin ja älykkäisiin kirjastoihin on myös suuri haaste jo pelkästään organisoinnin kannalta. Tätä varten studiot ja intohimoisimmat harrastajat päätyvät rakentamaan kehysohjelmia tai koodikirjastoja, joilla hallinnoidaan työnkulkua ja saadaan rigien yksittäiset osat liittymään paremmin keskenään, sekä päivittymään automaattisesti. Englanniksi näistä ohjelmista käytetään englanniksi nimitystä framework, jonka voisi vapaasti suomentaa ohjelmistokehykseksi. Yhden tällaisen kehysohjelman kehittäjä Miguel Campos havainnoi kuvassaan miten resepteistä saadaan aikaiseksi erilaisia hahmoja ja näiden variaatioita (Kuva 18.). Tarkemmin ajateltuna tällaisten ohjelmien tai kirjastojen käyttö tähtää siihen, että kehitysaika menisi manuaalisten prosessien toiston sijaan monimutkaisempien ongelmien

ratkaisuun, esimerkiksi juuri moduulien kehittämiseen. Kun luodaan uusi moduuli tai jonkin ongelman ratkaiseva päivitys, voidaan se lisätä automaattisesti haluttuihin rigeihin, jolloin säästetään sekä aikaa, että vältetään inhimillisiä virheitä. Samalla kehitystyö muuttuu lineaarisesta työstä rinnakkaiseksi, jolloin vaikka kaksi ihmistä voi suunnitella ja toteuttaa osia samaan rigiin yhtä aikaa (Shotarov 2017.)

Ajatuksena on että, tulevaisuuden rigit ovat nopeampia, vakaampia ja niiden järjestelmätukea on helpompi laajentaa (Crouzet 2014.)



Kuva 19. Valmis mGear-rigi Synoptic ja Shifter-työkalujen välissä (mGear Developers & San Pellegrino 2019

Isojen studioiden in-house ohjelmien, kuten aiemmin mainittu Bungien Xrig, lisäksi on olemassa myös muutamia kaupallisia ratkaisuja, kuten Rapid Rig ja Advanced Skeleton 5. (Hunt 2015; Nelson 2019; Animation Studios 2020). Myös ilmaisia avoimen lähdekoodin ratkaisuja on olemassa, näistä tunnetuin on mGear, jonka käyttöä esittelen seuraavassa luvussa.

6 mGear käytännön esimerkkinä

mGear on avoimeen lähdekoodiin perustuva rigaus- ja animaatioslisäosa, jonka voi luoda hyvin monenlaisia rigejä (Kuva 19). Sen tarkoituksena on nopeuttaa ja tehostaa

edistynyttä rigausta sekä tuoda automaattisia työkaluja animaattoreille ja rigaajille. mGear:in esikuvana on Softimage-ohjelmalle luotu Gear-lisäosa, jonka erityisesti modulaarisia ominaisuuksia se pyrkii jäljittelemään (mGear developers 2019.) Mainittakoon, että vaikka Softimagea ei juuri enää käytetä, oli sillä suuri merkitys erityisesti 2000-luvun taitteen viihteessä ja esimerkiksi alkuperäinen Jurassic Park, Star Wars ja Legend of Zelda-tuotannot käyttivät sitä hahmojen rigaamisessa.(Burns 2014; InspirationTuts 2019). Vaikka Softimage lopulta hävisi monessa suhteessa Mayalle, on joillekin sen jo kuopatuille ominaisuuksille edelleen kysyntää, mikä onkin syy alkuperäisen mGear-projektin syntymiselle.

Erityisen houkuttelevaksi esimerkiksi mGear:ista tekee se, että se itse on myös erittäin modulaarinen, tukee omien mekanismiensa muokkaamista ja mahdollistaa uudelleen rakennettavien rigien tekemisen käyttäen valmiita työkaluja ja kirjastoja. Esittelen seuraavaksi mGearin toimintaa, osia ja omaa mGear:in Shifter-työkalulla tehtyä projektiani, joilla on helppo havainnollistaa millaista moduulityöskentely voi olla käytännössä ja millaisia ongelmia mGearin kaltaiseen järjestelmään sisältyy.

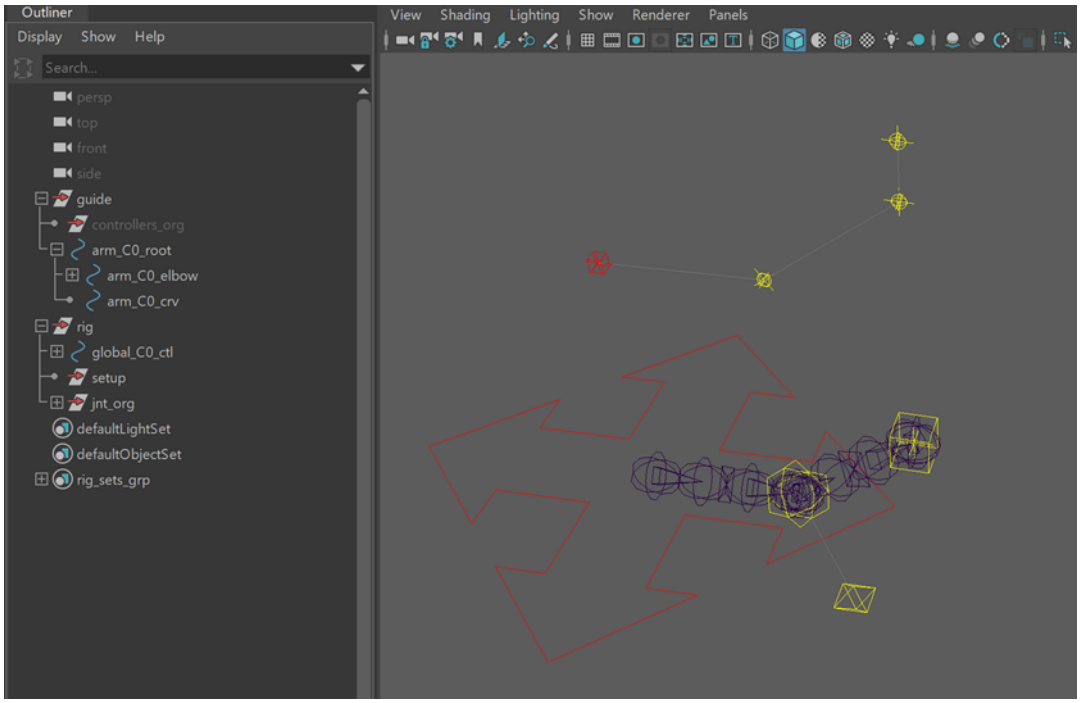
6.1 mGearin merkittävimmät osat lyhyesti

- ***Shifter Rig Builder***, mGearin moduulijärjestelmä ja autorigaaja
- ***Rigbits* & Animbits**, sekalainen kokoelma rigaus- ja animaatioaiheisia työkaluja
- ***RBF manager***, työkalu Radial Basis Function-pohjaisten deformeireiden työstämiseen
- ***Synoptic & animpick***, työkalu graafisen valintakartan tekemiseen ja ylläpitoon
- ***Simplerig***, yksinkertainen autorigaaja, sopii esimerkiksi aseille.
- ***Flex*,** geometrian päivitystyökalu, jonka avulla vanhaa geometriaa voidaan päivittää asteittain ilman uudelleen skinnausta. Työkalulla voidaan analysoida eroja kahden geometrian välillä.
- ***Mocap & humanIK tools***, työkaluja joiden avulla biped-rigi voidaan säätää humanIK-yhteensopivaksi ja tuoda sisään motion capture-dataa.

(mGear developers 2019.)

6.2 Shifter-rigauksen ydinkonseptit

Shifter on mGear:in puoliautomaattinen rigaustyökalu, joka sisältää valmiita moduuleja. Automaattisia rigaustyökaluja, ns. autorigaajia, yhdistää yleensä se, että ne ovat kaksivaiheisia, ensimmäinen vaihe sisältää jonkinlaisen aseteltavan esirigin, jonka käyttäjä asettelee vastaamaan haluttua mallia, yleensä jonkinlaista humanoidia. Seuraavaksi käyttäjä valitsee joitain asetuksia ja käynnistää varsinaisen autorigauksen, jonka seurauksena syntyy esirigin asentoa myötäilevä varsinainen rigi (mGear Developer 2020)

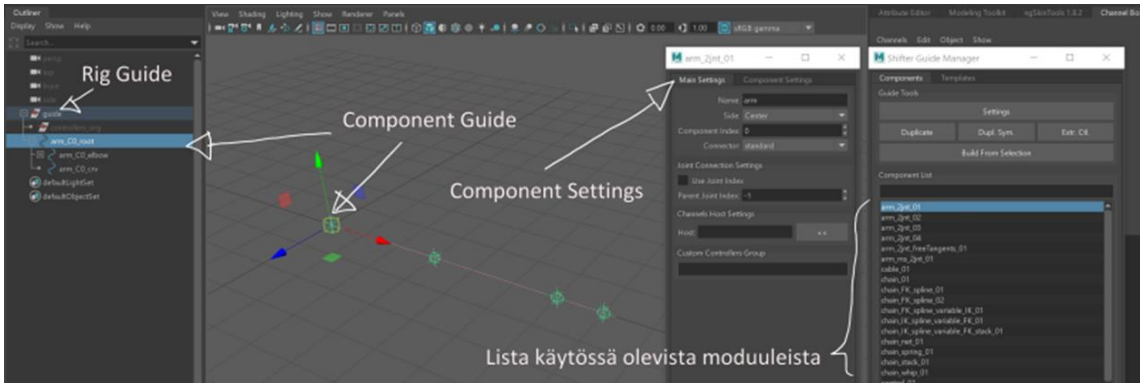


Kuva 20. Ylhäällä Shifterin luoma Guide ja alhaalla sen pohjalta rakennettu rigi. (mGear Developers 2020).

6.2.1 Component

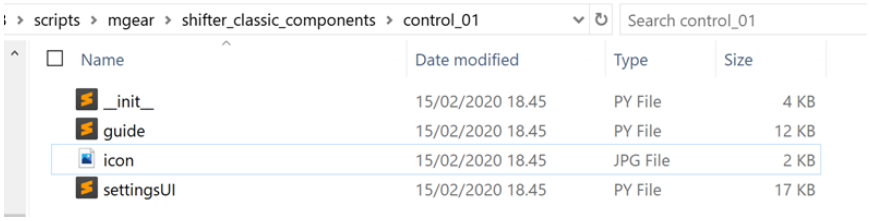
Shifterin kontekstissa yhtä rigin moduulia kutsutaan nimellä Component. Työkalun mukana tulee kirjasto, nimeltään Classic Components, joka sisältää valmiita komponentteja eli moduuleja. Nämä komponentit ovat vapaasti tutkittavissa ja muokattavissa, myös kaupallisesti. Tällä hetkellä kirjasto sisältää esimerkiksi erilaisia jalkoja, ketjuja, selkärankoja ja apuobjekteja. Kirjastoa laajennetaan mGear-päivitysten ohessa, jonka lisäksi, tavalliset käyttäjän jakavat omia komponenttejaan mGear-keskustelupalstalla.

Jokaisella komponentilla on kaksi muotoa; esirigi, nimeltä component guide ja varsinainen rigi, joka pohjautuu käyttäjän asettelemaan guideen (Kuva 20).
(mGear Developers and users 2020.)



Kuva 21. Havainnoiva kuva Shifterin käyttöliittymästä (Liite 1, 1/1).

Seuraavaksi kerron tarkemmin, kuinka Shifterin moduulit syntyvät, sillä luodakseen uusia moduuleja, on käyttäjän muokattava tai jatkettava seuraavaksi mainitsemiani osia, jotka on kirjoitettu Python 2.7 -kielellä.



Kuva 22. Kuvakaappaus Control_01-komponentin sisältämistä moduuleista (Liite 1, 2/2).

Teknisesti jokainen Component koostuu seuraavista Python-moduuleista:

- **Component Guide**, jossa määritellään millaisia osia component guidessa on. Käytännössä nämä osaset merkitsevät paikkoja tuleville rigin osille ja niitä asettelemalla käyttäjä voi määritellä mihin asentoon varsinainen rigi rakennetaan.

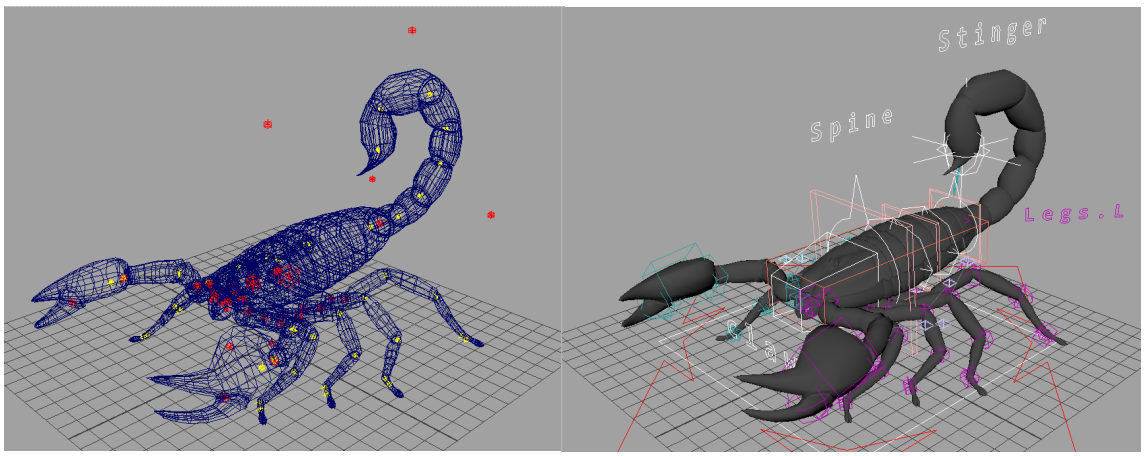
- **Settings UI**, miltä komponentin käyttöliittymä näyttää ja mitä asetuksia siinä on. Tämän luokan avulla moduulin kehittäjä voi antaa käyttäjälle mahdollisuuden vaihtaa ja antaa rigin rakentamiseen liittyviä erilaisia muuttujia, kuten montako niveltä rakennetaan tai onko IK-solver oletuksena päällä vai ei.

- **_Init_**, jossa määritellään miten kyseisen guiden pohjalta rakennetaan rigi. Luokka saa visuaalisen Component Guiden asennon pohjalta tiedon mihin asentoon rigi rakennetaan ja asetuksista esimerkiksi nimen rakennettavalle rigi moduulille. Aihetta selkeyttääkseni olen ottanut kaksi kuvakaappausta havainnollistaakseni miltä Component näyttää Mayassa ja miltä taas tekstimuodossa (Kuva 21;Kuva 22).

Kun aiemmin mainitut osat suoritetaan Mayassa, syntyy ensin visuaalinen Guide, jota lopuksi rakentuva rigi käyttää pohjanaan. On tärkeää huomata, että yhdistelemällä esimerkiksi kaksi komponenttia syntyy tuloksena yksi ylätason Rig Guide, eli Guide ja Component guide eivät ole synonyymejä keskenään.

6.2.2 Rig Guide

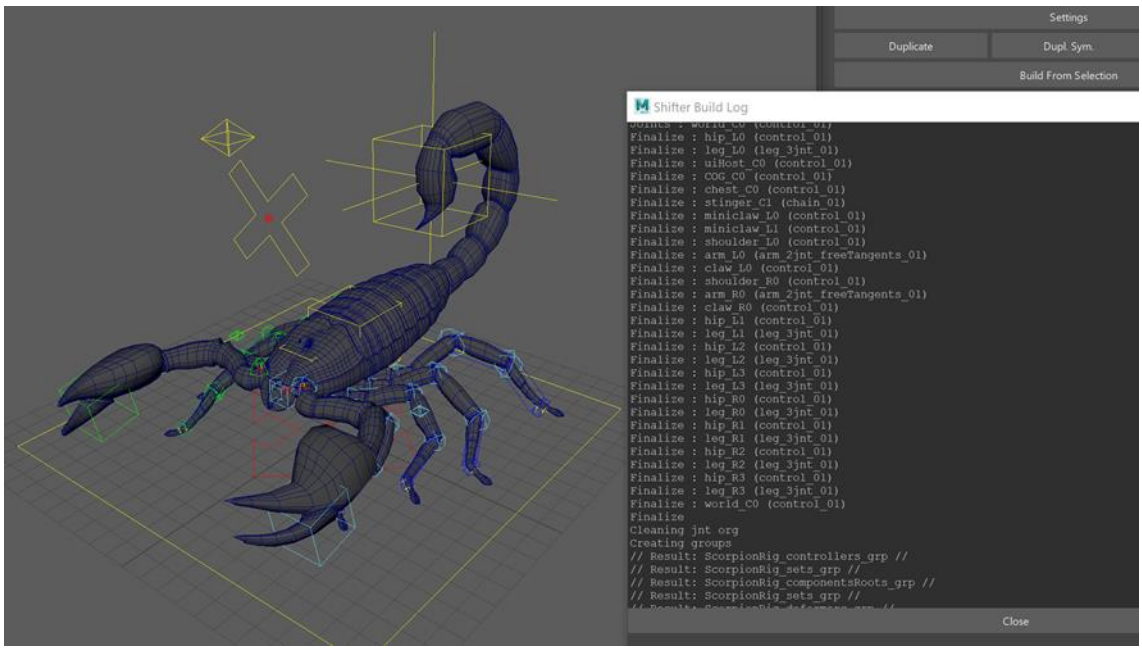
Komponentin ensimmäinen näkyvä osa. Kun käyttäjä luo komponentin, luo Shifter automaattisesti sille guiden, guide voi sisältää useita komponentteja ja on ns. aseteltava luonnos lopulta rakennettavasta rigistä. Kun asettelu on tehty, valitaan guide ja painetaan ”Build from selection” Tällöin Shifter rakentaa varsinaisen rigin Guideen sisältyvien osien pohjalta (Kuva 23). mGear käyttää sekä component guidesta, että Rig Guidesta nimeä guide ja eroa selventääkseni olen antanut kokonaista rigiä edustavalle guidelle erisnimen Rig Guide.



Kuva 23. Projektin Rig Guide ja tuloksena syntyvä rigi (Liite 2 1/2)

6.2.3 Build ja Rig

Varsinainen rigi rakentuu automaattisesti vaiheittain ennalta määrittelemässä järjestyksessä buid-työvaiheessa joita voidaan tarkastella Shifterin Build logista (Kuva 24). Tuloksena on rigi, jota voidaan testata ja korjata tai käyttää sellaisenaan. (mGear developers 2019.)



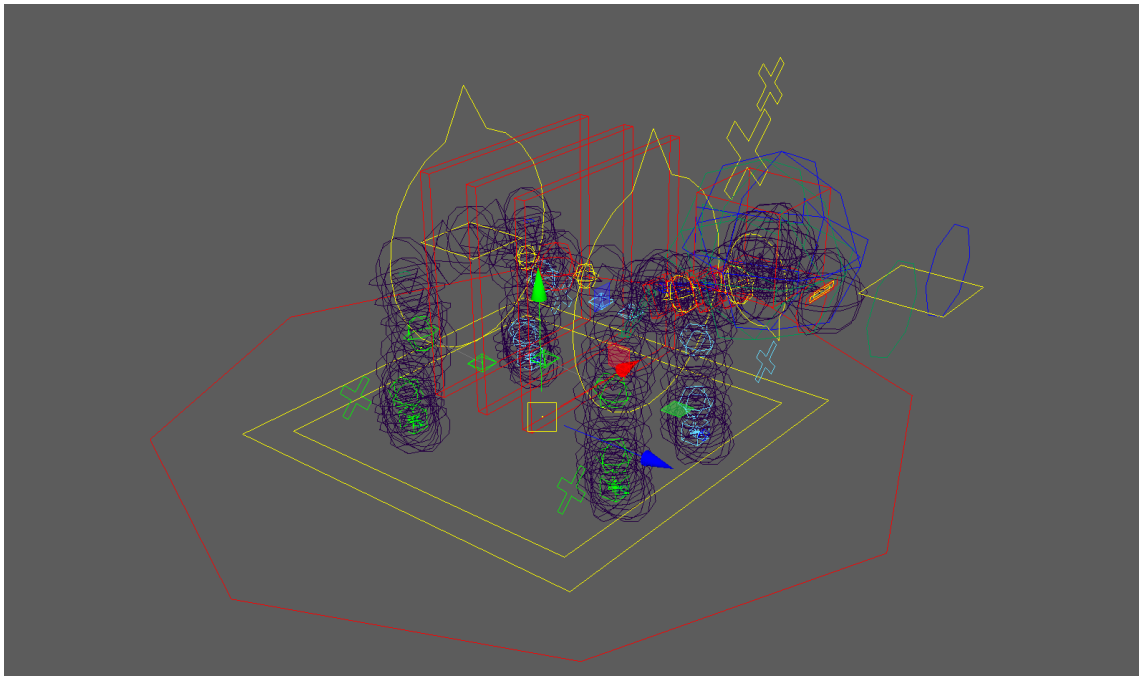
Kuva 24. Scorpion-rigi rakentumassa. Kuvassa oikealla Shifter Build Log (Liite 2, 1/2).

6.2.4 Custom Steps

Tuloksena syntyvää rigiä voidaan valmistella tai viimeistellä Custom Pre- ja Post Stepeillä. Rigin rakennusvaiheen edelle tai jälkeen voidaan lisätä omia skriptejä, jotka ajetaan Build-vaiheen ohessa automaattisesti, esimerkiksi kontrollien nimiä voidaan vaihtaa jälkikäteen koskematta itse guideen. (mGear developers 2019.)

6.2.5 Guide Template

Hyväksi todetusta yhdistelmästä voi tallentaa itselleen Guide Templaten, eli useammasta guidesta koostuvan kokoonpanon. Oletuksena näitä on valmiina kaksi: Biped, joka on suunnattu ihmismäisille humanoideille ja Quadruped, joka on tarkoitettu nelijalkaisille olioille. Alla kuva esimerkiksi koiralle sopivasta Guide Templatesta (Kuva 25; mGear developers 2019.)



Kuva 25. Samannimisen guide Templaten pohjalta rakennettu Quadruped-rigi (Liite 2 1/2).

6.2.6 mgear.Core

mGearin sisäinen logiikka, kirjasto Python-moduuleja, joista Guide rakentuu ja joka ajaa varsinainen rigin rakennusprosessin läpi, sekä määrittelee mm. käyttöliittymien rakenteen ja toiminnan. Muokkaamalla Corea käyttäjä pääsee käsiksi koko mGearin toimintaan (mGear developers 2020).

7 Valmiiden moduulien yhdistämisen työnkulku

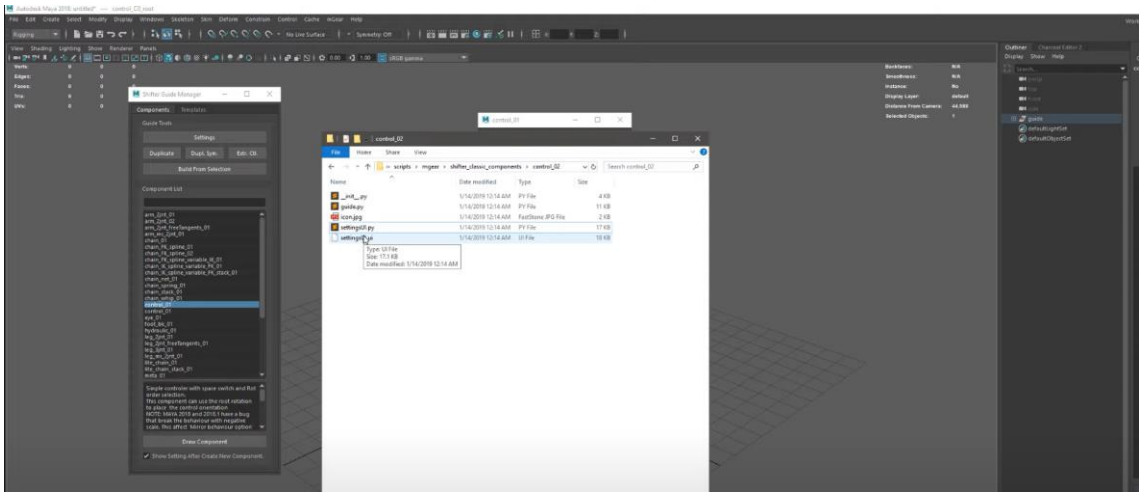
1. Tuodaan geometria sisään.
2. Avataan Shifter Components-valikko.
3. Valitaan Component, esim. *shoulder01*, muokataan asetuksia, suljetaan valikko.
4. Shifter muodostaa komponentille visuaalisen guiden.
5. Guide asetellaan haluttuun asentoon.
6. Valitaan uusi Component esim. *arm_2jnt_01*, säädetään asetuksia ja suljetaan valikko.
7. Viedään syntynyt arm, shoulder-komponentin alle.
8. Valitaan koko Guide ja rakennetaan rigi testiksi painikkeella ”Build from Selected”
9. Testataan rigin käyttäytymistä esim. testianimaatiolla.
10. Poistetaan testattu rigi, jotta scene pysyy puhtaana.
11. Muokataan asetuksia, vaihdetaan tai muokataan komponentteja, muutetaan guiden asentoa yms.
12. Rakennetaan rigi ja testataan uudelleen.
13. Vaiheita 10-12 toistetaan, kunnes ollaan tyytyväisiä tulokseen.

8 Omien moduulien luominen

Typistetysti moduulin luominen menee näin; kopioidaan valmis component, avataan se sekä kolmiulotteisena Mayassa, että sen Component Guide-osat Python-muodossa tekstieditorissa, esimerkiksi Sublime textissä tai ohjelmointiympäristössä, kuten pyCharmissa. Visuaalista guidea testataan rakentamalla sen pohjalta rigi, tehdään muutoksia sen käyttämiin Python-luokkiin ja tallennetaan muutokset. Muutokset ladataan reload components-komennolla, luodaan uusi rigi ja toistetaan niin kauan, kunnes komponentista syntyneet rigit vastaavat haluttua. (Kuva 26; Campos 2019.)

Kehittämiseen liittyvän ongelmanratkaisun apuna voi lukea valmiita moduuleita, näiden sisältämiä luokkia, pyMEL-dokumentaatiota ja esimerkkejä, pikahakua esimerkiksi mGear-projektien avoimista gitHubeista, sekä lukemalla ja olemalla aktiivinen mGear-foorumeilla.

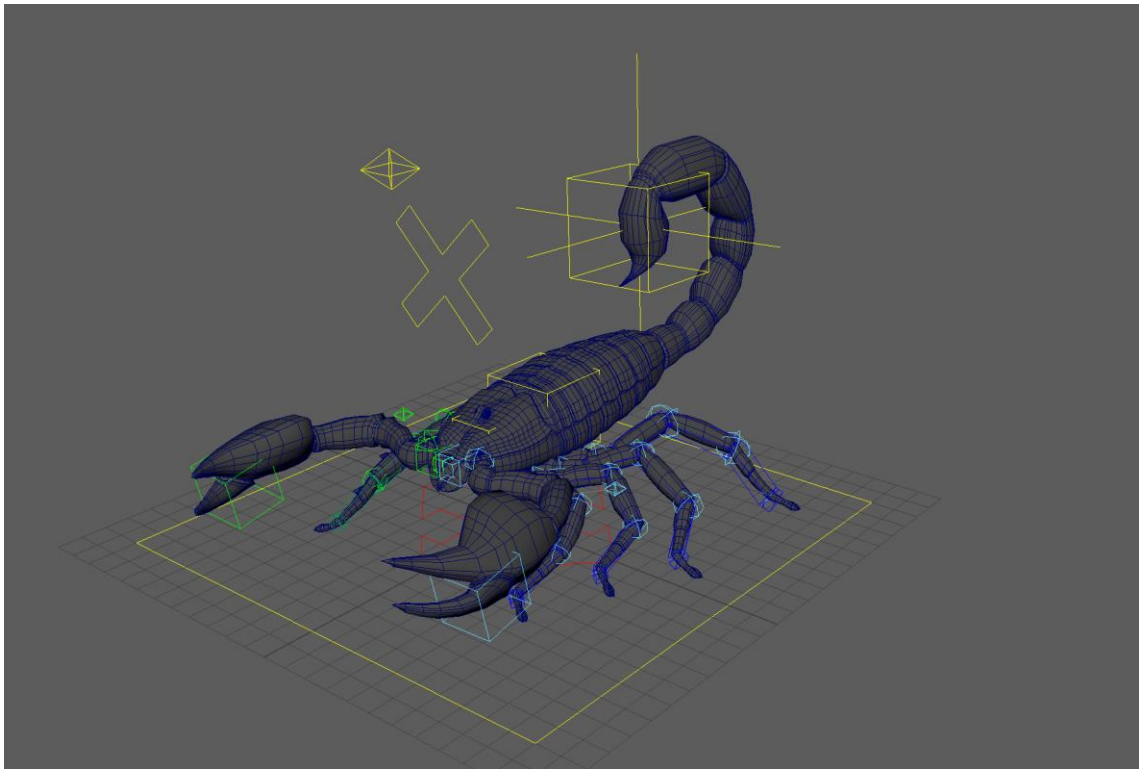
Sekä tallennetut moduulit, että valmiiksi kootut pohjat ovat siis itseasiassa Python-moduuleja ja luokkia, joiden luonnin mGear on tehnyt puoliautomaattisiksi. Moduulin eli Shifter-termein komponentin tallentamista skriptiksi ja luomiseksi uudelleen skriptistä kutsutaan I/O-, eli input-output-testaukseksi. (Fragapane 2018).



Kuva 26. mGear-kehittäjä Miguel Campos editoimassa Control01-komponenttia (Campos 2019).

9 Projektina modulaarinen Skorpioni

Seuraavaksi esittelen opinnäytetyötä varten tekemääni modulaarista rigiä, jonka avulla yritin selvittää omakohtaisesti modulaarisuuden hyviä ja huonoja puolia, sekä opiskella Shifterin käyttöä lisää (Kuva 27).



Kuva 27. Kuva projektin keskivaiheilta (Liite 2, 2/2).

9.1 Kohteen valinta

Tutkittuani modulaarista rigausta kiinnostuin hyönteisistä, sillä niissä esiintyy luonnostaan paljon modulaarisuutta. Lopulta kokeiltuani muutamia malleja, päädyin tähän kyseiseen Skorpioniin, sillä siinä oli sopivasti haastetta ja variaatiota. Oli myös mielenkiintoista nähdä, kuinka lähinnä ihmismäisille hahmoille suunnatut moduulit soveltuisivat hämähäkkieläimelle.

9.2 Projektin tavoitteet ja ominaisuudet

Päätin että rigillä olisi kolme tavoitetta; syvempi tutustuminen Shifteriin, käytännön kokemus modulaarisen rigin toteutuksesta ja rigin tulisi olla kokonaan uudelleen rakennettavissa. Tuloksena syntyisi modulaarista rigauksen aloittamista helpottava opinnäytetyö,

jollaisen olisin itse halunnut lukea tutustuessani aiheeseen. Koska olin aiemmin tehnyt mGearillä lähinnä ei-modulaarisia rigejä, päätin kokeilla läpi kaikki valmiit moduulit ja pyrkiä toteuttamaan rigin niillä mahdollisimman pitkälle, sekä muokkaamaan niitä tarvittaessa.

Halusin rigille ainakin FK/IK-jalat, FK/IK-pistimen, sekä toimivan Central of Gravity-Controllin, lisäksi haaveilin erillisestä kaksiulotteisesta käyttöliittymästä, jonka toteuttaisin mGearin uudella animpicker-työkalulla.

9.3 Haasteena toistaiseksi dokumentaation puute ja humanoidikeskeisyys

Opin nopeasti, että valtaosa Shifterin moduuleista on suunniteltu ihmismäisille hahmoille ja hyvin tietynlaista tuotantoa varten, haasteena oli pakottaa nämä moduulit toimimaan myös eläimelle, jonka ranka on hyvin erilainen. Vielä merkittävämpi haaste oli se tosiasia, että opiskelumateriaalia järjestelmän käyttöön on vähän. Vasta opinnäytetyön loppumetreillä alkoi ilmestyä lisää dokumentaatiota, jossa listattiin muutamien moduulien ominaisuuksia ja kehitystä syvemmin (Ragtag 2020). Lisäksi mGear seuraa Softimagesta peräisin olevia termejä, jotka eroavat täysin Mayan käyttämistä termeistä. Tästä hyvä esimerkki on *blade*, jonka vastine Mayassa olisi pole vector. Kehittäjät onneksi tuottavat kuukausittain uusia Shifter-aiheisia videoita ja ovat hyvin aktiivisia vastaamaan mGear-foorumilla esitettyihin kysymyksiin. Valitettavasti videoiden laatu vaihtelee paljon ja niiden läpikahlaamiseen menee merkittävästi aikaa. Tilannetta kuvastaa se, että alussa päädyin käännättämään myös japanin- ja venäjänkielistä materiaalia ymmärtääkseni järjestelmää paremmin. Kolmas tapa opiskella järjestelmää on lukea ja muokata sen lähdekoodia, joka oli aluksi kaoottista, mutta lopulta paljon palkitsevampaa, kuin videoiden passiivinen katselu.

9.4 Rikkautena uusia ratkaisuja

Shifterin hienous piilee siinä, että moduulien tekijöillä on ollut paljon kokemusta edistyneestä rigauksesta ja moduuleja on optimoitu esimerkiksi käyttämällä matriisinodeja, uudenlaisia solvereita ja puoliautomaattista nimeystä. Hyödyntämällä valmiita moduuleja tulin tutustuneeksi myös uusiin rigaustekniikoihin, joita en aiemmin tullut edes ajatelleeksi.

9.5 Moduulit apuna ja hidasteena

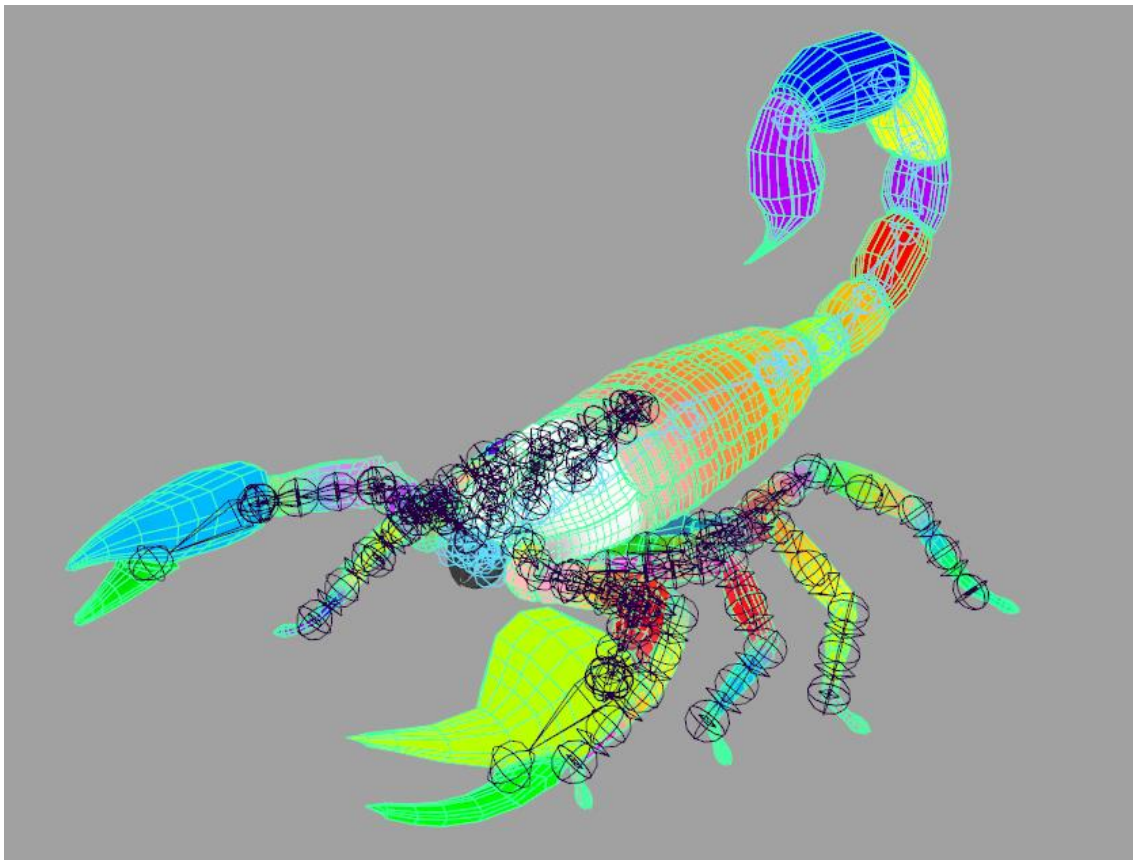
Yleisesti rigin tekeminen ja esimerkiksi erilaisten hierarkioiden kokeileminen oli erittäin nopeaa, mutta heti erikoisemman haasteen tullen koin törmääväni kuin seinään, jota oli opeteltava kiipeämään ylös. Vaikka itse järjestelmän muuttaminen on nopeaa ja tuettua, sen ymmärtäminen ja lukeminen oli haasteellista ja perustui yritykseen ja erehdykseen.
Esimerkiksi sopivan jalkamoduulin valintaan meni yllättävän paljon aikaa, sillä halusin paljon kontrollia isolle määrälle niveliä, jotka eivät tässä mallissa ole aivan suorassa linjassa millään akselilla. Tämä taas aiheutti helposti ongelman Shifter-moduulien solvereille, jotka tarvitsevat yhden suoran akselin, jotta liikkeen laskeminen onnistuisi Koska tietoa moduuleista on vähän, päädyin kokeilemaan erilaiset ratkaisut käsin läpi.

Lopulta päädyin käyttämään uutta kolmen nivelen jalkamoduulia, jossa on vaihtoehtona kolmijalkaisille eläimille suunnattu Spring Solver, käytännössä sen idea on jakaa translaation saavuttamiseksi tehtävä rotaatio mahdollisimman tasan nivelten kesken. Kyseinen solver on ollut tavallisessa Mayassa jo jonkin aikaa, mutta sitä näkee vähän, koska työkalulle ei ole ollenkaan omaa käyttöliittymää ja se pitää luoda skriptaamalla.
Jos tämä rigi menisi johonkin tuotaantoon asti, olisi seuraava askel luoda oma, itselle sopivampi moduuli, jolle voisi vapaasti määrittää jointtien määrän, solverin ja kenties tehdä siistityn käyttöliittymän, jolla voisi ohjata esimerkiksi solverin jäykkyyttä paremmin eri nivelten välillä.
Myös Soft IK:ta, eli pehmeästi ääriasentoon taipuvaa IK-ketjua hyödyntävä moduuli olisi hienoa toteuttaa tulevaisuudessa itse.

9.6 Shifter & automaattinen skinnaus

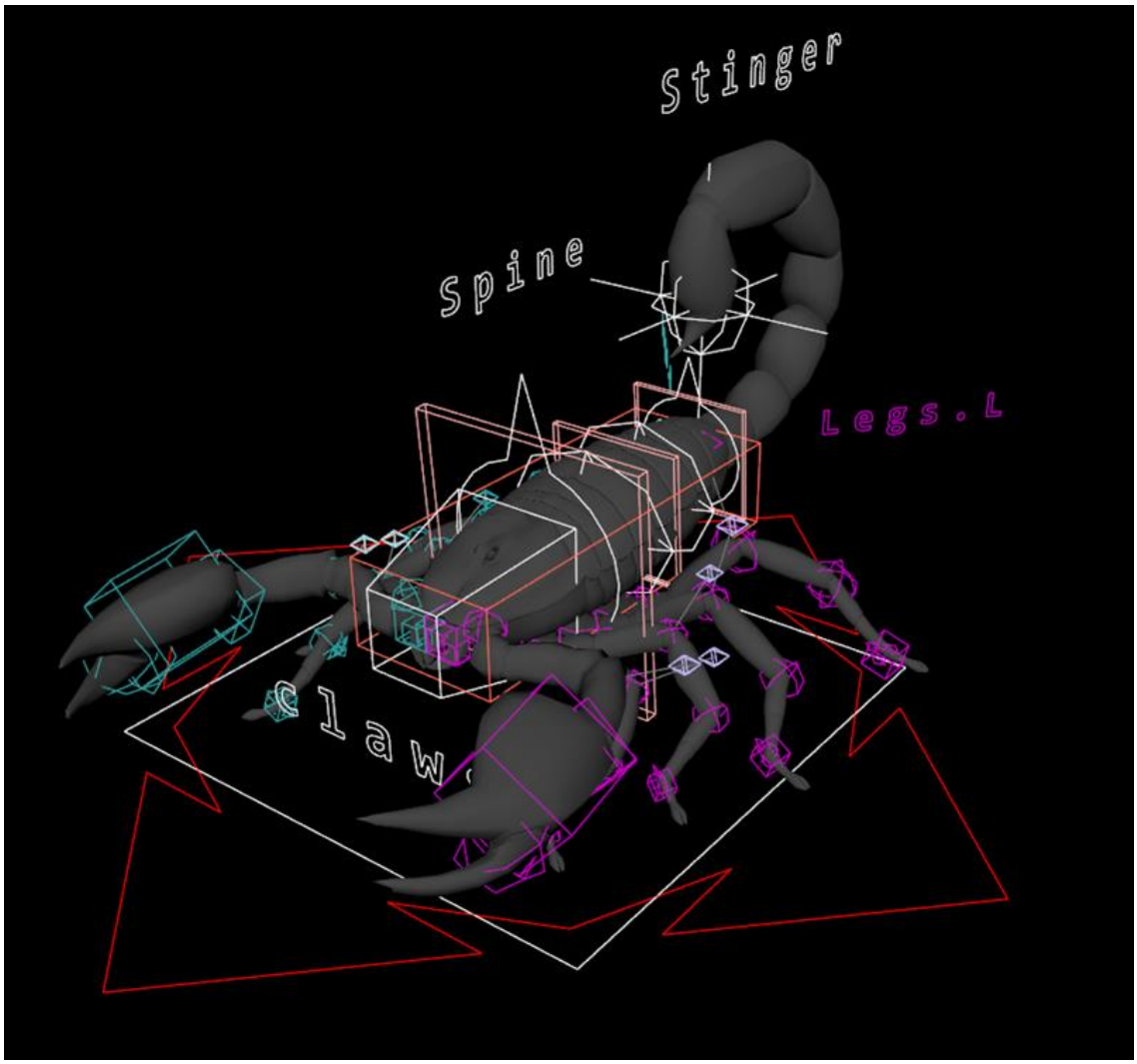
Skinnaus, eng. skinning on työvaihe, jossa geometria sidotaan liikkuviin jointteihin. Hyvä ja sulavasti taipuva skinnaus vie paljon aikaa ja tietotaitoa ja käsittelen sitä lyhyesti, säästääkseni lukijan aikaa tulevaisuudessa.
Koska rigaaminen Shifterillä on hyvin iteratiivista, eli rigi syntyy ja tuhoutuu usein, olisi raskasta lähteä lähteä käsin skinnaamaan moduuleja, joita ei välttämättä lopulta edes käytä ja jonka lopputulos tuhoutuisi helposti. Itse ratkaisin asian seuraavalla tavalla. Jaoin mallin ensin taipuviin osiin, esimerkiksi jalat ja keskiosa erikseen, annoin Mayan automaatti-skinnaajan tehdä testiskinnauksen jokaiseen osaan erikseen, jolloin esim raajat eivät sulautuneet toisiinsa, eikä niitä tarvinnut siistiä niin paljon (Kuva 28). Yhdistin

osat ja näiden skinit takaisin toisiinsa huippurigaaja Perry Lejtenin ilmaisella Skinning-työkalulla.(Lejten 2020). Avasin syntyneen skinclusterin Victor Makauskas:in luomassa ngSkinning-työkalussa, tein skinnauksesta kopion, jota peilasin ja siistin vähän. (Makauskas 2020). Lopuksi tallensin skinin mGearin omalla skinning-työkalulla gskinning-formaattiin ja toin sen takaisin alkuperäiseen malliin. Tein ylätason guidessa asetukseksi valinnan, jonka mukaan guidesta syntyvän rigin tulee aina rakentumisen jälkeen ladata kyseinen gskin. Näin sain nopeutettua testausta huomattavasti.



Kuva 28. Havainnoiva kuva rigin jointeista ja influenceista, jokainen väri edustaa yhtä influencea (Liite 2, 3/3)

10 Projektin lopputulos



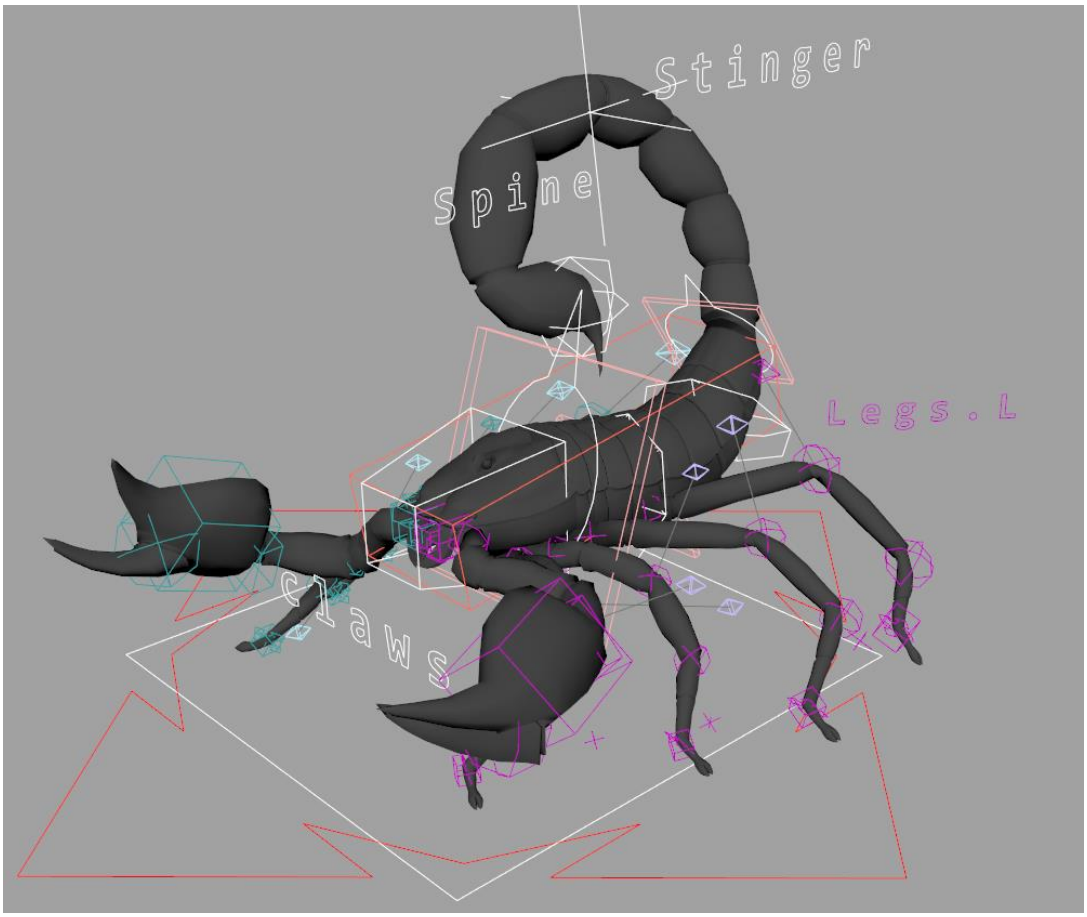
Kuva 29. Projektin tavoitteisiin nähden valmis rigi (Liite 2, 3/3).

Lopputuloksena syntyi modulaarinen skorpioni, jolla pitäisi teoriassa pystyä tekemään erilaisia kävely- ja hyökkausanimaatioita, joskaan en pitäisi sitä vielä mihinkään produktioon valmiina, koska esimerkiksi sen skinning vaatisi vielä siistimistä (Kuva 29). Se kuitenkin demonstroi itselleni hyvin kaikenlaisia ongelmia ja ratkaisuja liittyen Shifter-rigaamiseen ja siksi olenkin tyytyväinen. Seuraavaksi listaan valmiin rigin osat demonstroidakseni rigissä tekemiäni valintoja. Totesin myös mgearin uuden animpicker-työkalun toimivaksi ja teinkin sillä alkeellisen käyttöliittymän, jonka lopulta tiputin pois ylimääräisenä ominaisuutena projektista, sillä se vei aikaa itse opinnäytetyön kirjoittamiselta.

10.1 Valmiin työn osat ja komponentit

Valmiissa rigissä on seuraavanlainen hierarkia, listassa ensin yksittäisen osan tarkoitus, osan nimi hierarkiassa, sen jälkeen toteutukseen käytetty Shifter-moduulin nimi. Koska moduulit löytyvät kaikki Shifteristä valmiina, on tämä myös vähän kuin kauppalista vastaavan rigin tekemiseen.

- 1 kpl Ylätason kontrolli, world, Control01
- 1 kpl Center of Gravity-kontrolli,COG, Control01
- 1 kpl rintakehä, Chest, Control01
- 2 kpl piensaksia, Miniclaw, Control01
- 1 kpl selkäranka, Spine, Spine FK
- 1 kpl pistin, Stinger, Chain01
- 2 kpl käsivarsia, arm, arm_2jnt_freeTangents_01
- 2 kpl saksia, Claw, Control_01
- 8 kpl lonkkia, hip, Control_01
- 8 kpl kolmen nivelen jalkoja,leg, leg_3jnt_01
- 4 kpl kontrolleja käyttöliittymille, Uihost, Control_01



Kuva 30. Rigi toiminnassa

10.2 Merkittävimmät ominaisuudet

- FK/IK-jalat, Spring-solverilla, joka sopii erityisesti kolminivelisille jaloille, sillä se jakaa taipumiskulmat automaattisesti tasaisesti nivelten kesken.
- FK/IK Sakset
- Center of Gravity-järjestelmä, kaikki raajat ja pistin voidaan määrittää joko seuraamaan tai pysymään paikoillaan ruumiin heiluessa.
- Squash and stretch FK+IK-selkäranka
- Squash and stretch FK/IK-pistinketju

11 Kannattaako mGearia opetella?

Projektin jälkeen voisin kuvailla mGearia kaksiteräiseksi miekaksi. Sen edistyneet ominaisuudet mahdollistavat nopeaa rigausta edistyneillä ominaisuuksilla, mutta en laskisi esimerkiksi freelance-projekteja pelkän mGearin varaan, ellei siitä ole jo paljon kokemusta ennestään. Opettelu vie paljon aikaa ja on prosessina helposti turhauttava. Ideaalitilaneessa Classic Components-kirjastoa voi pitää esimerkkeinä siitä miten moduuleja voi ohjelmoida, eikä sen ole tarkoituskaan kattaa itsessään käyttäjän kaikkia rigaustarpeita, vaan käyttäjän on todella sukellettava järjestelmään ottaakseen siitä kaiken irti. Kuitenkin geneeristen humanoidien autorigaana mGear on hyvin kilpailykyinen ja näen että tässä käytössä sen soisi yleistyvän enemmänkin. Itse aion opetella mGearia enemmän ja kenties jatkaa sen nykyiseltään vähäistä dokumentaatiota, jota kuitenkin kehitetään jatkuvasti. Parhaimmillaan olisi mukava julkaista yleishyödyllinen moduuli vapaaseen käyttöön. Kokonaisen opinnäytetyön jälkeenkin, koen että oma polkuni aiheen parissa on vasta alussa.

12 Yhteenveto – milloin modulaarisuus kannattaa?

Modulaarisuus lähtee kahdesta ajatuksesta; ongelmat voidaan jakaa pienempiin osiin, jolloin niiden ratkaiseminen helpottuu ja syntyneistä ratkaisuista voidaan luoda palikoita, joita yhdistelemällä syntyy jälleen hyvin monimutkaisia, mutta kuitenkin ylläpidettävissä olevia kokonaisuuksia.

Laadukkaiden moduulien suunnittelu vaatii paljon tietotaitoa ja iteratiivista työskentelyä, eli kokemusta ja aikaa, jonka vuoksi sitä näkee lähinnä isoissa tuotannoissa. Opiskelun alussa saman työvaiheen toistaminen on tärkeää muistamisen lisäksi siksi, että se altistaa inhimillisille virheille ja opettaa korjaamaan niitä, siksi en suosittele edistyneempiä moduuleja aivan opiskelun alussa, joskin kannattaa harjoittaa mahdollisimman paljon kopiointia ja peilausta, jotta aiemmin mainittu DRY- periaate tulee tutuksi. Lisäksi modulaariset työkalut ovat usein niin monimutkaisia, että ne voivat hämmentää ja jopa haitata perusasioiden hahmottamista.

Jos rigien tai rigaajien määrä kasvaa on jo perusteltua käyttää aikaa edes muutamiin moduuleihin, joita jakamalla säästetty aika moninkertaistuu käyttäjien määrällä. On myös hyvä huomata, että edistyneen kehysohjelman käyttö ei ole pakollista sovellettaessa moduuliajattelua rigaukseen, sillä alkeellinen moduuli voi olla myös erikseen tallennettu rigin

osa, joskin tällainen ratkaisu vaatii huomattavasti enemmän manuaalista työtä ja tehdyt muutokset ovat tällöin tuhoisampia.

Moduulien kehittäminen kysyy resursseja, mutta alkeellinenkin moduuli nopeuttaa työskentelyä ja tuo rigeihin automaattisesti standardeja, mikä helpottaa sekä ylläpitoa, että opettamista uusille rigaajille ja animaattoreille. Suosittelenkin modulaarisuuden tuomista omaan työskentelyyn ja erityisesti tuotantoihin asteittain, sillä tuotettavien ja ylläpidettävien rigien määrän ja vaatimustason kasvaessa on modulaarisuus suorastaan välttämätöntä pärjätäkseen markkinoilla, oli kyseessä sitten freelancer tai studio.

Jos taas kyseessä on enintään muutaman kuukauden projekti, eikä rigausta ole ajatus jatkaa eteenpäin, on uniikki ja destruktiivinen työskentely varmasti nopeampaa ja vapaampaa verrattuna modulaarisen järjestelmän vaatimaan aikaan ja opiskeluun. Olen kuitenkin vakuuttunut, että jos rigaustyö jatkuu esimerkiksi vuoden tai pitempään, alkaa modulaarisuus maksaa itseään takaisin sekä nopeammin syntyvinä rigeinä, että yleisen teknisen osaamisen lisääntymisenä, jolle on merkittävästi kysyntää muuten kilpaillulla 3D-median alalla.

Opinnäytteen suurimpana löytönä pidän kahta jatkumoa: manuaalisesta automaattiseen ja ilmaisusta tekniseen. Erinomainen modulaarinen rigi liikkuu kummallakin janalla päästä päähän, eikä tässä ole mitään ristiriitaa. Parhaimmillaan automatiikka tuo lisää aikaa käsityölle ja teknologia avaa kokoajan uusia ovia luovalle ilmaisulle. Myös teknologian kehittäminen itsessään on luova prosessi ja toisaalta ilman hyviä tarinoita ja designeja, meillä ei olisi yhtä hyvää tekosyytä ratkaista kiehtovia ongelmia.

Lähteet

Alithographica 2020. Science Fact Friday. Kuvajulkaisu, Deviantart. https://www.deviantart.com/alithographica/art/Science-Fact-Friday-Homology-670901312

Animation Studios 2020. Advanced Skeleton 5. Työkalun kotisivut. https://www.animationstudios.com.au/advanced-skeleton (katsottu 7.5 2020)

Anzovin, Raf 2018. Rigs are software. Blogi. https://www.justtodosomethingbad.com/blog/2018/4/16/rigs-are-programs (luettu 12.5 2019)

Anzovin, Raf; Campos, Miguel,Fragapane Rafaele 2018.”Rigs as Software”-keskustelu, https://youtu.be/LF8CcKqlQqg?t=480 (katsottu 28.5)

Assaf, Eyal 2015. Rigging for Games: A Primer for Technical Artists Using Maya and Python. Yhdysvallat: CRC press.
Kirja. (Luettu 13.5 2019)

Autodesk 2014. Pole Vector constraints. Verkkoartikkeli. https://knowledge.autodesk.com/support/maya/learn-explore/caas/CloudHelp/cloudhelp/2015/ENU/Maya/files/CSCo-Pole-Vector-constraints-htm.html (luettu 7.5 2020)

Autodesk 2018. File referencing. Verkkoartikkeli. https://knowledge.autodesk.com/support/maya/learn-explore/caas/CloudHelp/cloudhelp/2018/ENU/Maya-ManagingScenes/files/GUID-238914C9-4129-454C-99D7-B8C57AF423DB-htm.html (luettu 7.5 2020)

Baldwig, Young Carliss & White, L. William 2000. Design Rules: The power of modularity. Sivu 63. (luettu 2.3 2020)

Berger, Jan 2013. Rigging Dojo Live :Modular Character System (MCS) under the hood with Jan Berger.Video. https://www.youtube.com/watch?v=px0Pc8fLrsY (katsottu 28.3 2020)

Calvert, Daniel 2018. Game Developer’s Conference: Repopulating the Earth: Character Production on Horizon Zero Dawn. Luento. https://youtu.be/2QRCwY382ck?t=2949 (katsottu 5.4.2020)

Campos, Miguel 2017. Shifter Component: arm_2jnt_01. Video
mGear: Rigging Framework-kanava. Youtube Kohta 10:31.
https://youtu.be/VDseV5fdjzw?t=631 (katsottu 12.5 2019)

Campos, Miguel 2018. Data Centric rigging workshop. Videosoittolista.
mGear: Rigging Framework-kanava. Youtube
https://www.youtube.com/watch?v=pjPzCBN5KB0&list=PL9LaIDCCDjfimQVcMdh0rG0MPabPG9FK-

Campos, Miguel 2019. mGear 3.0: Shifter New Component: Chain IK variable FK number. Opetusvideo.
https://www.youtube.com/watch?v=bkd7QXd3aV0 (katsottu 28.4 2020)

Campos, Miguel 2019. Rigging Framework Live Stream. Stream-tallenne.
mGear: Rigging Framework-kanava. Youtube.
https://www.youtube.com/watch?v=2Q_nfNZkNlo (katsottu 1.3.2020)

Crouzet, Christopher 2014. Modern Rigging Workflow: The Modular Approach. Blogi.
https://zcrou.com/blog/cg/modern-rigging-workflow (luettu 12.5 2019)

Dombrova, Chad 2009. PyMEL for Maya. Dokumentaatio.
https://help.autodesk.com/cloudhelp/2018/JPN/Maya-Tech-Docs/PyMel/index.html

Fragapane, Rafaele 2018. Modular Components and Rig Assembly, Cult of Rig. Blogi.
http://www.cultofrig.com/2018/06/30/season-01-episode-37-modular-components-reassembly/ (luettu 12.5 2018)

Fragapane, Rafaele 2018. Scripting Retrieving Objects. Cult of Rig. Youtube-kanava.
https://www.youtube.com/watch?v=kY9XMrrg51Q (katsottu 19.5 2019)

Hunt, Andrew & Thomas, David 1999. The Pragmatic Programmer : From Journeyman to Master. Kirja. (luettu 12.5 2019)

Hunt, David & Sderlind, Forrest 2015. Tools-Based Rigging in Bungie's Destiny. Video.
Game Developers Conference-kanava. Youtube.
https://www.youtube.com/watch?v=U_4u0kbf-JE (katsottu 6.5 2019)

Hunt, David 2009. Modular Procedural Rigging.
Game Developers Conference 2009. Powerpoint-esitys.
http://downloads.bungie.net/presentations/ModularProceduralRigging.zip

InspirationTuts 2019. The history of Softimage. Verkkoartikkeli.
https://inspirationtuts.com/2019/12/13/the-history-of-softimage/
(luettu 7.5.2020)

Joensuu, Janne 2016. 3D-alan sanasto: 3D-grafiikan termit suomeksi. Opinnäytetyö.
http://urn.fi/URN:NBN:fi:amk-2016060612045 (luettu 7.5 2020)

Kilpeläinen, Toni 2001. 3D-sanastoa. Tivi-lehti. Verkkoartikkeli.
https://www.tivi.fi/uutiset/3d-sanastoa/ad71bb2f-b964-3358-9290-d04b926d8dc8

Kutlu, Arda 2018. T-Rigger – Tentacle Rig. Verkkosivu.
http://www.ardakutlu.com/t-rigger-tentacle-rig/ (katsottu 18.4 2020)

Lejten, Perry 2020. Maya Skinning Tools. Ilmainen työkalu Maya-rigaajille.
https://gumroad.com/peerke#peVHN (katsottu 5.5 2020)

Makauskas, Victor 2020. ngSkin Tools Project. Suosittu rigaustyökalu Maya-käyttäjille.
https://www.ngskintools.com/ (katsottu 5.5 2020)

Mgear Developers 2020. mGear Framework, Open source rigging and animation framework for Autodesk® Maya®. Verkkosivu.
http://www.mgear-framework.com/(luettu 6.5 2019)

Mgear Developers and Users 2020. mGear Framework Forum.
Keskustelufoorumi, http://forum.mgear-framework.com/ (luettu1.3.2020)

McComb, David 2016. ” The Data-Centric Revolution: Data-Centric vs. Data-Driven”
Verkkoartikkeli.
https://tdan.com/the-data-centric-revolution-data-centric-vs-data-driven/20288

Neistadt, Sir Wade 2020. Exploring the SpiderVerse: Animation Rig.
Sir Wade Neisstadt-youtubekanava.
https://www.youtube.com/watch?v=BhoM3PBMDv8 (katsottu 13.4 2020)

Nelson Dustin 2019. Rapid Rig Modular: Procedural Auto Rig. Työkalun kotisivut.
https://www.highend3d.com/maya/script/rapid-rig-modular-procedural-auto-rig-for-maya
(katsottu 7.5 2020)

O’Neilly, Rob 2015. Digital Character Development: Theory and Practice, Second Edition. Kirja. Luku 20.

Peters, Chris 2012. 3 Key Software Principles you must understand, Envato tuts+,
Verkko-opetusmateriaali.
https://code.tutsplus.com/tutorials/3-key-software-principles-you-must-understand--net-25161 (luettu 18.5 2019)

Pinsard, Loïc 2019. “Proposal: Moving Blender’s rigging to a component based workflow” Keskustelufoorumi.
https://devtalk.blender.org/t/proposal-moving-blenders-rigging-to-a-component-based-workflow/7412

Piroshi 2020. Piroshi Lego Works. Twitter-tili.
https://twitter.com/ginkyoka/status/1234106047033294848/photo/1 (katsottu 2.3 2020)

Plularsight 2014. Key 3D Rigging Tems to Get You Moving. Verkkoartikkeli. https://www.pluralsight.com/blog/film-games/key-rigging-terms-get-moving (luettu 7.5 2020)

Puppeteer Lounge 2018. Rigging workshop, Student reels. Videosoittolista, Youtube. https://www.youtube.com/playlist?list=PLHJoZPTzp309HgfMgAE4djVaMqfruFvHo (luettu 18.5 2019)

Puppeteer Lounge 2018. Bird Wing Setup | Rig Demo. Esittelyvideo, Youtube. https://www.youtube.com/watch?v=l8XRCc-LuYQ (katsottu 28.4.2020)

Ragtag 2020. “Adding Quick Start doc for shifter” Github-ohjelmavarasto. https://github.com/ragtag/mgear_dist/blob/master/docs/source/shifterComponentReference.rst (luettu 27.6 2020)

Ross, Corey 2015. Quadruped Rigging Tutorial Part 1: Ribbon Spine. Opetusvideo, Youtube https://www.youtube.com/watch?v=OlTd6LgajmI (katsottu 28.4.2020)

Shotarov, Vasil. 2017. Rigging Systems – Reusability, modularity and extensibility. Blogi. https://bindpose.com/rigging-systems-reusability-modularity-extensibility/ (luettu 6.5 2019)

Spacey, John 2016. What is Modular Design, Simplicable (luettu 1.3.2020) https://simplicable.com/new/modular-design

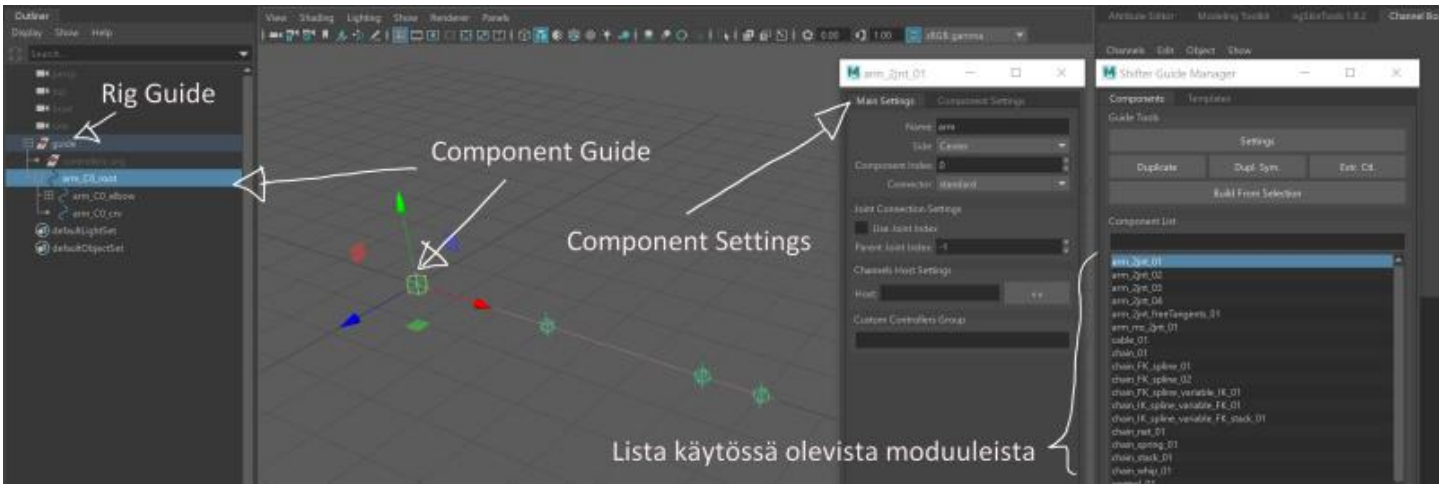
Zauner, Sofia 2017. MODULAR RIGGING SYSTEM. Verkkosivu. http://www.mattschwarz-animation.de/modular-rigging-system/ (luettu 28.3 2020)

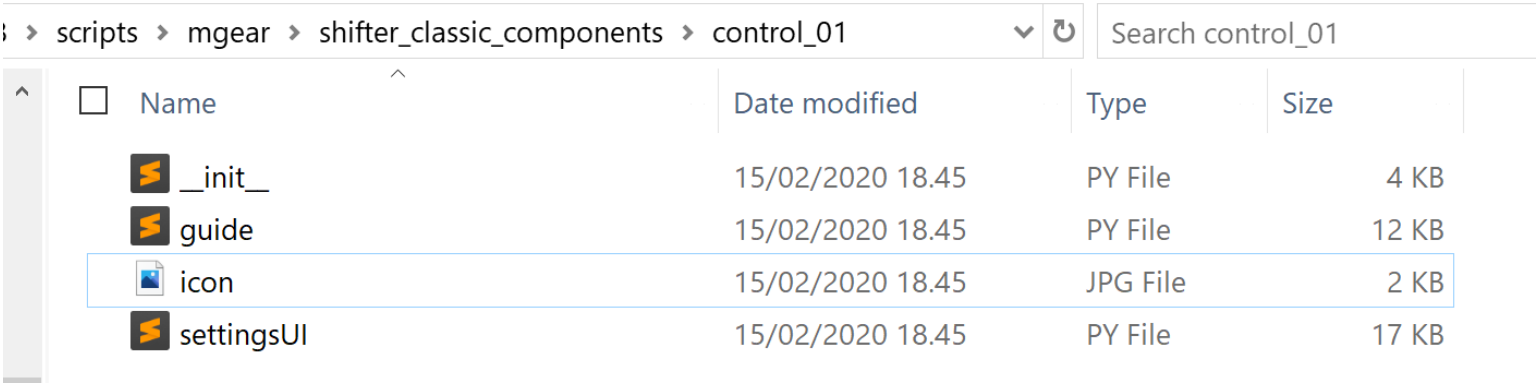
Talbot, Jeremie 2016. How Pixar created its most complex character yet for 'Finding Dory' (CNET News). Uutisvideo, Youtube https://www.youtube.com/watch?v=Nn0S2vmSCU0 (katsottu 28.4 2020)

Kasakov, Vladislav 2016. The Role of Python in Visual Effects Pipeline : Case: Talvi Tools. Opinnäytetyö. http://urn.fi/URN:NBN:fi:amk-2016100614871

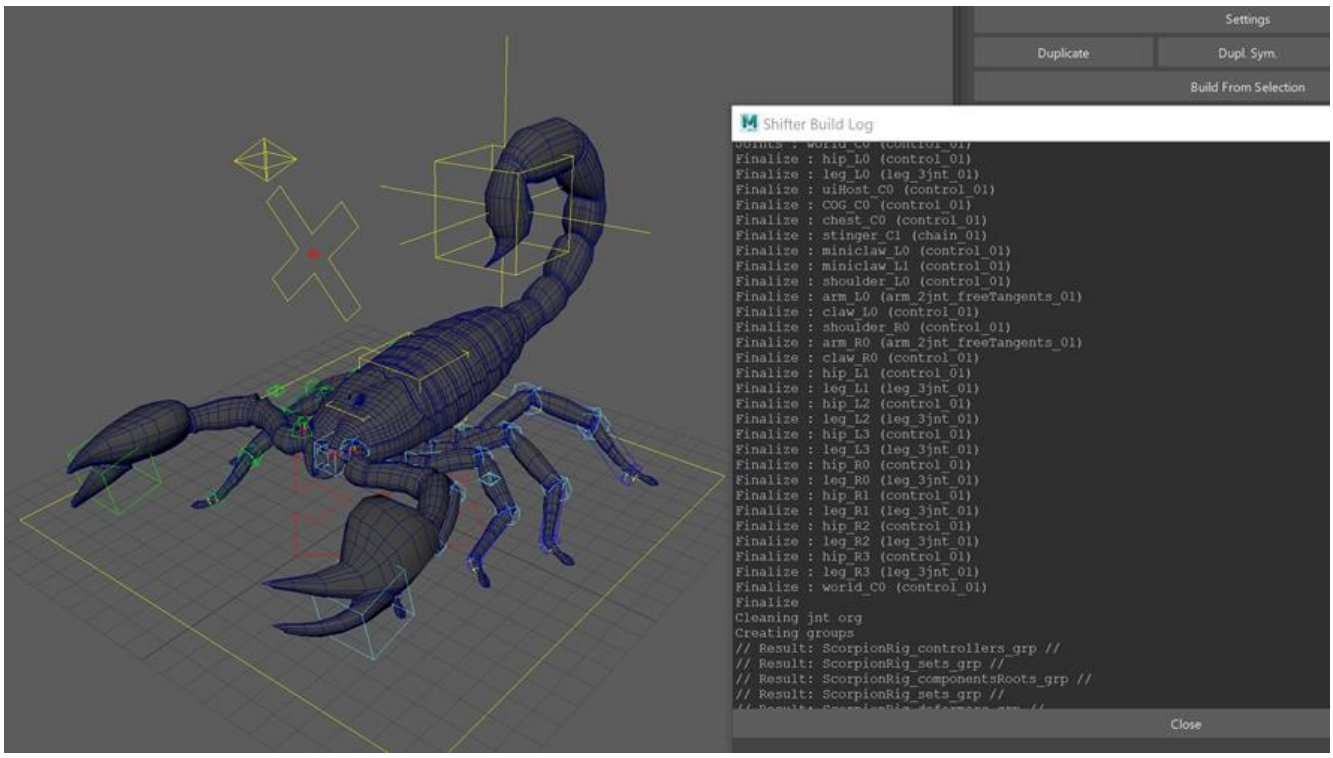
Burns, Michael 2014. The long and lonely death of Softimage. Verkkoartikkeli. https://www.digitalartsonline.co.uk/features/motion-graphics/long-lonely-death-of-softimage/ (luettu 5.5 2020)

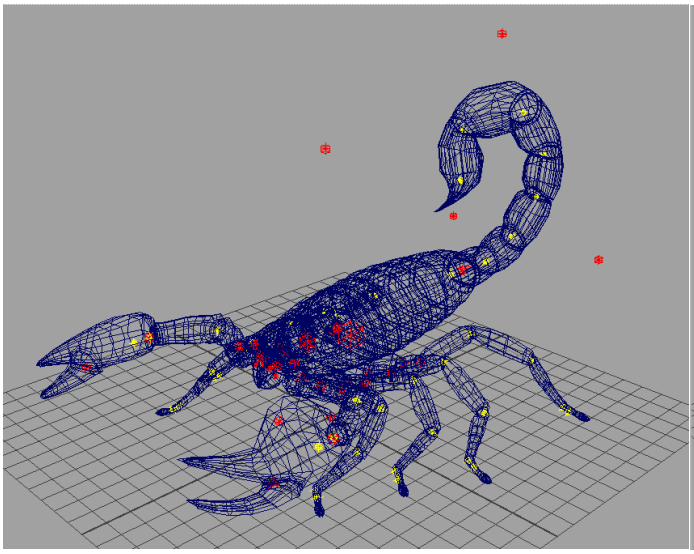
Kuvakaappauksia Shifter-ohjelmasta

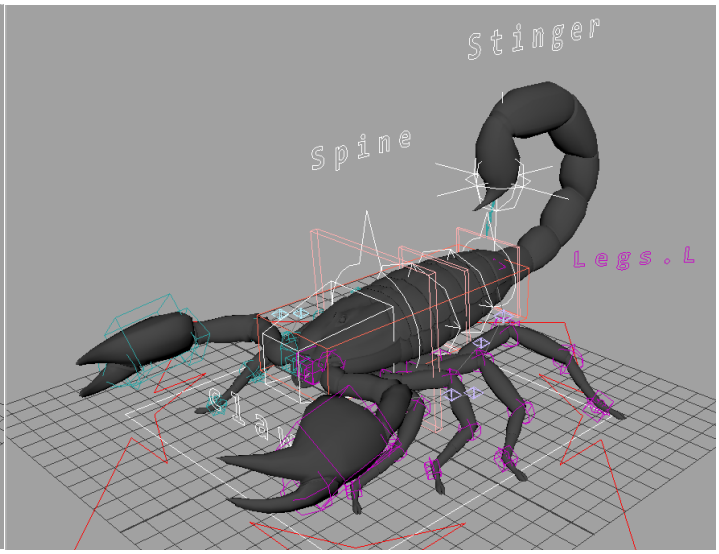


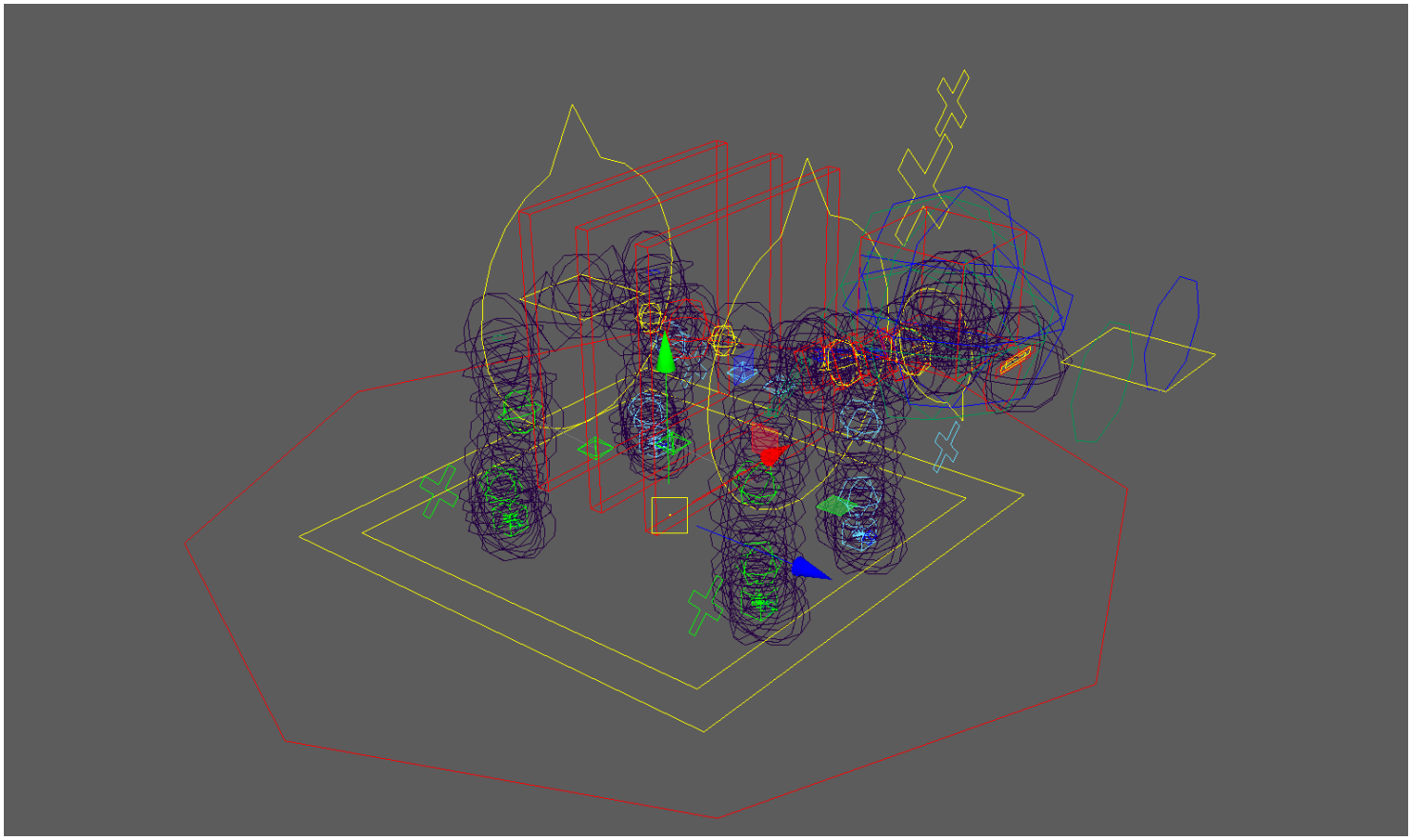


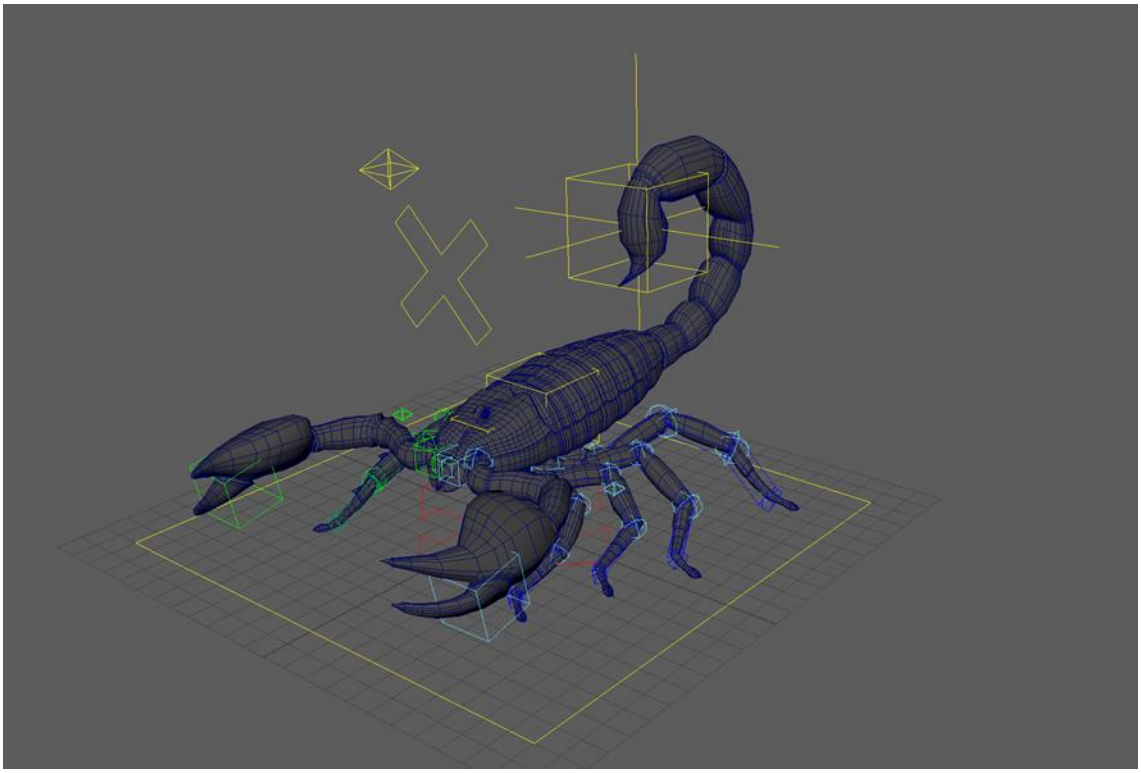
Kuvia Scorpion-rigin rakentamisesta

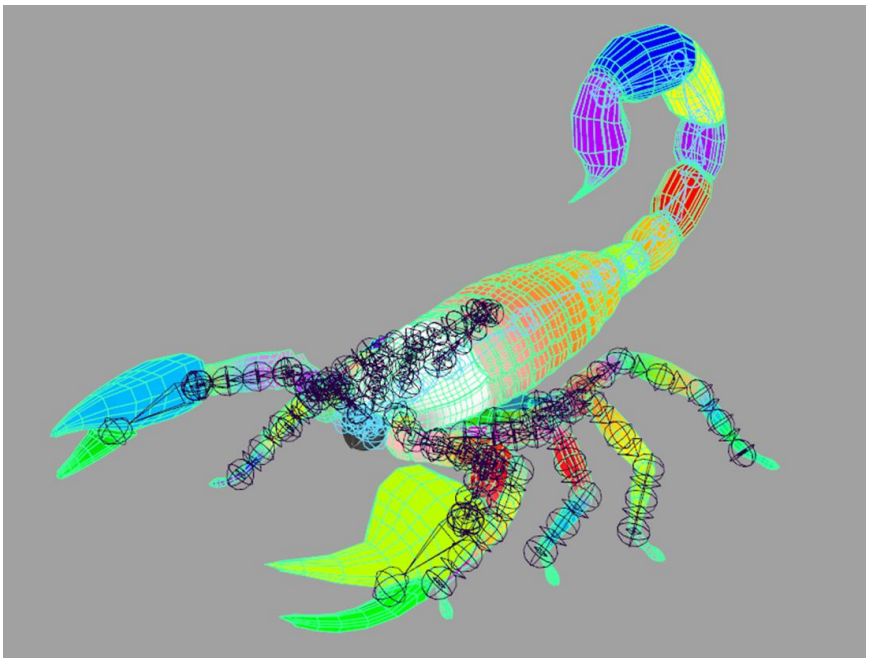


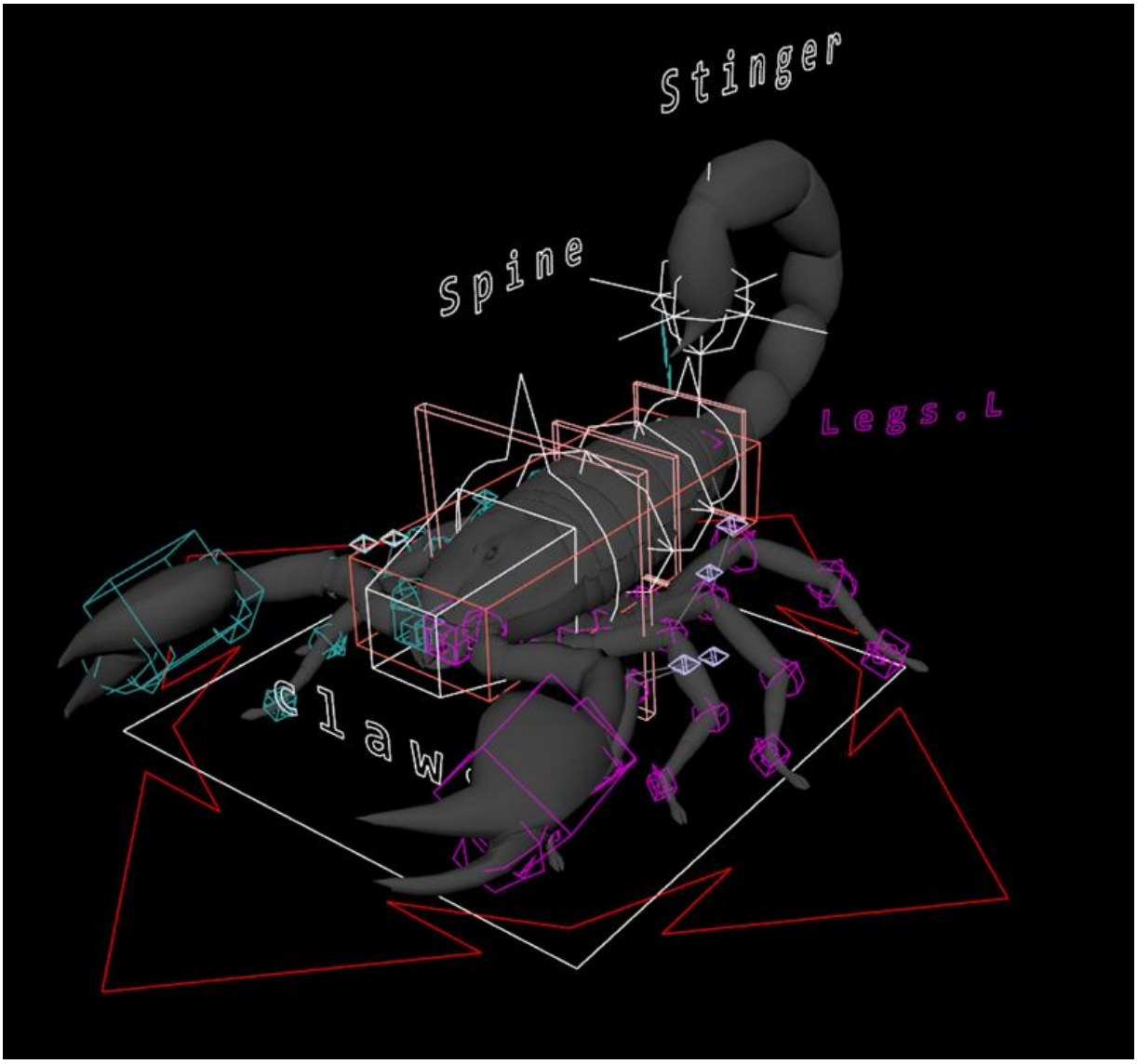


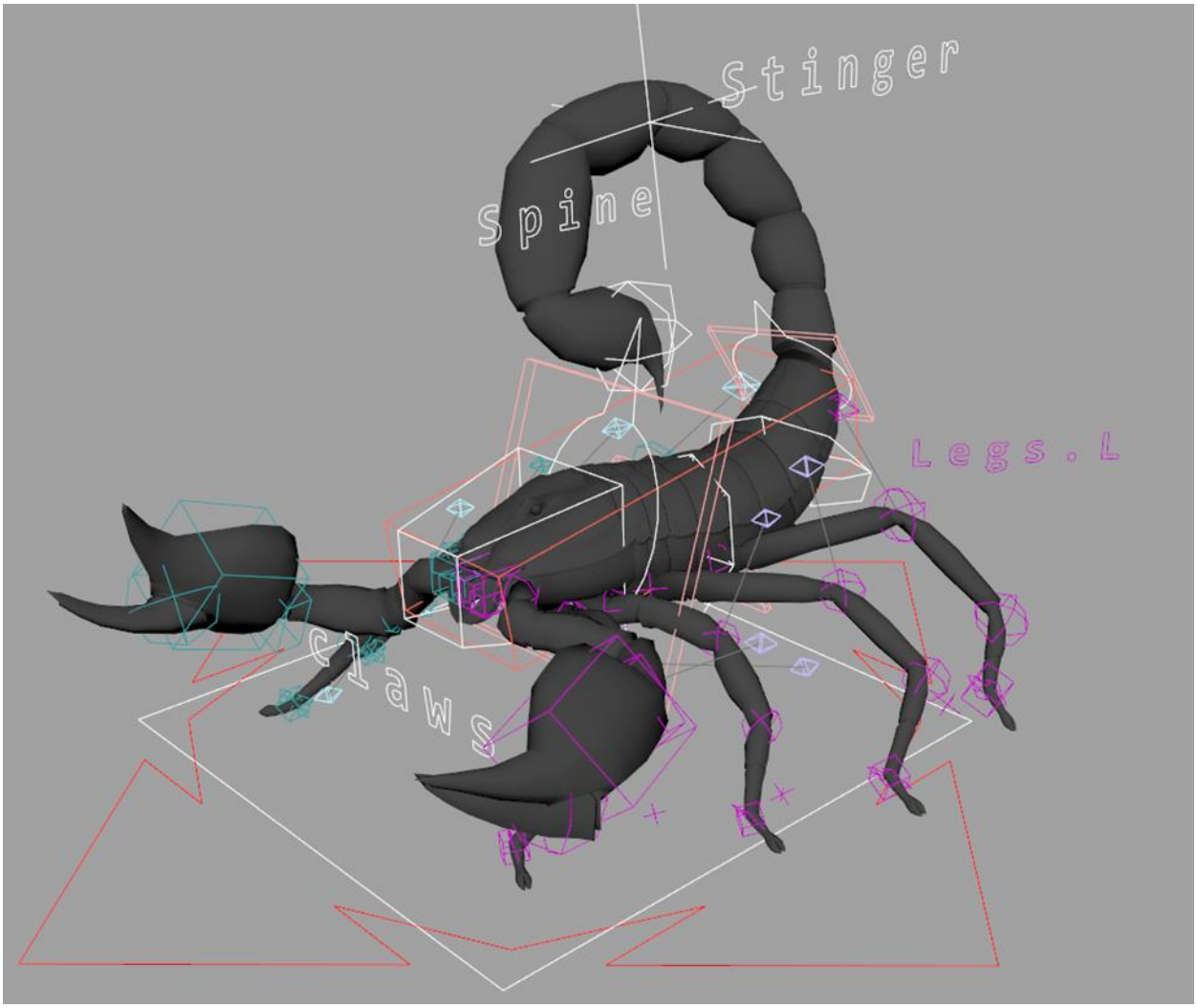









Stinger
Spine
Legs.L
Claw


stinger
spine
Legs.L
claws