Antti Herva

Vokselointi ja kudosten simulointi animaatiotuotannossa

Metropolia Ammattikorkeakoulu
Viestintä
3D-animointi ja –visualisointi
Opinnäytetyö
28.5.2012

Tekijä(t) Otsikko Sivumäärä Aika	Antti Herva Vokselointi ja kudossimulaatio animaatiotuotannossa 46 sivua + 1 liitettä 28.5.2012
Tutkinto	Viestinnän koulutusohjelma
Koulutusohjelma	3D-animointi ja visualisointi
Suuntautumisvaihtoehto	suuntautumisvaihtoehdon nimi
Ohjaaja(t)	Opettaja Kristian Simolin Opettaja Jaro Lehtonen

Tässä toiminnallisessa opinäytetyössä tutkitaan modulaarisen vokselointi- ja kudossimulointijärjestelmän kehittämistä animaatiotuotantoja varten. Järjestelmän kehitystarve syntyi Adel Abidinin "Twins"-animaatiotoimeksiannon yhteydessä. Animaatiossa kaksi sikiötä taistelee elintilasta äitinsä kohdussa verisin lopputuloksin. Toimin tuotannon toteuttaneessa Fake Graphics Oy:ssä teknisenä johtajana projektin yhteydessä.

Muuttuvat vuorovaikutussuhteet, topologiamuutokset ja tilavuuskuvantaminen ovat edelleen haasteellisia nykyisillä monikulmioverkkoihin ja staattisiin hierarkioihin perustuvilla kaupallisilla kolmiulotteisen animaation tuotanto-ohjelmilla. Näin ollen oli perusteltua laajentaa kehitystyö kohden uudelleenkäytettävää järjestelmää, joka helpottaisi tuotantotyötä myös tulevissa samankaltaisissa tuotannoissa.

Opinnäytetyössä esitellään järjestelmän kehitystyö ja perustellaan tehdyt valinnat. Järjestelmä on toteutettu visuaalisella ohjelmoinnilla teknisin, taloudellisin ja tuotannollisin perusteluin valitussa Autodesk Softimage-ohjelmiston Interactive Creative Environmentissa (ICE). Koska Fake Graphicsin tuotantolinjan painopiste on hiljattain siirretty Softimage-ohjelmistolle, oli ICE soveltaminen luonteva vaihtoehto.. Järjestelmä on silti refaktoroitavissa eri ohjelmistoille tai jatkokehittämällä omana ohjelmanaan.

Yhteenvetona voidaan todeta, että tutkituista vaihtoehtoisista totetustavoista kokonaisvaltaisesti pistepilviin perustuva järjestelmä tarjosi parhaan uudelleenkäytettävyyden ja käyttäjäystävällisyyden suhteessa hybridiin monikulmioverkkoja ja pistepilviä yhdistelevään järjestelmään. Järjestelmälle ilmeni myös monia optimointi- ja jatkokehittämismahdollisuuksia.

Avainsanat	Vokseli, Animaatio, Simulointi, Pistepilvi, Ohjelmointi

Author(s) Title Number of Pages Date	Antti Herva Voxelization and Tissue Simulation in an Animation Production 46 pages +1 appendice 28 May 2012
Degree	Bachelor of Culture and Arts
Degree Programme	Degree programme in Media
Specialisation option	3D animation and visualization
Instructor(s)	Kristian Simolin, Lecturer Jaro Lehtonen, Lecturer

The objective of this thesis was to develop a reusable system for topology independent voxel based tissue simulations for use in 3D animation. The need for this system rose from the requirements of an animation commisioned by the artist Adel Abidin called "Twins". The animation features two rival fetuses struggling for space in their mothers womb with gory consequences. I worked as the Technical Director of the project at Fake Graphics Oy.

Variable dependencies, changing topologies and volume rendering are still somewhat challenging to accomplish feasibly with modern commercial 3D animation software. Thus it was justified to expand the task at hand to the development of a modular and reusable system for use in future productions.

The development and reasoning behind the system is thoroughly explained in this thesis. The system was developed using visual programming in the Interactive Creative Environment (ICE) of Autodesk's Softimage 3D animation software. Softimage was found to provide the best overall technical, financial and production efficiency given Fake Graphics' current animation pipeline. However the system can be refactored for other software or even as a stand alone application with some additional effort.

In conclusion it was found that the approach based entirely on manipulating pointclouds was superior to the hybrid approach of isosurfacing meshes and pointclouds both in terms of reusability and ease of use. In addition many ideas for possible further development and optimization of the system were conceived.

Keywords	Voxel, Animation, Simulation, Point Cloud, Programming

Sisällys

1 Johdanto

Tietokoneiden laskentatehon alati kasvaessa laadukkaat tekniset ratkaisut kolmiulotteisissa animaatiotuotannoissa löytyvät yhä useammin todellisuuden tarkasta matemaattisesta mallintamisesta ovelien tietokonegraafisten tekniikoiden ja käsityöllä saavutettujen huijausten sijasta. Tarve huijauksille silti tuskin poistuu, sillä mielikuvitus ja todellisuus eivät nykytiedon puitteissa tunne suoranaisia laskennallisia rajoja. Toimeksianto, joka on tämän opinäytetyön teososa, on eräänlainen esimerkki siitä, millä tavoin on mahdollista tavoitella realismia animaatiotuotannossa nykypäivänä. Toimeksianto oli Adel Abidinin Abu Dhabin näyttelyä varten toteutettu kaksiminuuttinen kolmiulotteinen animaatio kahdesta sikiöstä, jotka taistelevat elintilasta äitinsä kohdussa. Olin mukana Fake Graphics oy:ssä teknisenä johtajana projektin toteuttaneessa työryhmässä.

Toimeksiannon visuaalinen ilme on mukaelma ihmiskohdun lääketieteellisestä kuvantamisesta, jota varten tilavuusdatan visualisoinnin tekniset keinot, kuten vokselit, on alun perin kehitetty. Opinäytetyössäni esittelen kehittämäni järjestelmän mielivaltaisten monikulmioverkkojen vokselointia varten sekä järjestelmän käyttöä vokselipohjaisin kudossimulaatioihin. Vokseloinnilla on mahdollista saavuttaa suurempi visuaalisen realismin aste, vaan se tarjosi myös ratkaisun muuttuvan topologian perinteiseen ongelmaan monikulmioverkkoihin pohjautuvassa kolmiuloitteisessa animaatiotuotannossa.

Toisena suurena hyötynä ratkaisuissa, jotka pohjautuvat liitännäisen tai lisätyökalun kehittämiseen on moduulien, tai peräti koko järjestelmän, uudelleenkäytettävyys muissa tuotannoissa. Tällöin saavutetaan paremmat katteet tulevia toimeksiantoja toteutettaessa, jotka käsittelevät samaa aihealuetta tai muutoin vaativat vastaavanlaisia tekniikoita.

Olen jakanut opinnäytetyöni selkeisiin vaiheisiin, joissa käsittelen kutakin osa-aluetta toimeksiannosta aina toteuttamiini järjestelmiin asti. Aluksi myös selvennän joitain opinäytetyöni kannalta olennaisia käsitteitä. Käyn katsausluontoisesti läpi

toimeksiannon eri vaiheet, mikä osuuteni projektissa oli sekä muiden projektiin osallistuneiden työpanoksen.

Pääpaino opinäytetyössäni on järjestelmissä jotka olen toteuttanut Autodesk Softimage-ohjelmiston ICE:llä (Interactive Creative Environment), joka on visuaalinen ohjelmointikieli. Käyn läpi kunkin järjestelmän vuokaavion ja selvennän, miten kukin komponentti toimii, millaisia haasteita sitä tehdessä ilmeni ja millaisen tutkimustyön kautta päädyin valitsemiini ratkaisuihin. Pyrin mahdollisimman yksityiskohtaiseen purkuun, jotta järjestelmästä kiinnostuneet pystyvät halutessaan uudelleen- tai jatkokehittämään sitä eri ohjelmointikieliä tai ohjelmistoja käyttäen.

Lopuksi yhteenvedossa ja pohdinnoissa käsittelen mahdollisia tulevia parannuksia järjestelmään ja sitä, kuinka uudelleenkäytettävä järjestelmästä lopulta muodostui.

2. Käsitteitä

Seuraavat käsitteet ja niiden ymmärtäminen on tärkeää epäselvyyksien välttämiseksi. Samassa se on alaan vihkiytymättömälle lukijalle erittäin lyhyt oppimäärä kolmiulotteisesta tietokonegrafiikasta. Pääasiallisena lähteenä tässä osiossa käytän englanninkielistä Wikipediaa, jossa on näistä peruskäsitteistä mielestäni yksinkertaisimmat kuvaukset.

2.1 Piste, Vokseli, Pistepilvi

Piste on geometrian objekti, jolla on sijainti, mutta ei kokoa toisin sanoen sillä ei ole pituutta, pinta-alaa, tilavuutta tai muutakaan mitattavaa ominaisuutta. Piste on siis nollaulotteinen objekti . Piste ilmaistaan suhteessa avaruuteen, missä se sijaitsee [(1)]. Tämän opinnäytetyön kontekstissa piste sijaitsee aina kategorisesti kolmiulotteisessa avaruudessa.

Vokseli *((engl. voxel*) eli volumetrinen pikseli on muutoin samalainen kuin piste sillä erolla, että se on sijaitsee säännöllisessä ruudukossa ja sen sijainti ilmaistaan yleensä suhteessa muihin vokseleihin eikä eksplisiittisenä avaruuden

koordinaattina[2]. Pikseli sijaitsee kaksiulotteisessa ruudukossa ja on bittikartan eli digitaalisen kuvan peruselementti. Vokseli puolestaan sijaitsee kolmiulotteisessa ruudukossa. Teknisesti ja ohjelmoinnillisesti opinäytetyössäni käsitellään vain pisteitä, mutta kirjoitan niistä ajoittain myös vokseleina ilmaistakseni niiden sen hetkiset tai alkuperäiset säännölliset tai ruudukolliset sijainnit suhteessa toisiinsa.

Pistepilvi *(engl. point cloud)* on joukko pisteitä, jotka opinäytetyöni kontekstissa sijaitsevat aina kolmiulotteisessa avaruudessa [3]. Pistepilvi on myös Softimage-ohjelmistossa primitiivi, eli tietyn tyyppinen kolmiuloitteinen perusobjekti, jota ohjelmassa voidaan manipuloida ja animoida eri tavoin. Tyypillisimpiin tapoihin manipuloida pistepilven pisteitä lukeutuvat niiden lisääminen, poistaminen, siirtäminen, kiertäminen, skaalaaminen, erilaisten ominaisuuksien eli attribuuttien määrittäminen ja edellä mainittujen muuttaminen suhteessa aikaan. Toisinaan tälläistä järjestelmää nimitetään myös partikkelijärjestelmäksi.

2.2 Verteksi, Monikulmio, Monikulmioverkko

Monikulmioverkko on kolmiulotteisen tietokonegrafiikan peruselementtejä. Se koostuu monikulmioista, jotka puolestaan koostuvat kolmioista ja edelleen vertekseistä. Monikulmioverkko sisältää tietoa siitä, mitkä verteksit koostavat kolmioita keskenään ja edelleen mitkä kolmiot muodostavat monikulmioita keskenään[4]. Lisäksi kahden verteksin välillä kulkee jana, joka muodostaa kunkin kolmion reunan *(engl. edge).*[5]

Monikulmio on geometrian objekti, joka koostuu useammasta kolmiosta, joilla voi olla useita automaattisia tai käyttäjän määrittelemiä ominaisuuksia, kuten esimerkiksi pinnan normaali (kohtisuora vektori monikulmion pinnasta) tai tekstuurikoordinaatti [6].

Verteksi eroaa edellämainitusta pisteestä siinä suhteessa, että sille on sijainnin lisäksi määritelty ominaisuuksia, kuten väri ja normaali. Verteksin normaali eli kohtisuora vektori on yleensä keskiarvo verteksin viereisten monikulmioiden ja niiden kolmioiden normaaleista.[7]

Tasa-arvopinta[8] *(engl. Isosurface)* on kolmi-ulotteisen jatkuvan funktion tasojoukko. Tämän opinnäytetyön puitteissa se tarkoittaa pistepilvestä marssivien kuutioiden algoritmillä[9] laskettua monikulmioverkkoa.

2.3 Skalaari, Vektori, Matriisi

Skalaari on yksittäinen suure, kuten vaikkapa lukumäärä, paino, tilavuus tai pelkkä luku, joka kuvastaa vapaavalintaista tai kuvitteellista suuretta [10]. Ohjelmoinnissa sen tyyppi voi vaihdella esimerkiksi binääri-, liuku-,tai kokonaisluvun väilllä.

Vektori on suure, jolla on suuruus eli magnitudi ja suunta. Vektorin suunta on sijainti koordinaatistossa minne vektorin kärki osoittaa origosta eli nollakohdasta käsin [11]. 3D-grafiikassa kolmiulotteinen vektori määritellään aina X-, Y- ja Z-komponenteilla tai alkioilla. Fysiikassa esimerkiksi nopeus on vektori, jolla on suunta ja pituus, eli toisin sanoen liikkeen suunta ja sen suuruus.

Matriisi on taulukko, jolla on tietty määrä rivejä ja sarakkeita. Taulukko voi sisältää lukuja, symboleja, lausekkeita, käytännössä mitä vaan [12]. Tietokonegrafiikan kannalta olennainen matriisi on affiini *kuvaus,* eli matriisi, jolla voidaan manipuloida monikulmioverkkoa siirtäen, kääntäen, skaalaten tai venyttäen. Affiini kuvaus takaa että tapahtunut muunnos säilyttää monikulmioverkon homogeeniset koordinaatit, eli pisteet, jotka olivat ennen muunnosta suorassa janalla ovat edelleen yhdensuuntaisesti janalla [13]. Hyvin monesti animaatiotuotannoissa pyritään pitämään monikulmioverkkojen koordinaatistot myös ortogonaaleina, eli kohtisuorina, suhteessa toisiinsa, jotteivät affiinin kuvauksen laskentajärjestyksestä johtuen käännökset vinoutuisi. Osa ohjelmistoista, kuten Softimage, oletusarvoisesti kompensoi vinoutumista vastaan, mikä helpottaa tietyissä tilanteissa animaatiohahmojen rakentamista.

Skalaarit, vektorit ja matriisit ovat käytännössä olennaisimmat peruskäsitteet tietokonegrafiikassa, sillä miltei kaikki käsiteltävät kokonaisuudet rakentuvat niille.

2.4 Sävytys, säteenjäljitysrenderointi ja pisterenderointi

Renderointi *(engl. rendering)* eli kolmiulotteisten kappaleiden, kuten monikulmioverkkojen kuvantaminen on niin visualisoinnin kuin tietokonepeligrafiikan ytimessä nykypäivänä[14]. Peligrafiikassa edellytetään reaaliaikaista laskentaa, jolloin kuvantamisen laadusta joudutaan tinkimään tai vaihtoehtoisesti keksimään ovelia tapoja huijata laskennallisesti intensiivinen todellisuuden imitointi. Animaatio- ja erikoistehostetuotannoissa on sen sijaan aikaa laskea kuvat valmiiksi, jolloin on mahdollista saavuttaa paljon tarkempi simulaatio todellisuuden valo -ja materiaali-ilmiöistä. Säteenjäljitys *(engl. ray tracing),* joka tunnetaan myös nimellä säteenseuranta, on yksi näistä tarkemmista ja suosituimmista menetelmistä[15].

Menetelmä pyrkii yksinkertaistettuun versioon valon ominaisuuksien havainnollistamisesta, ja nykypäivänä kuvannettavasta aiheesta riippuen pystytään saavuttamaan varsin realistinen jäljitelmä valon fysikaalisesta käyttäytymisestä. Yksinkertaistettuna kyse on yhdestä pisteestä, tässä tapauksessa virtuaalikameran sijainnista, kuvatason läpi kuvannettavaa maailmaa kohti piirrettävistä säteistä. Säteiden kimpoilua monikulmioverkkojen pinnasta ja pinnan läpi tarkastellaan niille määriteltyjen materiaaliominaisuuksien puitteissa.

Kuvannettavien monikulmioverkkojen materiaaliominaisuuksia kuvataan sävyttimillä *(engl. shader).* Sävytyksessä huomioidaan myös virtuaalisten valonlähteiden vaikutus. Kun valojen ja materiaalien vaikutukset on laskettu yhteen, syntyy ns. näyte *(engl. sample)* ja kun laskostetaan(*engl. antialiasing*) joukko kuvatasolla lähekkäisiä näytteitä, voidaan lopuksi piirtää bittikarttaan tietyn sävyinen pikseli. Kun jokainen kuvatason pikseleistä on laskettu, lopputuloksena on bittikartta, jonka voi tallettaa tietokoneen muistista kuvatiedostoksi kovalevylle. Prosessia nopeutetaan eri keinoin, kuten esimerkiksi laskemalla millä etäisyydellä monikulmioverkkojen kolmiot sijaitsevat ja näkyvätkö ne kameralle, jolloin voidaan tehostaa kolmioiden säteenjäljitysrenderointia.

Pisterenderointi *(engl. splatting)*, jota opinäytetyöni toiminnallisen osuuden renderoinnissa on käytetty, on osa tilavuusrenderoinnin *(engl. volume rendering)* menetelmiä, jotka eroavat perustavanlaatuisesti monikulmioverkkojen renderoinnista

[16][17]. Monikulmioverkkojen sijasta renderoidaan tilavuusdataa, kuten edellämainittuja vokseleita tai pistepilviä. Pisterenderoinnissa jokainen tilavuusdatan tai pistepilven vokseli tai piste renderoidaan syvyysjärjestyksessä piirtämällä yksi primitiivi, yleensä ympyrä, per piste, jonka koon voi määrittää jokaiselle pisteelle erikseen. Menetelmällä menetetään aito valon kimpoilu väliaineessa mutta saavutetaan merkittävästi nopeampi renderointiaika. Tilavuusrenderoinnin eri menetelmät ovat erittäin suosittuja tieteellisessä visualisoinnissa ja ovatkin tärkeä osa lääketieteellistä kuvantamista, kuten tietokonekerroskuvausta tai magneettikuvausta. Pisterenderöinnin ja tilavuusdatan jäljittely monikulmioverkkojen animointiin ja renderointiin tarkoitetussa Softimage-ohjelmassa oli haastavaa ja hidasta, mutta samalla myös oikotie haluttuun ja aidomman oloiseen visuaaliseen ilmeeseen.

3. Case: Twins-animaatio

Opinnäytetyöni toiminnallinen osuus oli taiteilija Adel Abidinin Twins-animaatiota varten Fake Graphics Oy:ssä teknisenä johtajana kehittämäni järjestelmät. Esittelen tässä kappaleessa projektin esituotannon ja tuotannon eri osa-alueita suppeasti selvittääkseni projektin puitteet. Yksinkertaisuuden nimissä viittaan jatkossa Fake Graphics Oy:hyn sanalla Fake.

3.1 Esituotanto

3.1.1 Toimeksianto -ja antaja

Taitelija Adel Abidin otti elokuussa 2011 yhteyttä Fakeen saatuaan idean uusimmalle teokselleen, jonka hän halusi esille Abu Dhabissa neljättä kertaa järjestettyyn “Art, Talks and Sensations 2011”-näyttelyyn [18]. Kaupungin edustalla Saadiyat-saarella Manarat Al Saadiyat Gardenissa järjestetyn näyttelyn teema vuonna 2011 oli saari ja eristyneisyys. Abidinin halusi kuvata saaria potentiaalisesti vaarallisina paikkoina, joissa ihmisten on taisteltava elintilasta. Toteukseksi hän valitsi animaation kaksosten taistelusta elintilasta kohdussa. Fake Graphics on auttanut Abidinia aiemminkin teostensa teknisen toteutuksen saralla, joten yhteistyökumppanin valinta oli luultavasti hänelle luonteva.

3.1.3 Työryhmä ja aikataulu

Työryhmässämme Fakella oli ohjaajana ja kompositoijana Sami Syrjä, mallintajana Heidi Härkönen, animaattorina Teemu Kutvonen, valaisijana ja renderöijänä Felipe Karvonen, tuottajana Neea Kilkki ja allekirjoittanut teknisenä johtajana. Toimeksiantajan lisäksi prosessissa oli mukana äänisuunnitelijana Miska Seppä Humina Oy:stä sekä editoijana, värisävyttäjänä ja näennäisultraääänilaitteen käyttöliittymägrafiikan suunnitelijana Martin Jäger. Aikatauluksi muodostui toimeksiantajan tarpeiden ja Faken resursoinnin kautta 31.8.2011- 10.11.2011. Teos esiteltiin ensi kertaa “Art, Talks and Sensations 2011”-näyttelyn avajaisissa Abu Dhabissa 15.11.2011.

3.1.4 Käsikirjoitus

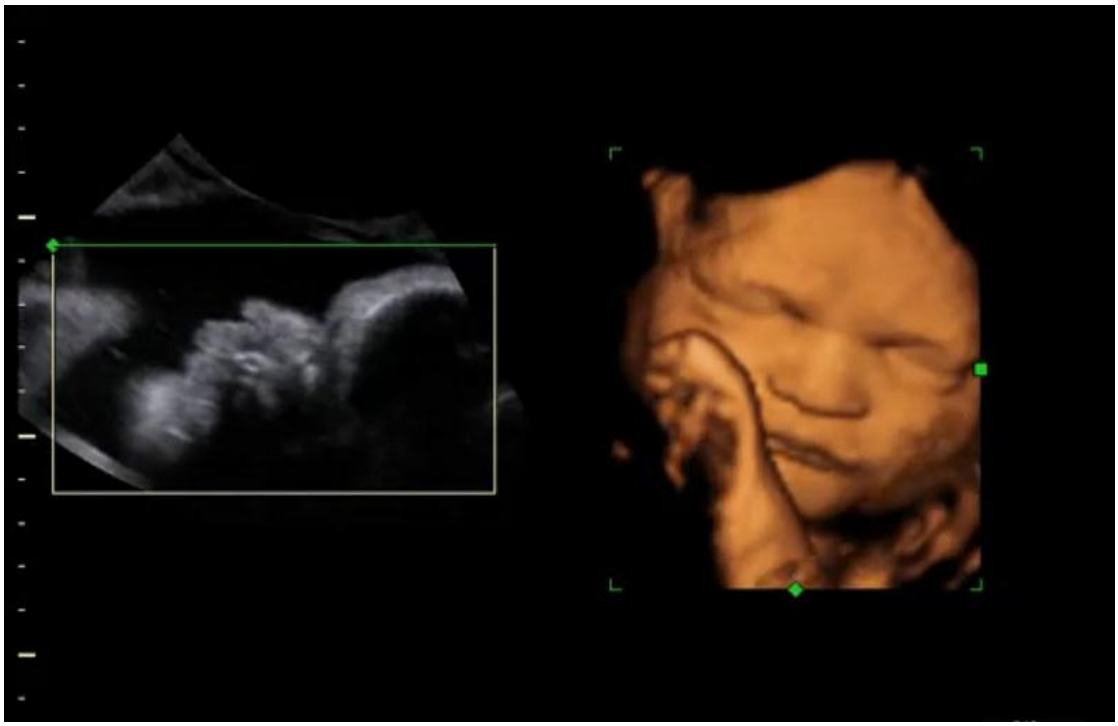
Tapahtumien kulusta Adel laati lyhyen noin sivun mittaisen käsikirjoitusehdotuksen jossa hän kuvasi, kuinka hahmojen välinen taistelu etenisi. Ensilukemalta osa kohtauksista todettiin liian vaikeiksi toteuttaa vaaditulla realismin tasolla ja halutulla aikataululla. Esimerkiksi kohdunsisäisen kannibalismin ja peristautiikan, eli ruoan etenemisen ruokatorvessa lähikuvaaminen todettiin epäkäytännölliseksi projektin aika -ja budjettirajoitteet huomioiden. Faken Sami Syrjä laati tarkemman purun animaation tapahtumista toimeksiantajallemme huomioiden projektin rajoitteet ja etenimme hyväksytettyämme sen kohden ensimmäistä hyvin karkeaa animaatioversiota *(engl. animatic tai previsualization).* Animaticissa ehdotimme toimeksiantajallemme kuvakulmia ja hahmojen asentoja, josta lisää aihetta käsittelevässä kappaleessa 3.2.4 *“Animatic ja animointi”.* Koska kuvakulmat ja ympäristö pysyvät suhteellisen samoina läpi animaation keston, mitään tarkempaa kuvakäsikirjoitusta *(engl. storyboard)* ei katsottu tarpeelliseksi tehdä.

3.1.5 Ilmeen suunnittelu

Adelin otettua yhteyttä Fakeen kävi ilmeiseksi, että hän antaisi työryhmälle suhteellisen vapaat kädet rajattuaan itse idean kohdussa taisteleviin kaksosiin.

Animaation ilmeen suunnittelu alkoi referenssien etsimisestä ja tarkastelusta. Haluttiin selvittää, mikä kohdunsisäisen kuvantamisen nykymenetelmistä olisi tarinankerronnan ja ilmeen kannalta paras. Vaihtoehtoina alussa olivat ultraäänikuva, 4D-ultraäänikuva ja röntgen.

Normaali ultraäänikuva oli tuotannon alussa Adelin mielestä toivottava ja tätä silmällä pitäen tutkimme aluksi erilaisia poikkileikkausjärjestelmiä monikulmioverkoille. Hyvin nopeasti kävi kuitenkin ilmi, että normaali ultraäänikuva ei mahdollistanut riittävän selkeätä kerrontaa tapahtumista eikä sopivaa tunnelmaa kuvallisen syvyyden puutteesta johtuen.



4D-ultraäänikuva, eli ultraäänien kohdussa kimpoilusta rakennettu kolmiuloitteinen malli suhteessa aikaan vaikutti huomattavasti lupaavammaltta vaihtoehdolta. Innostuimme 4D-ultraäänikuvan ominaisuuksista, sillä se oli luontaisesti rosoinen ilmeeltään, mikä olisi mahdollistanut kevyemmän esityön hahmoja rakennellessa. 4D-ilmettä lähestyttiin hahmojen alustavien monikulmioverkkojen perusteella luotuja pistepilviä käyttäen. Testeissäni edellä mainituista pistepilvistä laskettiin tasa-arvopinta animaation joka ruudussa käyttäen Softimagen emPolygonizer-liitännäistä.

Vertailin monikulmioverkkojen pinnalle sirottelulla (*engl. scattering*) luotua pistepilveä monikulmioverkkojen sisään vokselonnilla luotuun pistepilveen. Tasa-arvopinnat tuottivat laadullisesti ja pinnanmuodoiltaan hyvinkin paljon 4D-ultraäänikuvan rosoisuutta muistuttavia tuloksia. Tästä prosessista lisää kohdassa 4.2 *"Tutkimustyö ja testit".* Lopulta katsottiin, ettei 4D-ilme olisi vielä riittävän tunnistettavissa lääketieteelliseksi kuvantamiseksi suurelle yleisölle, eikä näin loisi tarpeeksi immersiivistä katselukokemusta installaatiossa.

3.1.6 Työkalut

Projektin toteuttamiseksi käytimme Fakella Autodeskin Softimagea, joka on kolmiulotteiseen animaatioon ja tehosteisiin erikoistunut ohjelmisto. Renderoinnissa käytimme Softimage-yhteensopivaa versiota Dna Research 3Delight-renderoijasta, sillä se oli tarjolla olleista vaihtoehdoista kaikista nopein pisterenderoinnissa.

Tutkiessani mahdollisia toteustapoja vokselointiin ja kudosten simulointiin päädyin soveltamaan Softimagen Interactive Creative Enviromentia, visuaalisen ohjelmoinnin ympäristöä, joka mahdollistaa ja helpottaa nykytuotannoille tyypillisten monimutkaisten, proseduraalisten ja uudelleen käytettävien järjestelmien rakentamisen. Toinen vaihtoehto olisi ollut Side Effects Houdini, joka on täysin proseduraalinen kolmiulotteiseen animaatioon ja tehosteisiin erikoistunut ohjelmisto. Houdini ei ollut taloudellisesti mielekäs vaihtoehto ohjelmiston lisenssin kovasta hinnasta johtuen, joka oli noin 7600 euroa kirjoitushetkellä [(18)].

Renderoitujen kuvien viimeistelyyn käytettiin Eyeon Fusion-ohjelmistoa, jolla on helppo yhdistellä ja muokata erilaisia renderoituja kuvatiedostojonoja. Projektimme kohdalla se tarkoitti eri kuvatiedostojen eri värikanavien (punainen, vihreä, sininen) sävykorjaamista ja yhdistämistä monokromaattiseksi harmaasävykuvaksi, josta lisää kappaleessa 3.2.5 *"Valaisu ja renderointi".* Lisäksi käytettiin Adoben After Effects-ohjelmistoa sekä Eyeon Generation-ohjelmistoa animaation kuvaruutujen harvennukseen ja leikkauksen työstämiseen, josta lisää kappaleessa 3.2.5 *"Animatic ja Animointi"*

3.2 Tuotanto

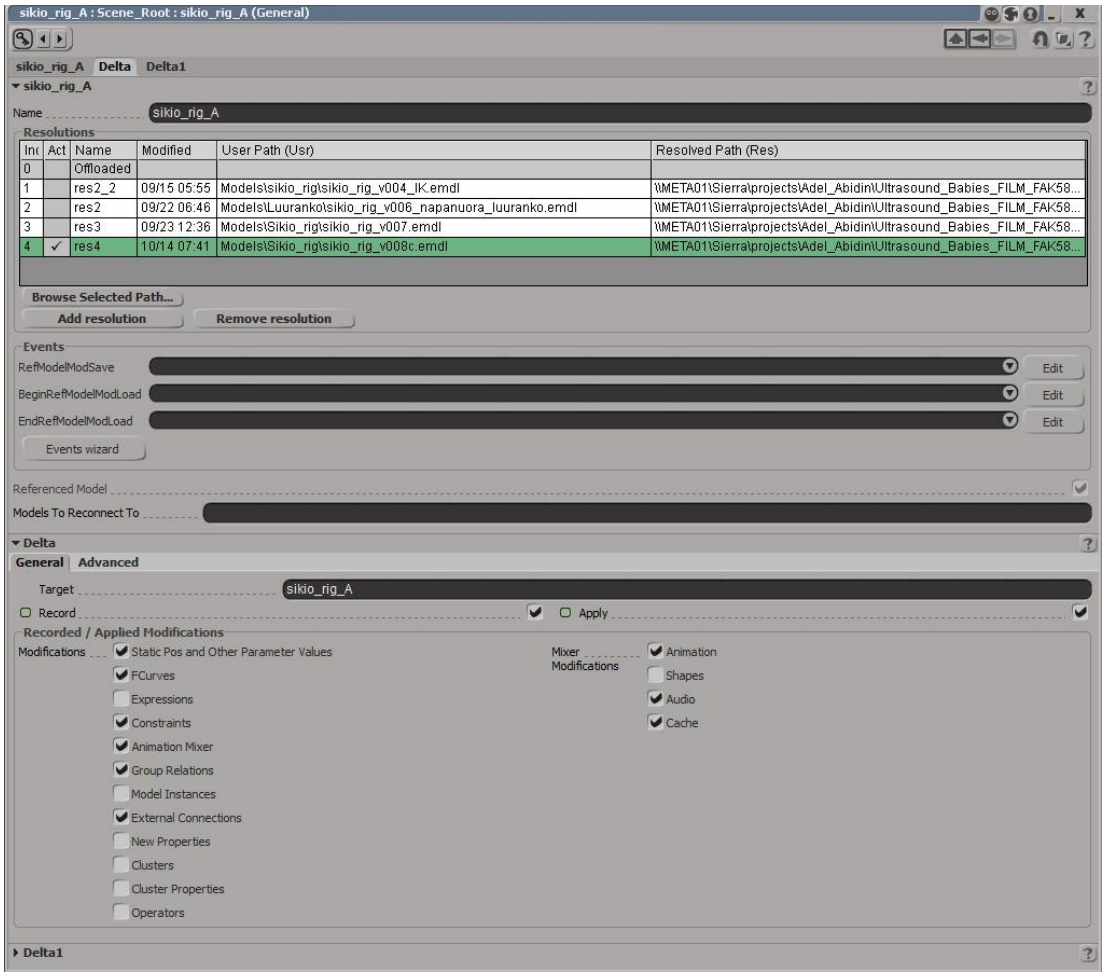
3.2.1 Projektinhallinta

Projektinhallinnassa Fakella käytämme omaa versiotamme Hirotaka Takeuchin ja Ikujiro Nonakin vuonna 1986 kehittämästä Scrumista, joka on ohjelmointimaailmasta tuttu ja suosittu projektinhallinnan muoto[(19)]. Puramme animaation kuviksi (*engl. shot*) ja resursseiksi (*engl. asset*). Resurssi on esimerkiksi hahmo, jota käytetään monessa kuvassa. Alustavan purun jälkeen jokainen kuva ja resurssi puretaan tehtäviksi (*engl. task)* ja niiden arvioitu kesto työtunneissa kirjataan ylös. Tehtävä on esimerkiksi hahmon monikulmioverkon mallinnus tai yksittäisen kuvan valaisu ja renderointi. Tehtävien keston arviointi suoritetaan yhdessä, mutta yhtä tehtävää suorittaa vain yksi työntekijä. Jokaista tehtävää liikutetaan post-it-lapulla tai vastaavalla vasemmalta oikealle ruudukolla kuvaten tehtävän etenemistä aina asiakkaan hyväksyntään asti. Pitämällä joka aamu pikapalaveri jokaisesta projektista siihen osallistuvien kesken saavutetaan selkeä käsitys projektin etenemisestä ja ongelmat tuodaan kaikkien tietoon mahdollisimman pianja ne on tehokkain ratkaista. Projektimme Scrum-purussa päädyimme 16 kuvaan ja yhteen resurssiin eli itse hahmoon, sillä taistelevat sikiöt ovat identtisiä.

3.2.2 Rinnakkainen työskentely ja versiointi

Kustannustehokkaassa modernissa animaatiotuotannossa on olennaista, että eri työntekijöiden on mahdollista työskennellä samanaikaisesti käyttäen, muokaten ja päivittäen samoja resursseja. Softimagessa tämä mahdollisuus on totetettu nk. referenssimallien avulla (*engl*. *reference models*). Referenssimallit ovat Softimagen malli-primitiivejä, jotka voivat sisältää monikulmioverkkoja tai miltei mitä tahansa muita Softimagen primitiivejä tai objekteja. Kun tälläisen mallin vie scn-tiedostopäätteen omaavasta softimagetyötiedostostaan erilliseksi emdl-tiedostoksi tallennusmedialle, on se ladattavissa referenssimallina takaisin muihin Softimagen scn-työtiedostoihin. Muiden työntekijöiden omissa scn-työtiedostoissaan tekemät muutokset malliin, kuten esimerkiksi animaatio -tai sävytinmuutokset, voidaan

tallentaa erillisiin delta-ominaisuuksiin, jotka voidaan halutessa tallentaa omiksi scn-työtiedostoista riippumattomiin delta-tiedostoikseen. Tällä tavoin kaikkien työntekijöiden scn-työtiedostot sisältävät linkin esimerkiksi animaatiohahmon emdl-tiedostoon, ja kun hahmon monikulmioverkosta tai animaatiosta on saatavilla uusi versio, se päivittyy kaikkiin työtiedostoihin automaattisesti.

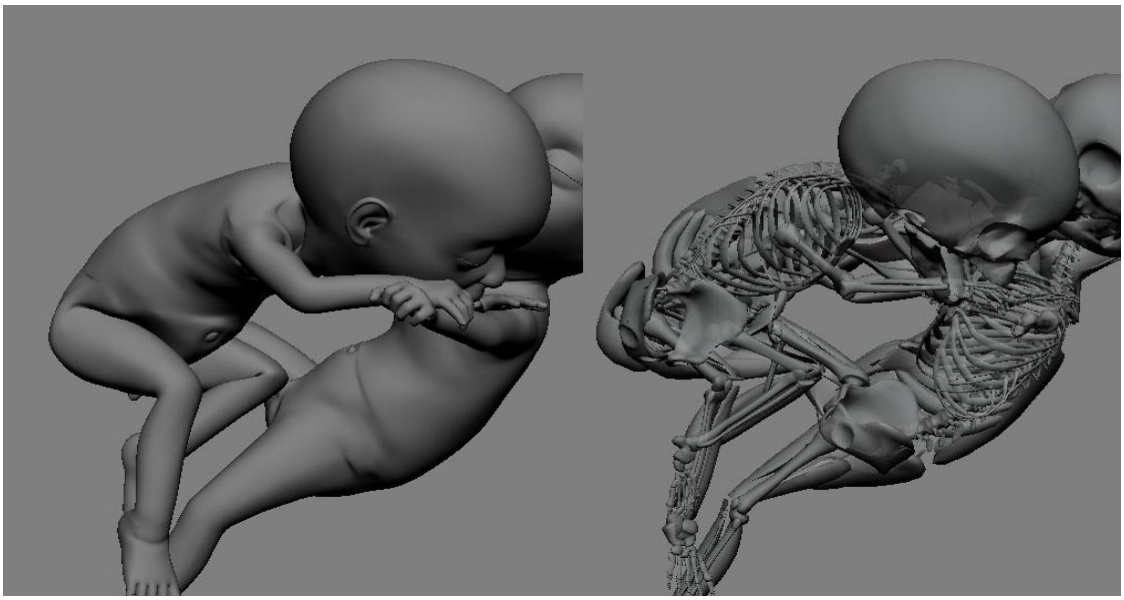


Halutessa uusi versio voidaan lisätä nk. resoluutioksi (*engl*. *resolution*) referenssimalliobjektin listaan, jolloin on mahdollista helposti vaihdella hahmon eri versioiden tai monikulmioverkkotarkkuuksien välillä. Raskaimmat monikulmioverkot eivät tarjoa interaktiivisia toistonopeuksia animaattorille, jolloin alhaisemman monikulmiomäärän omaava resoluutio mallista on tarpeen animointityön helpottamiseksi. Tyypillinen referenssimallien käyttötilanne projektiin osallistuvien työntekijöiden kesken on hahmon samanaikainen valmistelu animoitavaksi, eli nk. rikaaminen (*engl*. *rigging*), ja teksturointi. Rikaaminen on lainasana purjehdussanastosta, mutta osuva kuvaus digitaalisen marionetin toiminnasta

näkymättömien digitaalisten köysien kautta. Vastaavasti hahmon samanaikainen animointi ja renderointi on erittäin suosittu työtapa. Ilman referenssimallien suomaa työskentelytapaa esimerkiksi animaatiohahmojen eri versioiden seuranta läpi eri työntekijöiden työtiedostojen muuttuu logistiseksi painajaiseksi. Tällöin myös työntekijöitä kalvaa epävarmuus uusimmasta tai relevantista hahmoversiosta.

3.2.3 Mallinnus ja rikaaminen

Koska animaation kaksoset olivat identtiset, niin olivat niiden monikulmioverkotkin. Heidi Härkönen Fakelta mallinsi kahteen kertaan käytetyn sikiöhahmon monikulmioverkot. Hahmon kudokset oli jaettu neljään eri tyyppiin ja näin ollen mallinnettiin uloimman ihomonikulmioverkon lisäksi lihas-, luu- ja sisäelinmonikulmioverkot. Hahmosta mallinnettiin lopuksi versioita, joissa niiltä puuttui osia vokseloinnin nopeuttamiseksi väkivallan edetessä. Tämän lisäksi mallinnettiin irtojalka- ja käsi. Tätä kerronnallisen jatkuvuuden säilyttämisen problematiikkaa käsittelen lopuksi opinnäytetyöni pohdinnoissa .



Teemu Kutvonen rikasi, eli loi hahmolle digitaalisen marionetin käyttäen Softimagen standardia nk. biped-hahmorikiä. Prosessi käsitti biped-hahmorikin luurankotyöpohjan asettelun hahmon monikulmioverkon sisään, jonka perusteella Softimage generoi lopullisen animaatiokäyttöön soveltuva digitaalisen marionetin, Bipedin. Biped-marionetti koostuu hierarkiasta Softimagen primitiivejä, joilla on erilaisia vuorovaikutus- ja alisteisuussuhteita. Tämän jälkeen hahmon uloimman ihon

käsittävän monikulmioverkon verteksit painotettiin (*engl*. *enveloping tai skinning)* Softimagen biped-marionetin animointiluurangon luuprimitiivien muunnosmatriiseihin (affiini kuvaus, ks. osa 2, käsitteet). Tästä seuraa, että kun Biped-marionettia animoi Softimagessa, monikulmioverkon verteksit deformoituvat eli liikkuvat marionetin mukana.

Vaikka hahmo oli mittasuhteiltaan aikuisesta merkittävästi poikkeava, standardin Softimagen Biped-rikin käyttö oli tässä tapauksessa perusteltua. Kokemuksesta tiedettiin, että alusta asti oman animaatiorikin rakentaminen kestäisi liian pitkään, eikä tässä yhteydessä tarjoaisi mitään animaatiotyötä helpottavia ominaisuuksia ilman kuukausien kehitystyötä. Hahmot eivät myöskään puhu tai avaa silmiään missään vaiheessa, joten tämä puhui myös valmiin rikin käytön puolesta, sillä monimutkaista kasvorikiä ei tarvinnut pohtia.

Hahmolle mallinnettiin myös Napanuoramonikulmioverkko, joka painotettiin kiinni bezier-käyrään (*engl*. *bezier curve*), jota puolestaan deformoitiin Softimagen Syflex-kangasimulaationliitännäisellä luonnollisen liikkeen saavuttamiseksi. Hahmon vokseloinnista ja napanuoran monikulmioverkkojen manipuloinnista lisää osiossa 4.3 *“Vokselointijärjestelmä”.*

3.2.4 Leikkaus, animatic ja animointi

Animaation kestoksi muotoutui 2 minuuttia 33 sekuntia, eli yhteensä 3828 bittikarttaa eli 25 bittikarttaa sekunnissa PAL-standardin mukaisesti. Tämä oli puolestaan jaettu 16 kuvaan leikkauksin. Leikkausta työstettiin Eyeon Generationilla, joka on aluperin Fakea koeryhmänä käyttäen kehitetty Juha Pinolan toimesta. Generationilla oli helppo vertailla animaation kehitystä kuvien eri versioiden välillä. Seuraavan sivun kuvassa on nähtävillä Eyeon Generationin käyttöliittymä, jossa kutakin animaation leikkausta kohden on kolme versiota joista uusin alimpana.

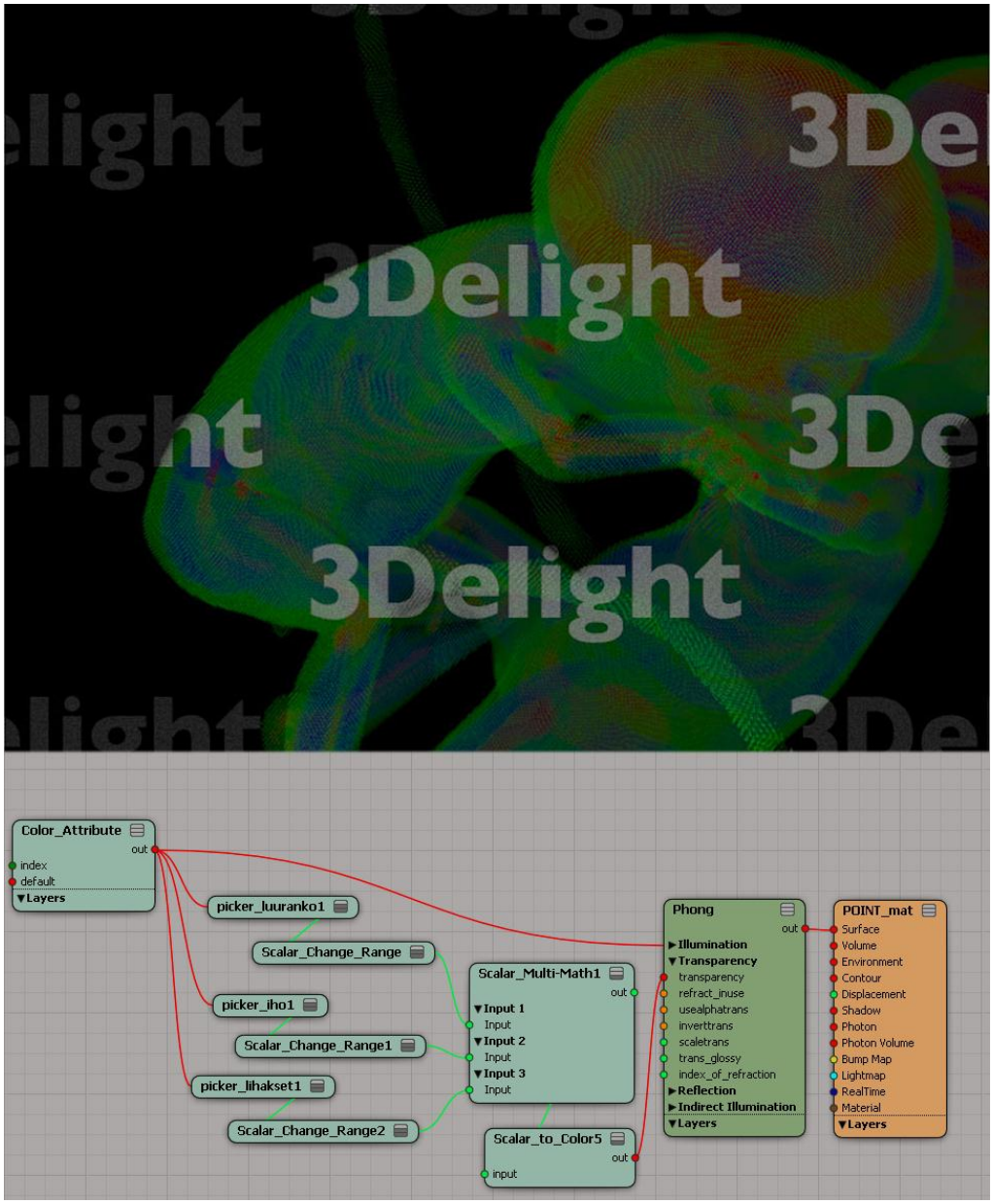
Teemu Kutvonen teki Adel Abidinin hyväksymän käsikirjoituksen pohjalta Softimagessa karkean animaation eli animaticin selvittääkseen kuinka paljon aikaa käsikirjoituksessa mainitut tapahtumat vaativat ja millainen leikkaus toimisi parhaiten. Animaatiota edistettiin referenssimallien avulla, joten päivittämällä referenssimallin tai avaamalla Softimagetyötiedosto uudestaan saatiin uusimmat päivitykset hahmoon. Esimerkiksi samalla kun animaatiotyö eteni sisempiä kudoksia mallinnettiin yhä. Toimeksiantajan toiveesta hahmojen animaatiota työstettiin kohden elokuvista tuttujen elävien kuolleiden hapuilevaa, tahatonta mutta tappavaa liikehdintää. Haasteena animaatiotyössä oli realismin ylläpito, sillä lapsivedessä sikiöt liikkuivat hitaasti ajelehtien ja hahmojen keskinäinen kontakti eli tarttuminen ja irtautuminen toisistaan oli ongelmallista rikien toisistaan erillisistä animointiobjektien hierarkioista johtuen.

Hahmojen raajat animoitiin siten, että niiden perusteella oli mahdollista simuloida vokseloinnilla monikulmioverkkojen sisään luodun pistepilven repeytyminen, joka on tämän opinnäytetyön varsinainen aihe, josta lisää osiossa 4.*“Vokseloinnin ja kudosten simulointi”*

3.2.5 Valaisu ja renderointi

Kukin animoitu, vokseloitu ja simuloitu yhden kuvan (*engl. shot*) käsittävä Softimage-työtiedosto valaistiin muutamalla valonlähteellä Softimagessa. Valonlähteet ovat primitiivejä Softimagessa kuten monikulmioverkot ja ne tulkataan renderoinnin

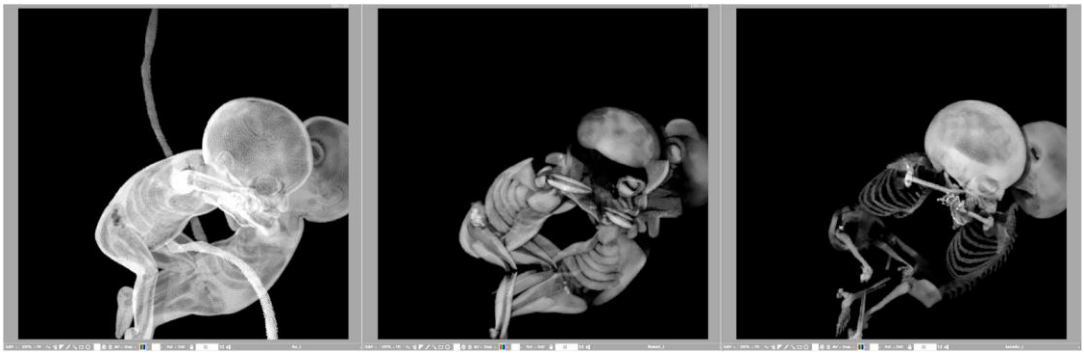
yhteydessä valitun renderointiliitännäisen ymmärtämiksi objekteiksi. Projektimme kohdalla renderoinnissa käytössä oli edellämainittu Dna Research 3Delight. Valaisussa määritettiin valon primääri suunta ja yleensä lisättiin myös hahmojen napanuorille oma valo sommittelun luettavuuden parantamiseksi, sillä muutoin napanuorat olisivat monesti jääneet primäärin kohdevalon ulkopuolelle. Valaistuksessa pyrittiin salamavalomaiseen ja intiimiin valoon, joka loisi hieman ahtaan ja jopa klaustrofobisen tunnelman.Vokseloinnin yhteydessä laskettiin läpinäkyvyys- ja väriarvot jokaiselle pistepilven pisteelle perustuen etäisyyksiin hahmojen eri kudostyyppien eli monikulmioverkkojen pinnoista.

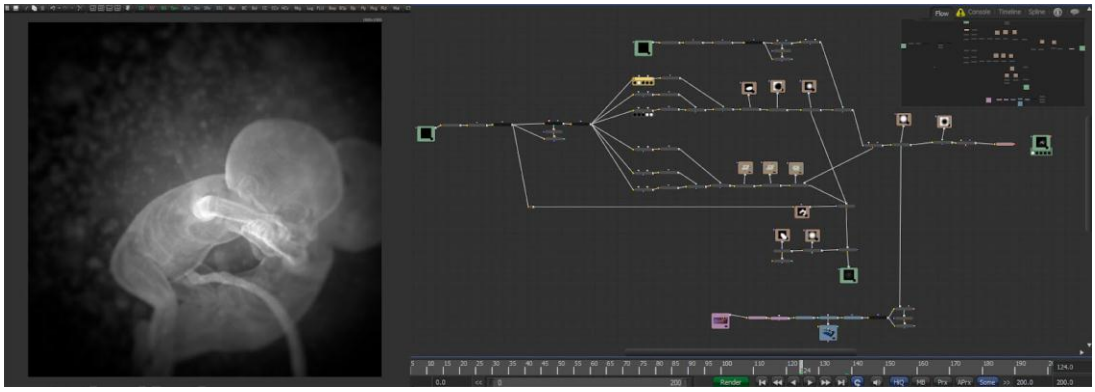


Animaatiohahmojen kudokset oli jaettu ja mallinnettu viiteen eri kudostiheystyyppiin, kun lasketaan mukaan napanuora, mutta renderoinnin yksinkertaistamisen nimissä lihakset ja sisäelimet muodostuivat yhdeksi kudostyypiksi, iho ja napanuorat toiseksi ja luut kolmanneksi. Näin ollen kullekin kudostyypille allokoitiin yksi lopullisen kuvatiedoston kolmesta värikanavasta; punainen, vihreä ja sininen. Tämä yksinkertaisti renderoinnissa syntyvien kuvatiedostojen määrää ja logistiikkaa. 3Delightissa edellämainitussa vokseloinnissa määritetyt eri ominaisuudet, kuten väri ja läpinäkyvyys, luettiin sävyttimeen. Kun pistepilvi-primitiiville vielä määritettiin 3Delightin ns. lightweight particles-ominaisuus, jolla ohitetaan aito tilavuusrenderöinti, saavutettiin mahdollisimman nopea pisteiden sävytys ja renderointi, joka silti vakuuttavasti muistutti lääketieteellistä kuvantamista.

3.2.6 Kompositointi ja jälkikäsittely

Renderoidut kuvat pilkottiin Eyeon Fusion-kompositointi-ohjelmassa Sami Syrjän toimesta värikanaviensa mukaan harmaasävykuviksi, jolloin niitä pystyttiin muokkaamaan eri kudostyypeittäin. Erilaisten sävykorjausten ja kuvankäsittelyiden jälkeen pilkotut värikanavat yhdistettiin lopulliseksi harmaasävykomposiitiksi.
Alla olevassa kuvassa kudostyypit vasemmalta oikealle: iho ja napanuorat, lihakset ja sisäelimet sekä luut. Fusionissa rakennettiin myös kohdun seinämät ja lapsiveden sameus keventäen 3Delight-renderointiurakkaa entisestään.





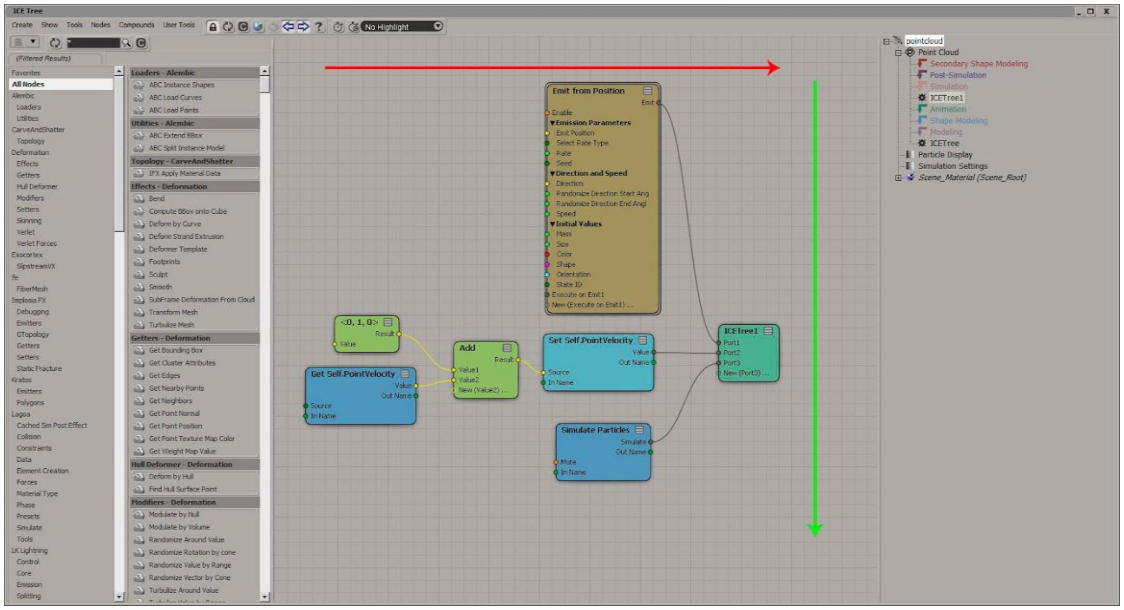
Koska ultraääninauhoitteet ja etenkin 4D-nauhoitteet ovat toistoltaan monesti nykiviä, päätimme tehdä animaation sulavana 25 kuvaa sekunnissa ja jälkikäteen Adobe After Effectsissä harventaa kuvia siten, että liikkeestä tulee samalla tavoin nykivää. Martin Jäger teki lopullisen toimittamamme kuvajonon harvennuksen ja käsittelyn sekä näennäisultraäänilaitteen käyttöliittymägrafiikat.

4. Vokselointi ja kudosten simulointi

4.1 ICE-ohjelmoinnin erittäin lyhyt oppimäärä

Softimagen ICE, eli Interactive Creative Environment, on visuaalinen ohjelmointikieli. Sitä käytetään pääasiallisesti pistepilvijärjestelmien tai monikulmioverkkojen muodonmuutosten eli deformaatioiden aikaansaamiseksi. Tätä nykyä ICE:n kattavuutta on myös laajennettu proseduraalista mallinnusta ja animaatiokinematiikkaa kohden.
Tämän opinäytetyön puitteissa tehty tutkimus -ja kehitystyö painottuu ICEn em. pääasiallisiin käyttötarkoitusten piiriin. ICE-ohjelmoinnissa luodaan vuokaavioita, joissa luodaan ja muokataan erityyppistä ja kontekstista tietoa (*engl. data*). Kontekstilla tarkoitetaan, sitä onko käsiteltävä tieto esimerkiksi sarja arvoja per piste vai sarja arvoja per pistepilvi. Tiedon eri tyypit ovat 3D-ohjelmoinnista tuttuja: liukuluku, kokonaisluku, teksti, vektori, väri, matriisi, jne. ICE:n vuokaavioita kutsutaan “*ICE-Tree*”-nimellä ja ne ovat operaattoreita Softimage-primitiivien rakennuspinossa (*engl*. *modifier stack* tai *construction history*). ICE-vuokaaviossa tieto kulkee vuokaavion kytkettyjä solmuja pitkin vasemmalta oikealle (seuraavan sivun kuvan punainen nuoli) kohden lopullista “execute”-solmua. Kytkentöjen

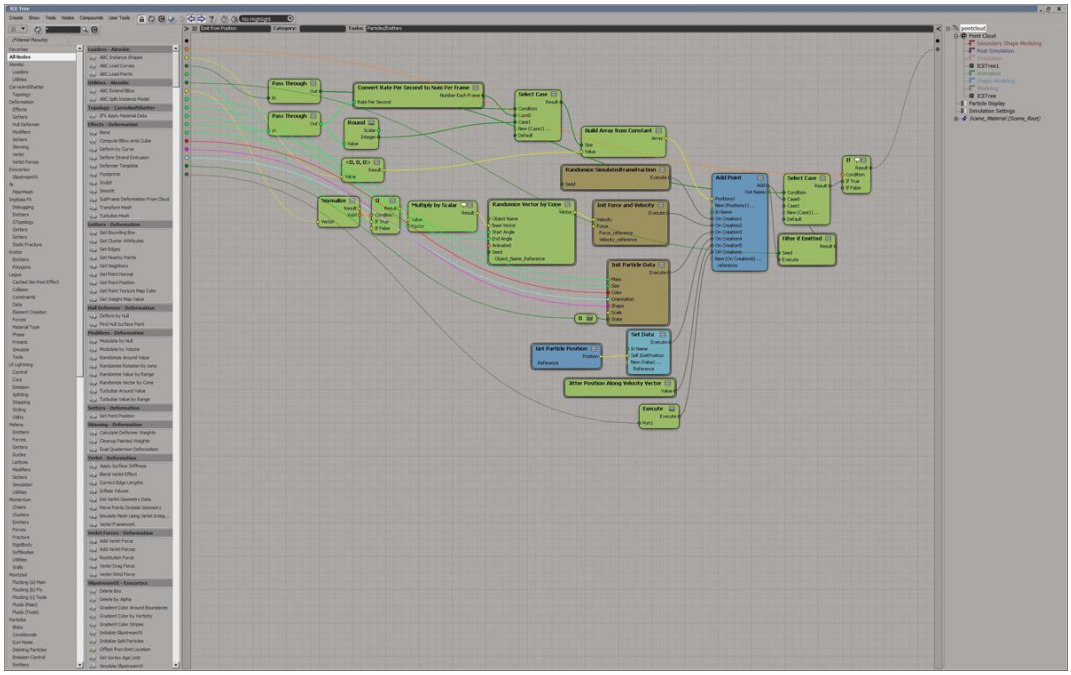
(johtojen) väri kertoo siinä kulkevan tiedon tyypin, esimerkiksi keltainen on vektori, punainen väri, vaaleanvihreä liukuluku, tummanvihreä kokonaisluku, jne.



Muokkaukset voivat kohdistua kaikkeen tai osaan tiedosta vapaavalintaisin perustein. Lopuksi muokkaukset lasketaan tai evaluoidaan (*engl. Evaluation*) kytkentäjärjestyksessä ylhäältä alaspäin (kuvan vihreä nuoli) jokaisessa animaation ruudussa. Evaluointiin vaikuttaa ratkaisevasti missä kohden rakennuspinoa ICE-vuokaavio sijaitsee. Mallinnustasolla (*engl. Modeling Stack*) sijaitsevat ICE-vuokaaviot evaluoidaan joka ruudussa uudestaan hukaten muistista edellisessä ruudussa tapahtuneet muokkaukset. Simulaatiotason ICE-vuokaavio sen sijaan käyttää edellisten ruutujen muokkausten kertymää lähtökohtana kunkin ruudun tiedon evaluointiin. Ylläolevassa kuvassa näkyy simulaatiotason ICE-tree keskinäkymässä ja suhteessa pistepilviprimitiivin rakennuspinoon oikeanpuoleisessa näkymässä.

ICE-vuokaavion osia eli kytkentöjä ja solmuja voi paketoida *(engl. Compound*), mikä helpottaa toteutetun visuaalisen ohjelmoinnin ymmärrettävyyttä ja käytettävyyttä. Alla olevassa kuvassa on navigoitu “Emit From Position”-paketin sisään paljastaen sen muodostavat solmut ja alipaketit, joita käytetään pisteiden

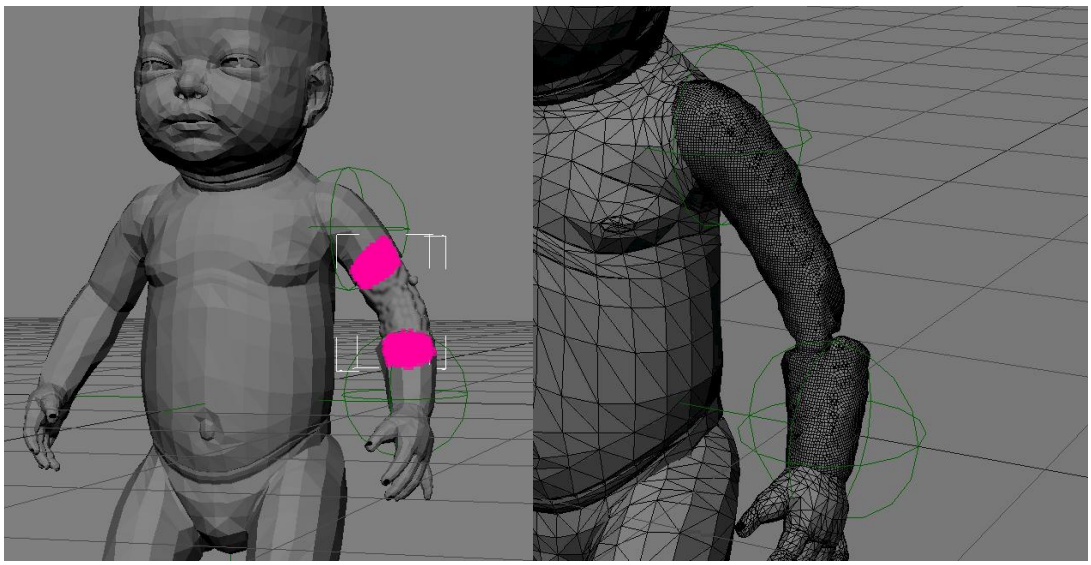
luontiin tiettyyn paikkaan avaruudessa ja tietyin lähtökohtaisin ominaisuuksin kuten massa, koko, nopeus, suunta, kierto, jne.



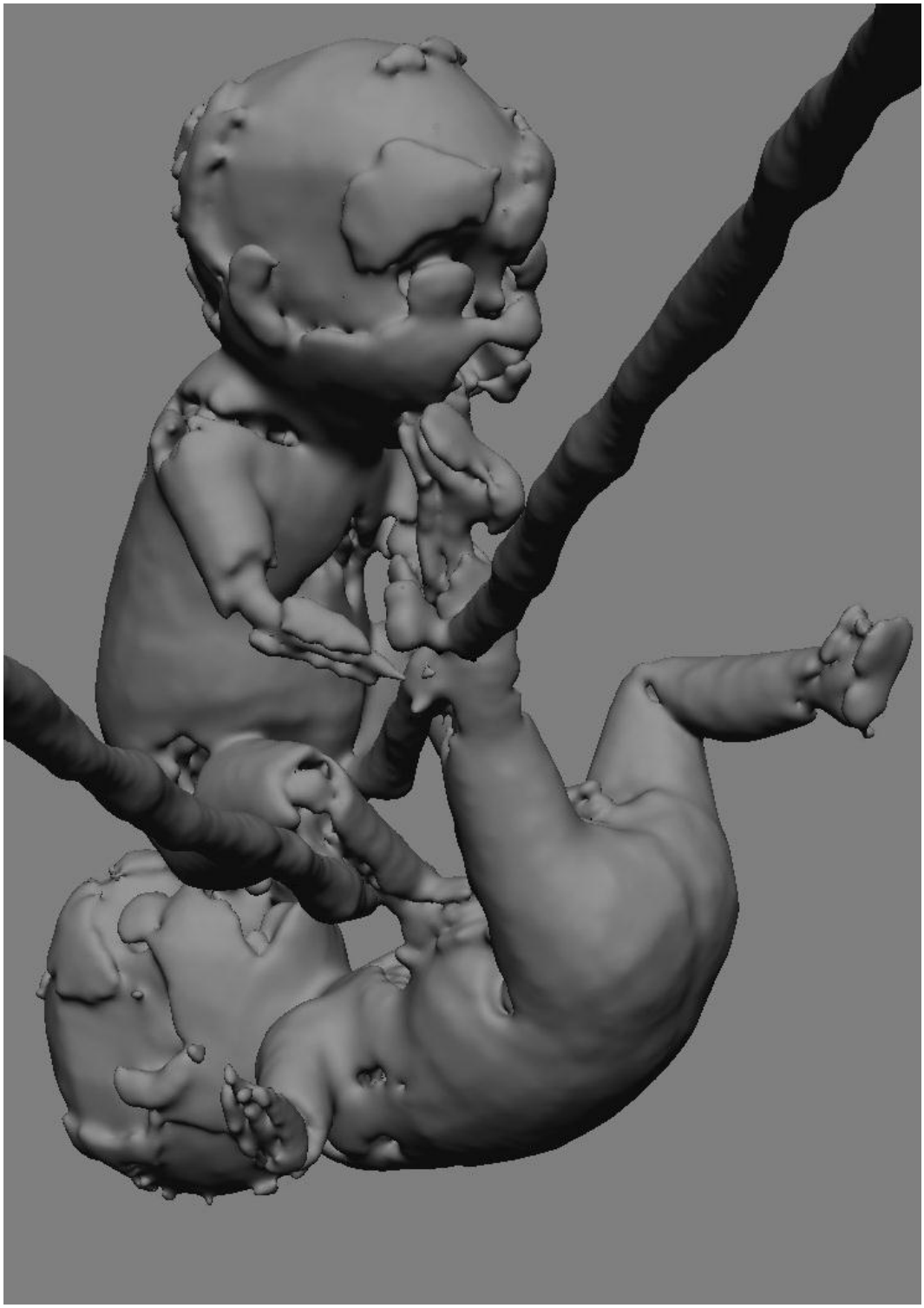
4.2 Tutkimustyö ja testit

Aloitin tutkimustyön selvittämällä, olivatko projektin haasteet ratkottavissa normaalilla monikulmioverkkojen manipuloinnilla. Koska laatukriteerit, animaatio ja lopullinen ilme oli tuolloin vielä kesken, oli järkevää perehtyä kaikkiin eri vaihtoehtoisiin toteutustapoihin. Tiedossani oli, että Side Effectsin Houdini-ohjelman proseduraalinen monikulmioverkkojen mallinnus tarjoaisi mahdollisesti työlään, mutta toimivan ratkaisun projektin ihmissikiöiden kudosten repeytymisen visualisointiin. Tuotannollinen ja taloudellinen paino projektissa oli kuitenkin hiljan tuotantolinjaan lisätyssä Softimage-ohjelmistossa. Ratkaisu oli näiden seikkojen valossa löydettävä Softimagen työkaluilla. Koska proseduraalisen monikulmioverkkomallinnuksen, tai ICE-mallinnuksen (*engl. ICE modeling*), työkalut ovat uusia ja suhteellisen kehittymättömiä, päätin lähestyä asiaa aluksi testaamalla tasa-arvopintoihin perustuvaa monikulmioverkkojen ja pistepilvien yhdistämistä. Tällä pyrin välttämään suurten pistemäärien aikaansaaman työskentelynopeuden hidastumisen. Tasa-arvopinnat loin Softimagen emPolygonizer3-liitännäisellä, joka ajaa marssivien

kuutioiden algoritmiä muodostaaksen animaation joka ruudussa uuden monikulmioverkon syötetyistä monikulmioverkoista ja pistepilvistä. Opinäytetyössäni ICE-mallinnusta on silti käytetty vain nimellisesti, lähinnä monikulmioverkkojen kopioinnin ja uudelleenjaon merkeissä, josta lisää seuraavassa kappaleessa.



Yllä olevissa kuvissa on tilapäisen vauvamallin monikulmioverkko eri tavoin yhdistettynä pistepilveen tasa-arvopinnaksi. Vasemmanpuoleisessa kuvassa tasa-arvopinta on röpelöinen alue kahden Null-primitiiveillä (vihreät ympyrät) kiinnitettyjen magentan väristen pisteiden kohdalla. Oikeanpuoleisessa kuvassa tasa-arvopinta kattaa olkavarren ja kyynärvarren monikulmioverkot sekä kudossimuloidut pisteet välissä (karkeasti repeävä osa). Kudosten simuloinnista lisää kohdassa “*4.4.1 Kudossimulaatio”.*

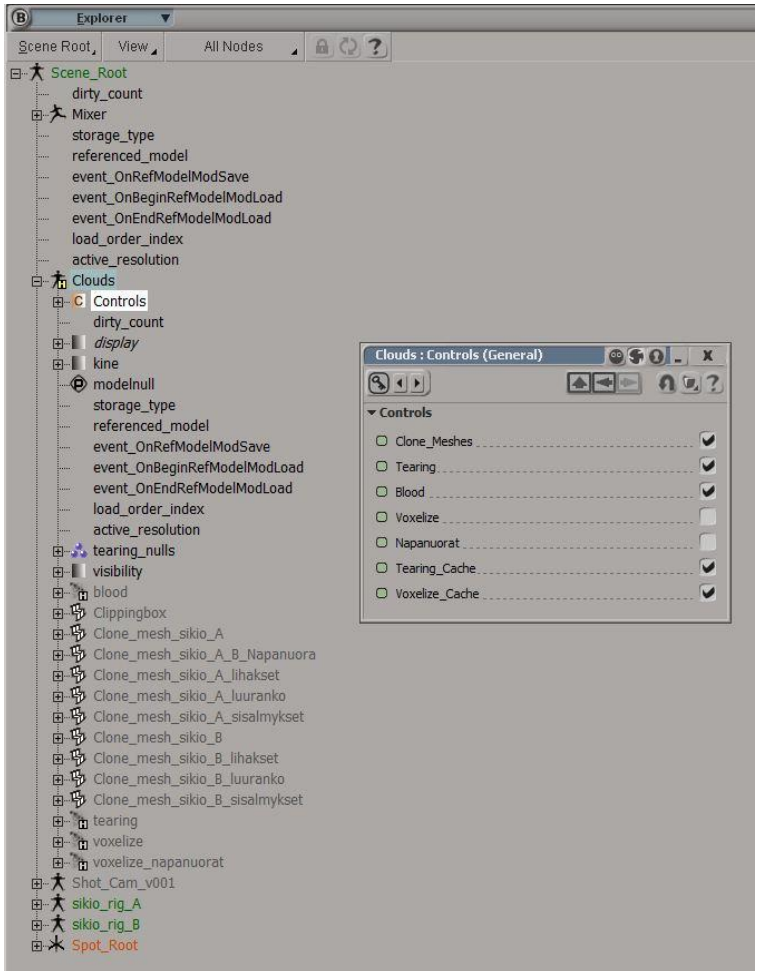


Tasa-arvopintatesti tuotannon myöhemmässä vaiheessa. Epätarkkuudet pinnassa ovat tarkoituksellista pisteiden seulontaa määritettyjen kudostiheyksien perusteella lääketieteellisen kuvantamisen jäljittelemiseksi.

4.3 Vokselointijärjestelmä

4.3.1 Rakenne ja työnkulku

Vokselointijärjestelmän ytimessä on Softimagen mallit (*engl. Models*), jotka mahdollistavat vokselointi- ja simulaatiojärjestelmien, eli pistepilvien ja niiden eri kontekstisten vuokaavioiden (*engl. ICE-Tree*) tallentamisen levylle uudelleenkäyttöä varten. Alla olevassa kuvassa malli "Clouds" eli pilvet on paikallisena kopiona eräässä Sofimage-työtiedostossa, jossa on animoitu "sikio_rig_A" ja sikio_rig_B"-malleja, jotka sisältävät sikiöhahmot. Mallit ovat paikallisina kopioina tässä esimerkissä. Mikäli niiden ikonit olisivat valkoisia, olisivat ne referenssimalleja, joita käytettiinkin kattavasti projektin alkuvaiheessa muutosten automaattista päivittymistä varten (ks. kappale 3.2.2 *Rinnakkainen työskentely ja versiointi)*. Lisäksi kuvassa näkyvässä renderöintityövaiheen Softimagetyötiedostossa on yksi valonlähde ja kamera ("Shot_Cam_v001" ja "Spot_Root").



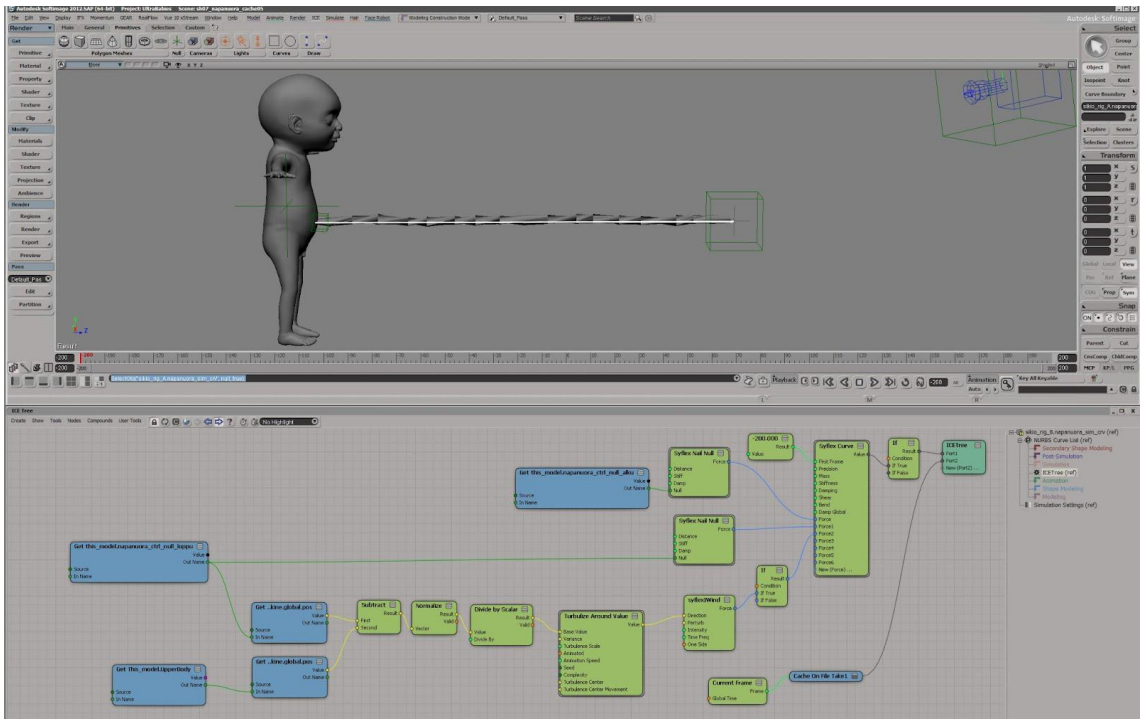
Työtiedoston Clouds-malli sisältää myös pistepilvet “voxelize”, “voxelize_napanuorat”, “tearing” ja “blood”, joista lisää seuraavissa kappaleissa. Kaikkia neljää pistepilveä käytettiin vain niissä kuvissa missä tarvittiin kudosten repeämisen simulointia. Mallissa on myös useita “Clone_mesh”-alkuisia monikulmioverkkoaihioprimitiivejä (*engl*. *empty polygon mesh*) sikiö-mallien eri monikulmioverkkojen ICE-mallinnuspohjaista kopiointia ja uudelleenjakoa (*engl. subdivision)* varten. Pistepilvet viittaavat näihin aihioihin mahdollistaen yhden lisävaiheen, jossa tehdä vapaavalintaisia muokkauksia koskematta alkuperäisiin animoituihin sikiömalleihin. Oranssina kuvassa valittuna oleva ja aukaistu “Controls”-ominaisuussivu *(engl. property page)* sisältää parametrejä, jotka ovat linkitetty moniin eri parametreihin ympäri Clouds-mallin eri osia helpottaen eri ominaisuuksien nopeata kytkemistä päälle ja pois. Lisäksi mallissa on “Clipping box”-kuutiomonikulmioverkko, jolla oli mahdollista rajata vokseloitavaa osaa sikiömalleista, sekä “Tearing_nulls”-ryhmän, joka sisältää viittaukset sikiömallien Null-primitiiveihin, joilla määritettiin repeytymiskohta kussakin repeytymistä sisältävässä kuvassa. Niin “Tearing_Nulls”-ryhmäprimitiivi kuin monikulmioverkkoaihioitkin mahdollistivat Clouds-mallin eriyttämisen Softimage-työtiedostoista ja helpon uudelleenkäytettävyyden eri kuvien työtiedostoissa.

Työnkulku Clouds-mallin järjestelmillä oli yksioikoista. Renderöintityötiedoston pystyi luomaan vaivatta tuomalla aluksi kamera-, sikiö-, valo- ja Clouds-mallin uuteen tiedostoon. Tämän jälkeen referenssimalleina tuotuihin sikiömalleihin ladattiin kyseisen kuvan animaation sisältävät delta-ominaisuudet (ks. kappale 3.2.2 *Rinnakkainen työskentely ja versiointi)*. Clouds-mallin kuvaan soveltuvat ominaisuudet kytkettiin Controls-ominaisuussivulta päälle, kuten “Clone meshes”, “Voxelize” sekä “Voxelize napanuorat”, mikäli kyseessä oli yksinkertainen kuva ilman repeytymisiä. Vokselointi ei vaatinut kätkömuistien käyttöä, vaan monikulmioverkkojen liikkeestä riippumatta vokselointi toistettiin renderöinnin yhteydessä kuva kuvalta riippumatta edellisien kuvien vokseloinneista (mallinnustason ICE-vuokaavio). Poikkeuksen kätkömuisteihin muodostivat napanuorat, joiden monikulmioverkkoja simuloitiin ladatun animaation pohjalta, josta lisää seuraavassa kappaleessa. Lopuksi kuvan valon suunta haettiin liikuttelemalla työtiedostoon tuotua valoa ja renderöintiasetukset tarkistettiin ennen lopullista renderöintiä.

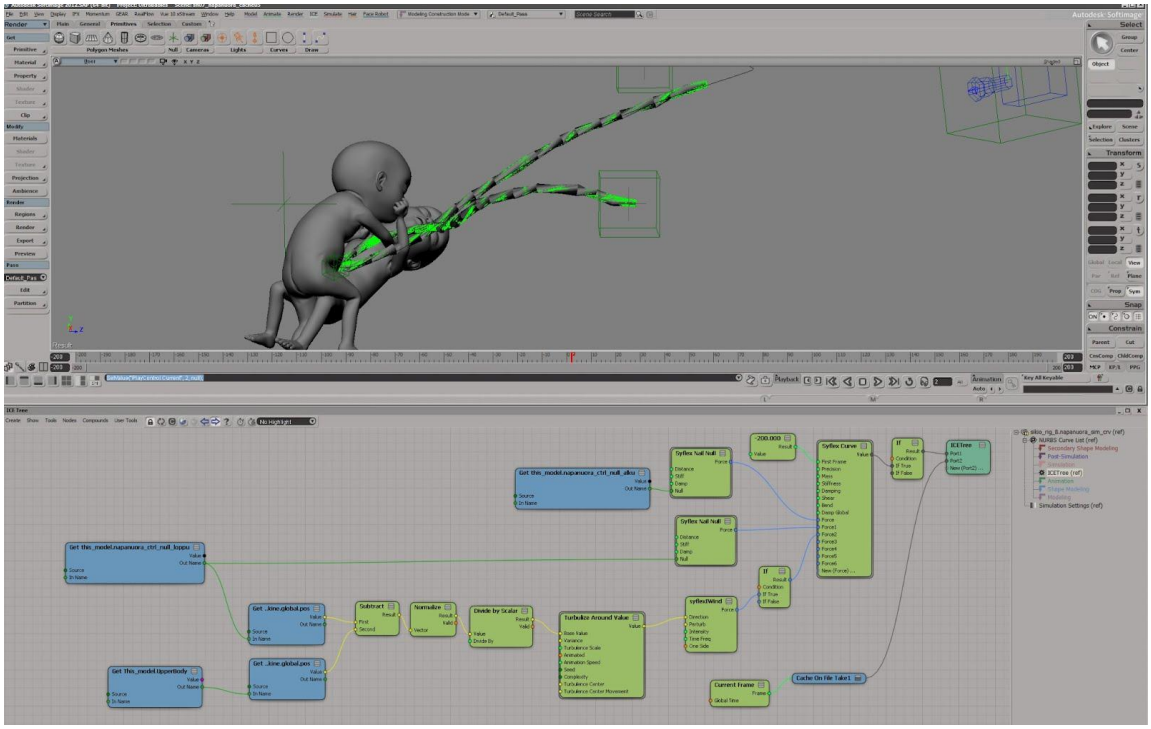
4.3.2 Napanuoramonikulmioverkkojen simulointi

Kuten edellä mainitussa työnkulussa selvennettiin, kuvakohtaisen animaation latauksen jälkeen napanuorien liike simuloitiin nopeimmaksi vaihtoehdoksi havaitulla ja Softimageen integroidulla Syflex-dynamiikkaliitännäisellä. Kaikki animaatio sisälsi sovitusti noin sata ruutua transitiota hahmon oletusasennon ja animaation ensimmäisen ruudun välillä. Simuloinnin kohde oli käyräprimitiivi (kuvassa valkoinen horisontaali viiva), jonka simulointitasolle luotiin ICE-vuokaavio, joka siirsi käyrän pisteitä joka ruudussa siihen kohdistuvien voimien mukaisesti.

Napanuorien monikulmioverkot oli puolestaan painotettu kiinni käyriin Softimagen mallinnustason “Deform by Curve”-operaattorilla. Simuloinnissa käytettiin apuna kahta Null-primitiiviä päätepisteobjekteina, joista toinen oli kiinnitetty linkittämällä kunkin sikiöhahmon napanuoran kohdalle ja toista animoitiin käsin pysymään kuvakulman ulkopuolella. Näin piilotettiin napanuoran kiinnityskohta kohdun seinämään ja vältettiin kerronnallisia jatkuvuusongelmia kuvien välillä. Näitä kiinnityskohtia kutsutaan Syflexin kohdalla “Nauloiksi”, joilla on oma ICE-paketti, “Syflex Nail Null”. Käyrän pisteisiin kohdistettiin ylimääräisiä voimia laskien sattumanvaraistettu voimavektori joka ruudussa napanuoran kuvan ulkopuolisen Null-primitiivin sijainnin ja hahmon keskipisteen välille. Tämä osoittautui testeissä laskentanopeuden kannalta tehokkaimmaksi tavaksi lisätä nesteessä ajelehtimista muistuttavaa liikettä napanuoraan. Samalla liike oli mahdollista laittaa talteen kätkömuistiin, mutta tästä luovuttiin, sillä oli järkevämpää, että simuloinnin yhteydessä napanuorien vokseloitu lopputulos (kuvassa vihreällä) tallennettiin levylle renderöintiä varten, jolloin saavutettiin haluttu lopputulos yhdellä simulointisessiolla.



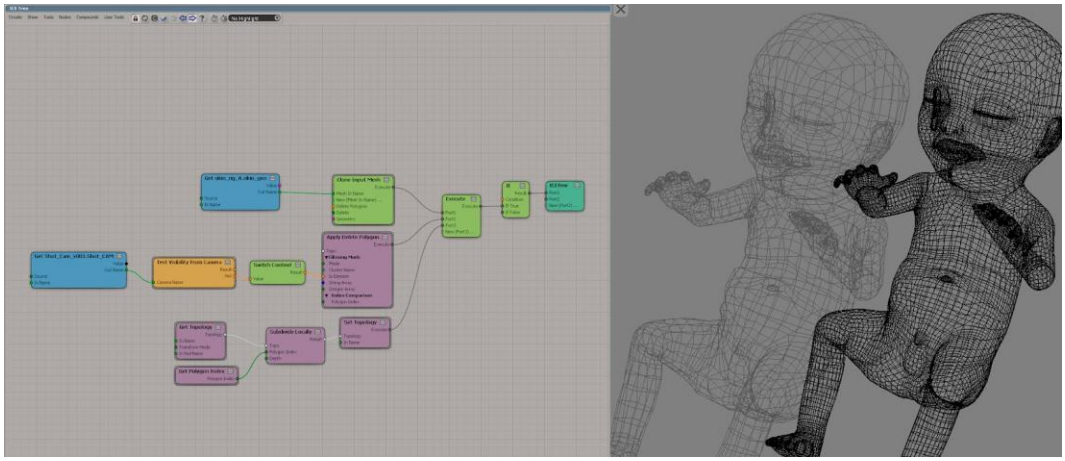
Napanuoran simulointi ruudussa -100 eli ennen animaation alkua, simuloitu käyrä valkoisella



Sata ruutua myöhemmin, kolmannessa ruudussa itse animaation alusta, simuloitu käyrä ohjaa napanuoramonikulmioverkkoja, jotka vokseloidaan(vihreät pisteet)

4.3.3 Monikulmioverkkojen kopiointi ja uudelleenjako

Jotta animaatioluurankoon painotettua monikulmioverkkoa voisi liikuttaa, kiertää, skaalata , deformoida tai uudelleen jakaa tarpeen vaatiessa, oli kehitettävä järjestelmä painotettujen hahmomonikulmioverkkojen eriyttämiseen itse vokselointijärjestelmästä. Etenkin uudelleenjako oli tarpeen ja tämä ei ollut mahdollista itse sikiöhahmon monikulmioverkkoon Softimagen referenssimallien rajoituksista johtuen. Tämä onnistui em. monikulmioverkkoaihioita käyttäen. Luomalla ICE-vuokaavio mallinnustasolle tyhjään monikulmioverkkoprimitiiviin (ts. monikulmioverkkoaihioon) ja käyttämällä Softimagen “Clone Input mesh”-pakettia (*engl*. *Compound*, kuvassa ylin vihreä laatikko keskellä), pystyttiin kopiomaan sikiöreferenssimallien eri kudosmonikulmioverkot (luuranko, lihakset, sisämykset, iho) ja muokkaamaan niitä ennen vokselointia. Lähikuvia varten monikulmioiden näkyvyys kameran kuvakulmasta asetettiin ohjaamaan monikulmioiden poistoa. Tämä saavutettiin yhdistämällä “Test Visibility from Camera”-paketti “Apply Delete Polygon”-pakettiin.
Lopuksi kopioidut monikulmioverkot uudelleenjaetaan (engl. *Subdivision*) “Subdivide Locally”-paketilla. Koska nämä paketit tulevat Softimagen standardiasennuksen mukana, en esittele niiden sisäistä toimintaa solmutasolla. Edeltävän rakennekappaleen kuvassa on nähtävillä kaikki eri monikulmioverkot, joita vokselointijärjestelmässä käytettiin, yhteensä yhdeksän, eli neljä per sikiöhahmo ja lisäksi yksi, jonne oli kopioitu molemmat napanuoramonikulmioverkot optimoinnin nimissä. Kuvassa harmaalla ihokudosmonikulmioverkko sekä mustalla uudelleenjaettu monikulmioverkko vokselointia varten.



4.3.4 Monikulmioverkkojen vokselisointi

Vokselointijärjestelmää oli mahdollista lähestyä monesta suunnasta, mutta ennen kaikkea tiedossa oli, että järjestelmän tulisi kyetä luomaan joka ruudussa miljoonia pisteitä mahdollisimman nopeasti. Eräs yksinkertaisimmista numeerisista vokselointialgoritmeista on tasojoukkometodi (engl. *level set method*, LSM[20]), joka pyrkii täyttämään monikulmioverkkojen tilavuuden tasaisella ruudukolla vokseleita (eli tämän opinnäytetyön kontekstissa pisteitä, ks. *2.1 Piste, Vokseli, Pistepilvi*). Tasojoukkometodi etsii monikulmioverkon äärimmäiset verteksit rajaavan laatikon *(engl. bounding box)* ja täyttää sen kerros kerrokselta seuloen monikulmioverkon ulkopuolelle jäävät pisteet merkityllä etäisyysfunktiolla *(engl. signed distance function*[21]), joka kertoo, onko piste sisä- vai ulkopuolella monikulmioverkkoa. Algoritmi on monimutkaisten ja aukinaisten monikulmioverkkojen kohdalla altis virheille ja siksi oli kiinnitettävä huomiota kopioitujen monikulmioverkkojen topologiaan. Ongelmia aiheuttivat erityisesti kohdat, missä hahmot ovat taipuneina ääriasentoihin, jolloin monikulmioverkkojen painotuksesta johtuen monikulmiot saattavat mennä sisäkkäin, luoden intersektion eli jatkuvuusongelman.

Vokselointialgoritmin testausta noin kymmenesosalla lopullisesta tarkkuudesta. Laatikoita muistuttavat pisteryhmät hahmojen lantioiden vierellä johtuvat viallisesta monikulmioverkkotopologiasta.

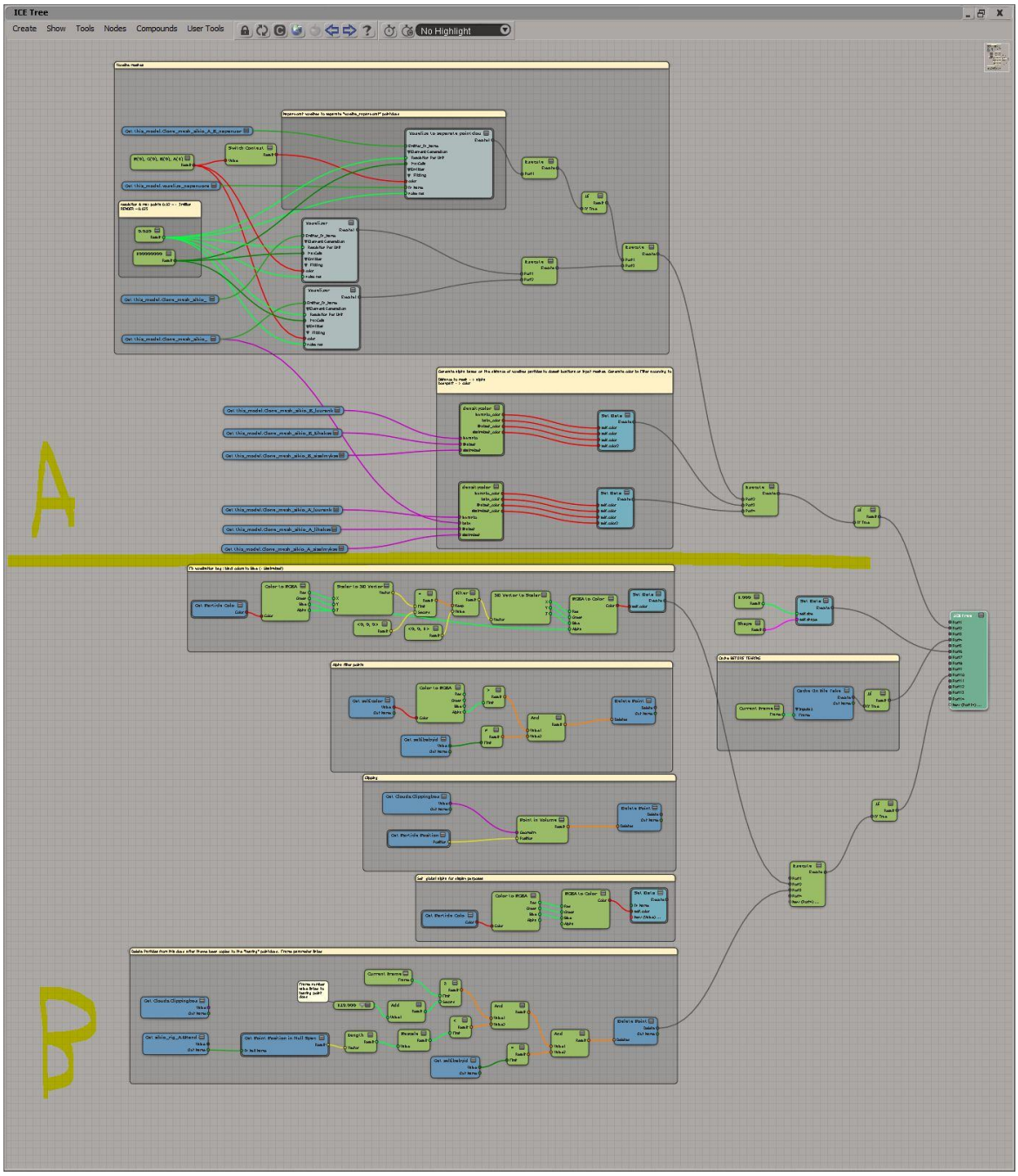
Vokselointijärjestelmän ytimeksi muodostui Softimagen mukana tulevan ICE:llä rakennetun Lagoa Multiphysics-järjestelmän “Lagoa Grid”-solmu, joka käyttää em. LSM-algoritmiä. Perusvokseloinnin kannalta turhat Lagoa-liitännäiset attribuutit pisteille jätettiin luomatta, joskin tilalle laskettiin muita, joista lisää seuraavassa kappaleessa. Hyöty lagoa Gridissä piili sen implementoinnissa. Vaikka samanlainen järjestelmä on helppo tehdä ICE:n normaaleilla komponenteilla, on Lagoa Grid C++-kielellä ohjelmoitu ja binääriksi kompiloitu itsenäinen solmu ja testatusti nopeampi kuin puhtaasti ICE-vuokaaviopohjainen implementaatio samoista toiminnallisuuksista.

Vokselointijärjestelmä em. Lagoa Grid-solmuineen sijaitsi Clouds-mallin “Voxelize”-pistepilven mallinnustason ICE-vuokaaviossa. Se koostui paketeista ja solmuista, jotka täyttivät joka ruudussa sikiöhahmojen uloimmat ihomonikulmioverkot pisteillä, laskivat pisteille väriarvon ja seuloivat pisteitä kuvakohtaisesti tearing-pistepilveen simulointia varten.

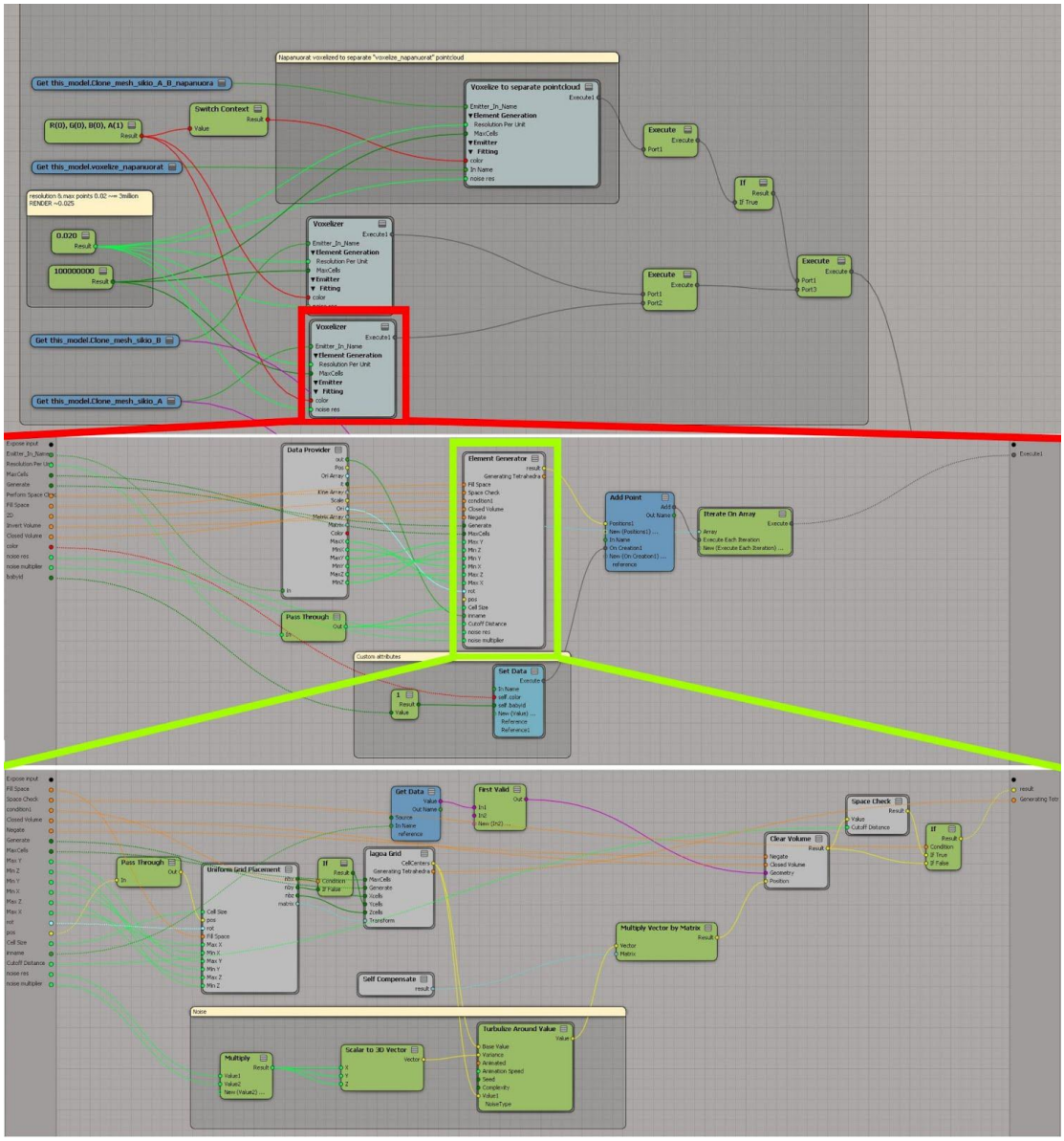
Selkeyden nimissä olen jakanut Voxelize-pistepilven sisällön, eli pakettien, solmujen ja niiden kytkennät kahteen osioon, jotka on rajattu seuraavan sivun kuvassa.

A-osiossa on Lagoan asennuksen mukana tulevaan “Emit from Volume”-pakettiin pohjautuva muokkaamani vokselointipaketti “Voxelizer”, joka sisältää alipaketeissaan lopulta itse Lagoa Grid -solmun sekä “Densitycolor”-paketin kudostiheyden ja pisteattribuuttien laskentaan. Käsittelen tässä kappaleessa vokseloinnin ja Densitycolor-paketin seuraavassa kappaleessa *4.3.5 Kudostiheyden laskenta ja pisteattribuutit.*

B-osio sisältää puolestaan pisteiden muokkausta ja seulontaa,mistä lisää kappaleessa 4*.3.6 Pisteiden seulonta*



Voxelize-pistepilvi:
Osio A sisältää vokseloinnin ja pisteattribuuttien määrityksen. Osio B pisteiden seulontaa, muokkausta ja mahdollisen siirron simuloituun tearing-pistepilveen.



Voxelizer-paketin alipaketit ja sisältö

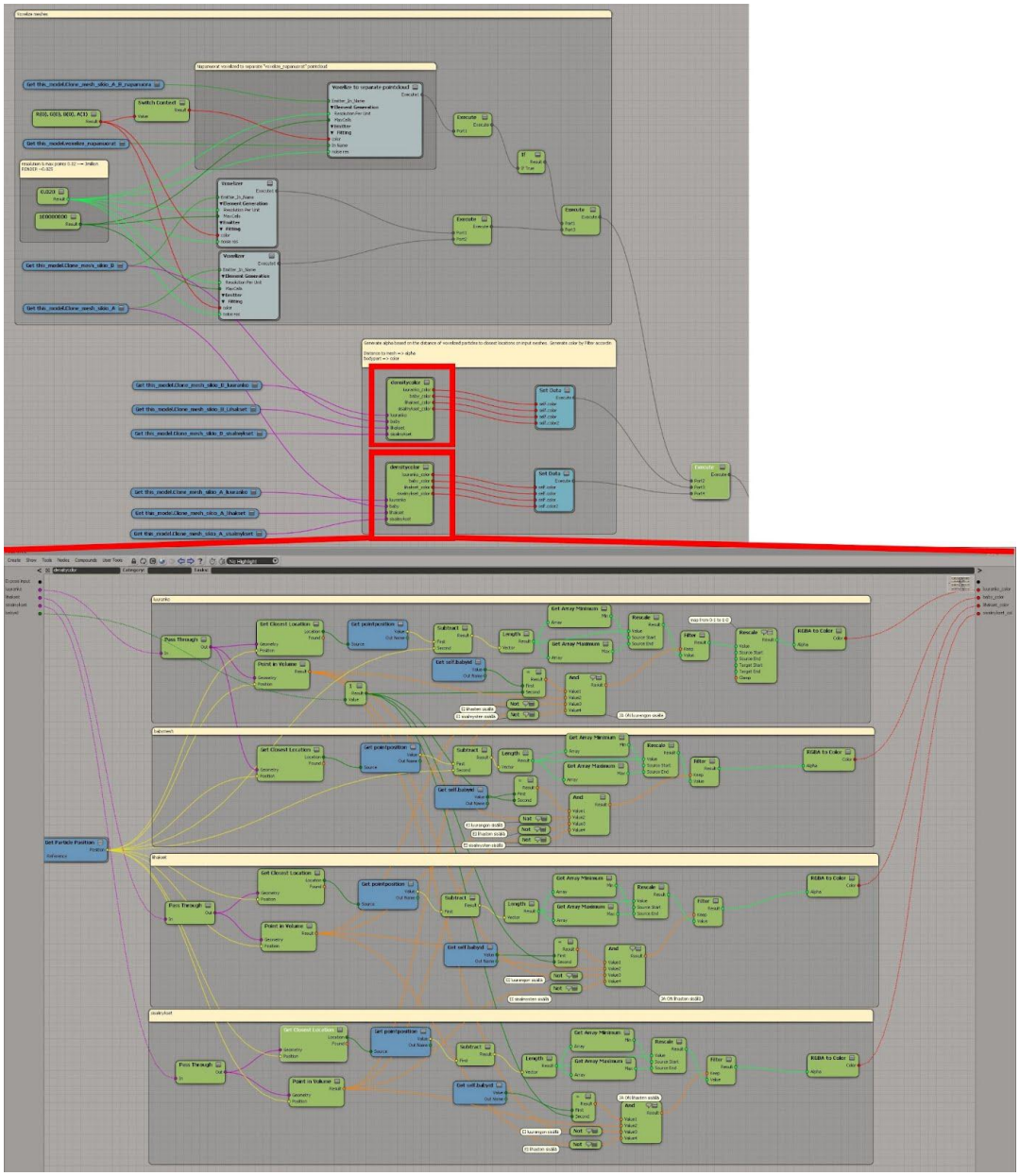
“Voxelizer”-paketti on muokattu versio versio Lagoan “Emit From Volume”-paketista. ja sisältää etupäässä “Element Generator”- ja “Data Provider”-paketit.

”Voxelizer”-paketti ottaa vastaan vokseloitavan monikulmioverkon nimen, pistetarkkuuden per mittayksikkö, pisteiden enimmäismäärän ja pisteiden sijaintien sattumanvaraistamiseen käytettävän liukulukuarvon. Pisteiden sijaintien sattumanvaraistamisella vältetään moire-ilmiö renderoidessa pisteitä. Tasojoukkometodi tarvitsee tietoa vokseloitavasta monikulmioverkosta, jonka “Data

provider”-paketti hakee ja toimittaa edelleen “Element Generator”-paketille, joka sisältää “Lagoa Grid”-paketin ja jonka sisältä lopulta löytyy itse Lagoa Grid-solmu, joka ajaa tasojoukkometodia. “Element generator”-paketti sisältää myös muokkauksen jolla pisteiden sijaintia sattumanvaraistetaan “Turbulize Around Value”-paketin sisältämällä “Turbulence”-solmulla. “Data Provider”-paketti toimittaa tiedon mm. monikulmioverkon raajavan laatikon, edellä mainitun bounding boxin, koosta ja vokseloitavan monikulmioverkon affiinista kuvauksesta.
“Element Generator”-paketti sisältää myös muita toiminallisuuksia, kuten “Negate”, joka kääntää algoritmin toiminnan päälaelleen vokseloiden monikulmioverkon ulkopuolisen tilavuuden aina rajaavan laatikon rajoille asti. Nämä lisäominaisuudet eivät ole käytössä järjestelmässä ja niistä on löydettävissä lisätietoa Softimagen dokumentaatiossa, joten en käsittele niitä tässä opinnäytetyössä.
Lopuksi lisätään myös kuvassa näkyvät pisteattribuutit (“Custom attributes”-ryhmä, alimpana kuvassa), kuten pisteen oletusväri ja “BabyID”, jolla määritetään, kumpaan sikiöhahmoon luotava piste kuuluu, sillä tämä ei ole pistepilven pisteen näkökulmasta ilmeistä. Määritellyt attribuutit arvioidaan sijaintien ohella “Add Point”-solmussa, joka kytkeytyy “Voxelizer”-paketista ulos ja edelleen aina lopulliseen ICE-vuokaavion “Execute”-solmuun saakka. Välissä on ehtosolmu “If”, joka on kytketty Clouds-mallin “Controls”-ominaisuussivun parametreihin kuvakohtaisen työskentelyn helpottamiseksi (ks. *4.3.1 Rakenne ja työnkulku)*

4.3.5 Kudostiheyden laskenta

Densitycolor-paketti ottaa sisään sikiöhahmon neljästä kudosmonikulmioverkosta kolme: lihakset, sisälmykset ja luurangon, jotka ovat kuvassa oikean puoleiset ohuet “Get Data”-solmut. Densitycolor selvittää “Point In Volume”-solmulla, mitkä ihomonikulmioverkon sisään vokseloiduista pisteistä ovat muiden kudosmonikulmioverkkojen sisällä. Yhdistämällä kolme “Point In Volume”-operaatiota Boolen algebralla[21] on mahdollista selvittää eri monikulmioverkkojen sisällä olevat pisteet. ICE-vuokaavion oranssit kytkennät kuvaavat Boolen alkioita 0 ja 1, tai tosi ja epätosi, joita yhdistellään Lisäksi tarkistetaan, että vokseloinnin yhteydessä pisteelle määritetty attribuutti “BabyID” on haluttu, eli piste sijaistee ainoastaan toisen hahmon monikulmioverkon sisässä. Densitycolor-paketteja onkin hahmojen erikseen käsittelemiseksi kaksi, samalla tavalla kuin Voxelizer-vokselointipakettejakin.



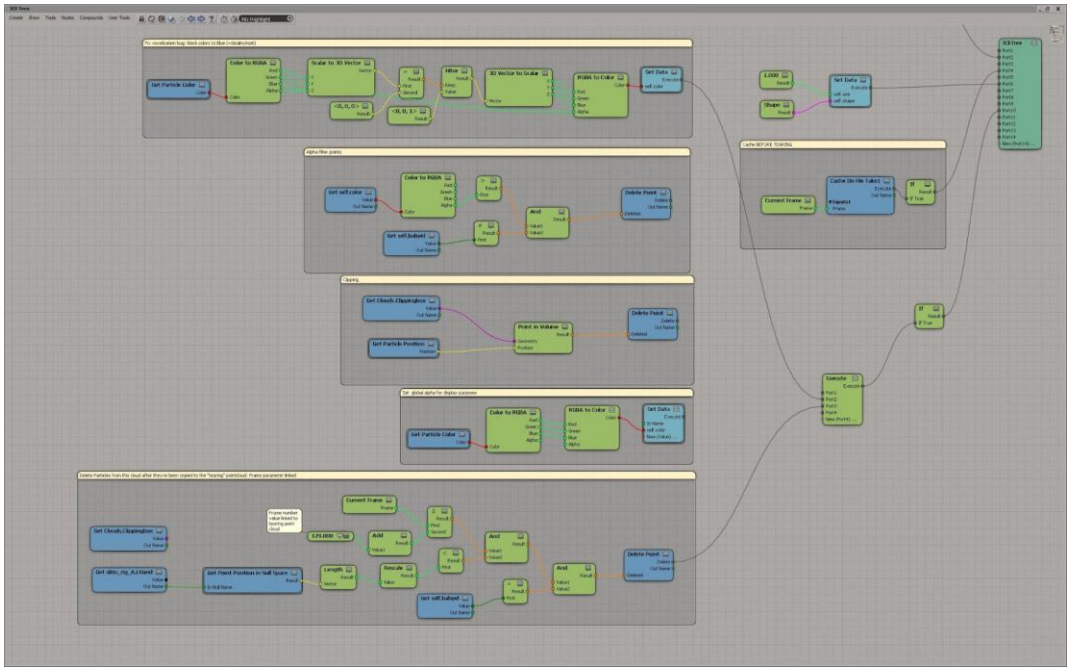
“Get Closest Location”-solmut selvittävät jokaiselle pisteelle lähimmän sijainnin kunkin monikulmioverkon pinnalla, joka vähennetään pisteen sijainnista. Tuloksena on vektori, joka suuruus on pisteen etäisyys monikulmioverkon pinnasta. “Rescale”-solmulla saatu etäisyys normalisoidaan ja sillä ohjataan pisteiden väriominaisuutta. Pisteiden väri määräytyy kudostyypin perusteella punaisen, vihreän ja sinisen välillä ja etäisyys monikulmioverkon lähimpään pintaan määrää läpinäkyvyyden. “Kudostiheyden” laskenta tarkoittaakin järjestelmän puitteissa värityyppisen Color-attribuutin, eli tarkemmin ottaen 32-bittisen (8 bittiä per kolme värikanava sekä yksi

läpinäkyvyyskanava) RGBA-väriarvon, laskentaa. Lopulta “Filter”-solmut eli suodatinsolmut takaavat, että vain tietyn kudoksen pisteiden väriarvot määritetään kerrallaan.

Tätä väriarvoa käytettiin myös simulaatiojärjestelmissä kudosmaisen simuloinnin tukena erottelemaan eri kudoksille omat ominaisuudet pisteille, josta lisää kappaleessa *4.4 Simulaatiojärjestelmät.*Väriarvoa käytettiin renderoinnissa kunkin pisteen sävytyksessä hakemalla väriarvo pistepilven materiaaliin “Color Attribute”-solmulla Softimagen Rendertree-sävytysympäristössä.

4.3.6 Pisteiden seulonta

Kun sikiömallit oli vokseloitu ja pistepilven pisteille oli laskettu väriarvo, voitiin pisteitä jatkokäsitellä tarvittaessa kuvakohtaisesti monella tapaa. Koska jatkuvuusongelmat sikiöhahmojen monikulmioverkoissa saattoivat aiheuttaa yllättäviä virheitä vokseloinnissa, näille virheille oli hyödyllistä saada oma väriarvo jatkokäsittelyä varten.



Ylläolevan kuvan ylimmässä solmuryhmässä mustat eli väriarvoa ilman jääneet artefaktit värjättiin sisälmyksien värillä ja poistettiin kompositoinnissa tarpeen mukaan renderointiartefaktien välttämiseksi. Kehitystyön tässä vaiheessa ei ollut tietoa, halutaanko lääketieteessä käytettyä kudostiheyteen perustuvaan seulontaa

kuvantaa ja näin ollen toiminallisuus jätettiin lopulliseen järjestelmään. Tämä oli yksinkertaista seulontaa perustuen pisteiden väriarvon läpinäkyvyyskomponenttiin, kuten kuvan toiseksi ylimmästä solmuryhmästä käy ilmi.

Myös mahdollisuus läpileikkauksen kuvaamisen säilytettiin em. Clouds-mallin “Clipping Box”-monikulmioverkkoa käyttäen, joka näkyy kuvan keskimmäisessä solmuryhmässä.Hahmojen muotojen hahmottamisen helpottamiseksi työskennellessä animaation parissa loin toiseksi alimman solmuryhmän läpinäkyvyyden säätämistä varten.

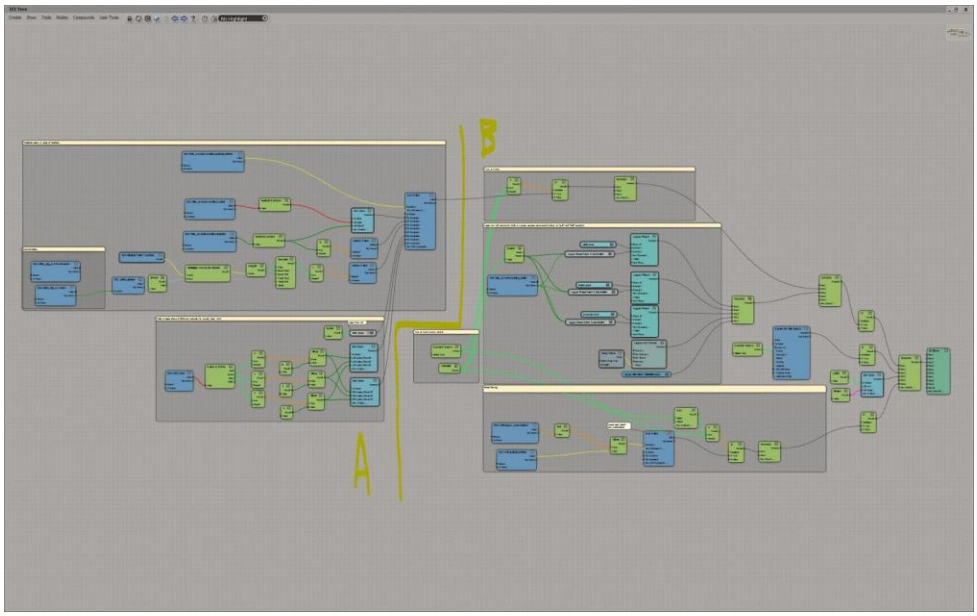
Tietyssä kuvaruudussa, jossa hahmo on animoitu repimään toiselta ruumiinosan, tietyt pisteet kopioidaan “Tearing”-pistepilveen samaisen pistepilven ICE-vuokaaviosta käsin, jossa myös kudosten repeäminen simuloidaan. Siirrettävät pisteet määritettiin etäisyyksilllä repivän hahmon kättä seuraavaan “Tearing Null”-primitiiviin sekä repeytyvän hahmon ruuminosan vastaavan primitiiviin. Tämä olennainen osa pisteseulontoja on kuvan alimassa solmuryhmässä, jossa välittömästi kopioinnin jälkeisessä ruudussa siirretyt pisteet poistetaan Voxelize-pistepilvestä.

4.4 Simulaatiojärjestelmät

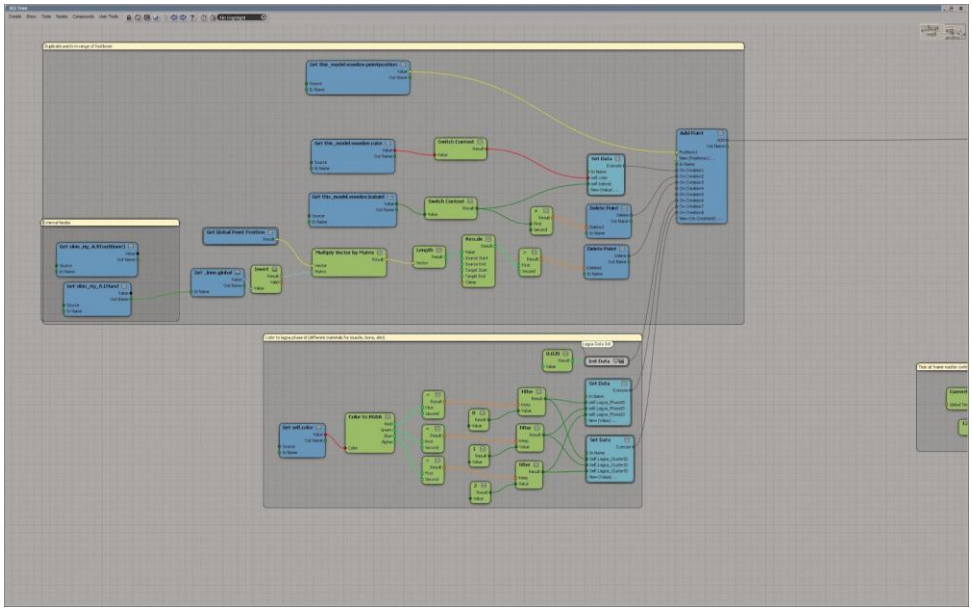
4.4.1 Kudossimulaatio

Samalla tavoin kuin Voxelize-pistepilvenkin, olen jaotellut selkeyden nimissä Tearing -kudossimulointipistepilven solmuryhmät kahteen osioon. A-osiossa valmistellaan oikeat pisteet B-osiossa simulointia varten. A-osio koostuu kahdesta olennaisesta solmuryhmästä. Ylimässä solmuryhmässä haetaan kaikki Voxelize-pistepilven pisteistä muuttaen ne “Switch context”-solmulla Tearing-pistepilven kontekstiin. Pisteet pilveen lisää “Add Point”-solmu ja samassa pisteille haetaan Voxelize-pistepilven väri- ja “BabyID”-ominaisuudet. Repivän sikiöhahmon pisteet sekä kaikki repeytyvän ruumiinosan ulkopuoliset pisteet poistetaan “Delete Point”-solmuilla perustuen “BabyID” -attribuutiin sekä etäisyyteen “Tearing Null”-primitiiveihin. Tällä saavutetaan kriittinen optimointi, sikäli ettei simuloida yhtään ylimääräisiä pisteitä ja työskentely oikeanlaisen simulaation parissa on riittävän nopeasti iteroitavissa lopulliseksi.

Väriominaisuudella eli kudostyypillä -ja tiheydellä puolestaan alustetaan ja määritetään Lagoa-simulointia varten “Lagoa_PhaseID”- ja “Lagoa_ClusterID”-attribuutit kuvan alimmassa solmuryhmässä. Väri muunnetaan komponenteittain liukulukuarvoiksi ja edelleen Lagoa-attribuuttien vaatimiksi kokonaisluvuiksi.

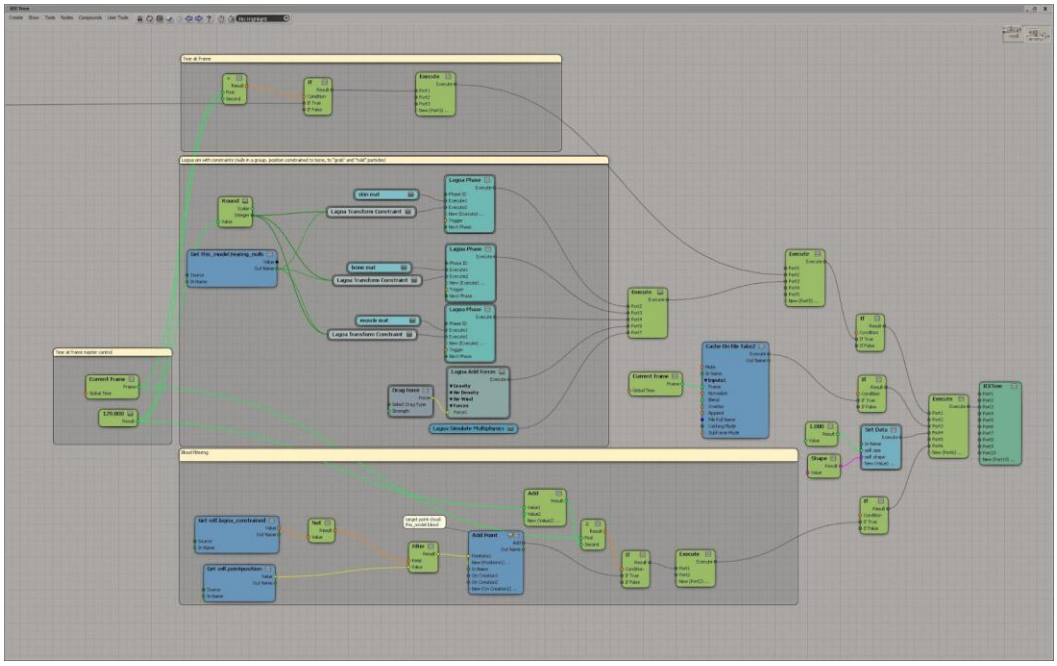


Tearing-pistepilven A-osiossa valmistellaan oikeat pisteet simulaatiota varten B-osiossa

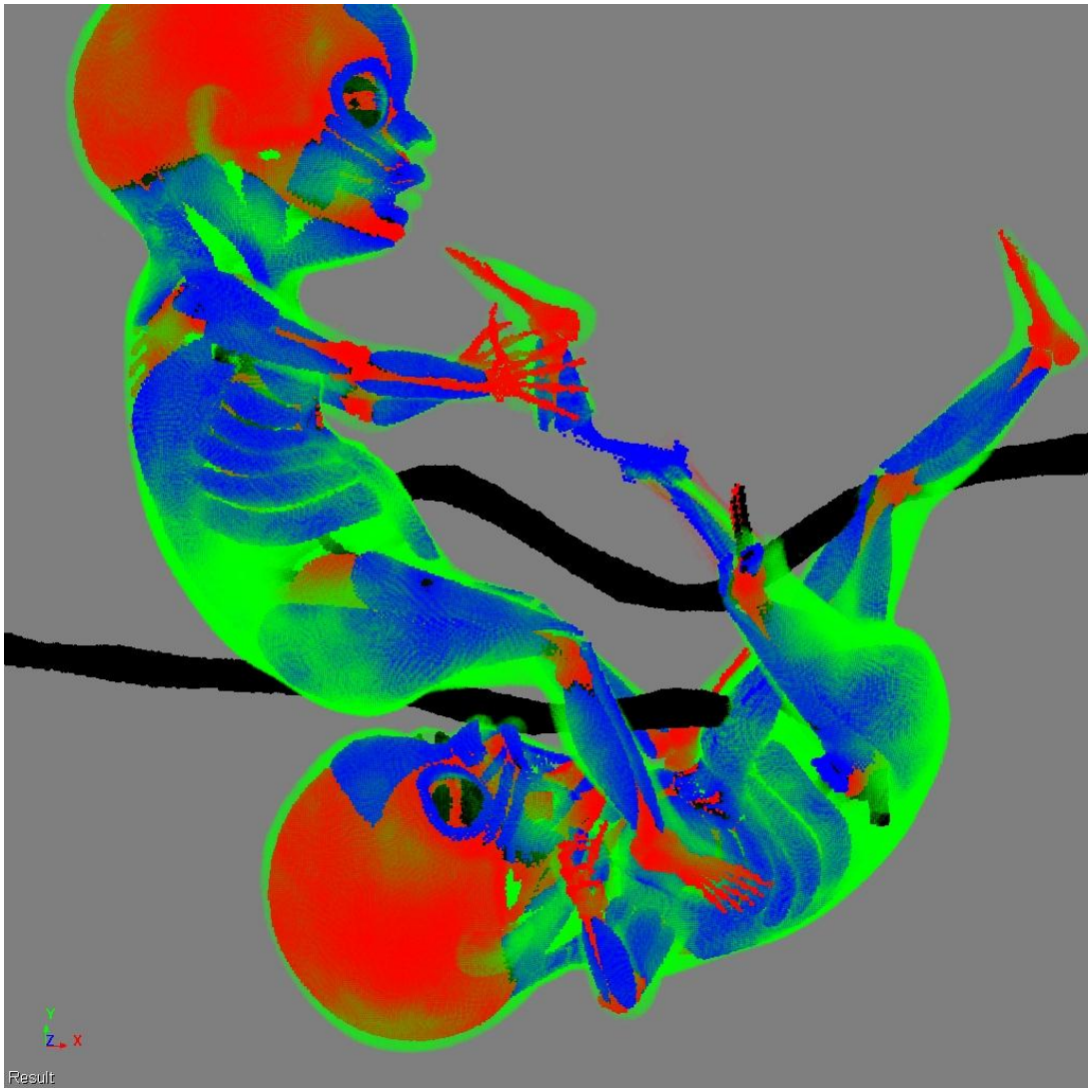


A-osion pisteiden kopioiva ylempi solmuryhmä ja alempi simulaatioattribuutit alustava solmuryhmä.

B-osiossa on ensimmäisessä solmuryhmässä ja sekä vasemmanpuoleisimmassa solmuryhmässä määritetty kuvaruutu, josta repeytyminen kussakin kuvassa alkaa. Keskimmäisessä solmuryhmässä pisteet simuloidaan Thiago Costan ICE:llä ja C++-ohjelmointikielellä kehittämällä “Lagoa Multiphysics”-liitännäisellä, jolla on mahdollista simuloida erilaisia aineiden fysikaalisia ominaisuuksia, kuten elastisuus, viskositeetti, pintajännite ja paine. Pisteille määritellään kudostyypit “Lagoa Phase”-paketeilla, joihin on kytketty “Lagoa Material”-paketit nimettyinä uudestaan kudosten mukaan; “Skin mat”,”Bone mat” ja “Muscle mat”. Lagoa-materiaalien asetukset on haettu pitkälti yrityksen ja erehdyksen kautta, jonka lisäksi ne ovat sidonnaisia Softimage-työtiedoston mittakaavaan. Koska erilaiset fysikaaliset materiaalit, joihin Lagoa Multiphysicsin attribuutit liittyvät ovat tyhjentävästi esitelty Softimagen käyttöohjeissa[25], en kertaa niitä tässä, mutta kuvakaappaukset materiaalien asetuksista löytyvät opinnäytetyöni liitteistä (Liite 1). Materiaalit pyrkivät jäljittelemään kudostyyppien ominaisuuksia etupäässä elastisuuden ja muodon säilyttämisen puitteissa, mikä tarkoittaa, ettei kaikki Lagoan materiaaliominaisuuksia käytetä niiden vähäisen kontribuution ja laskentahitauden ansiosta.



Tearing-pistepilven B-osiossa pisteet simuloidaan kolmen kudosmateriaalin ominaisuuksien mukaan (keskellä) ja valmistellaan Blood-pistepilveen siirrettävät pisteet jatkosimulointiin.



Lagoa Multiphysics-simulaatio eräässä animaation kuvassa. Ainoastaan repeytyvä osio pisteistä on mukana itse simulaatiossa. Kuvasta puuttuu verisimulaation pisteet.

“Lagoa Phase” -pakettiin kytketyt “Lagoa Transform Constraint” -paketit kertovat eri kudosten repeytymiskohdan ulkopuoliset pisteet repivän käden “Tearing Null”-primitiivin affiiniin kuvauksella liikuttaen niitä repivän käden mukana. On tärkeä huomata, että “Tearing Null”-primitiivien on repeytymiskuvaruudussa oltava kierroltaan yhteneviä yksikkömatriisin kierron kanssa, mutta niiden sijainnit ovat vapaavalintaiset.Tällä tavoin pisteet perivät sikiöhahmon repimisen liiikkeestä syntyvät voimat sekä “Lagoa Force”-paketin kautta kytketyn “Drag Force”-paketin hidastuvuusvastavoiman. Syntyneet voimat lasketaan yhteen “Lagoa Simulate Multiphysics” –solmussa, joka lopulta liikuttaa pisteitä kuvaruudusta toiseen. On

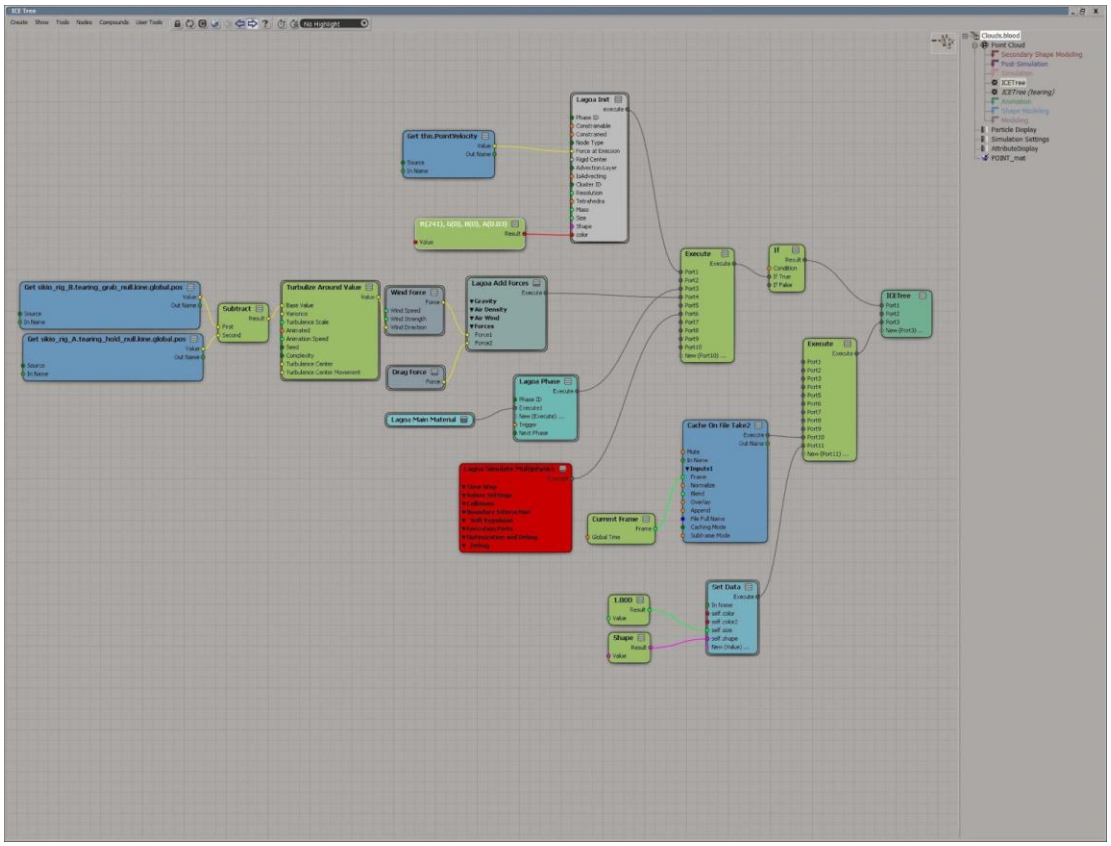
tärkeää huomata että kuvaruutujen välisten aliaskelten (*engl. substep*) tulee olla lukuisat johtuen kovista kiihtyvyksistä simulaatiolähtökohtana. Aliaskelten määrää säädetään em. “Lagoa Simulate Multiphysics”-solmusta käsin. Huomasin riittäväksi asetukseksi realistiselle kudosten repeytymistä muistuttavalle liikkeelle 120 aliaskelta jokaista animaatiokuvaruutua kohden. Aliaskeleet muuttavat olennaisesti sekä simuloitujen pisteiden liikettä ja laskenta-aikaa.

Alin solmuryhmä seuloo lopuksi repeytymishetkellä purskahtavan veripistepilveen soveltuvat pisteet, jotka tässä tapauksessa ovat kaikki simuloidut kudospisteet, jotka eivät on “Lagoa Transform Constraint”-paketin sidonnan piirissä, mikä testataan Boolen algebralla per piste. Pisteet lisätään “Add Point”-solmulla, jonka asetuksista kohdepistepilveksi on asetettu “Blood”-pistepilvi.

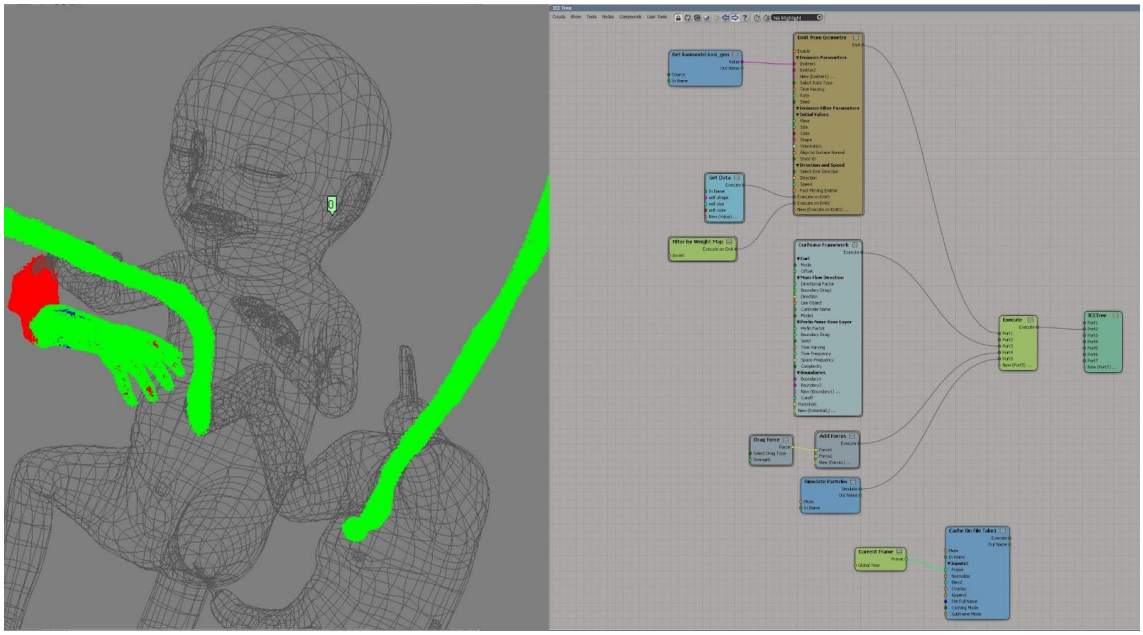
Kuvassa näkyy myös solmuryhmistä irrallisia kätkömuistisolmuja simulaation kovalevylle tallentamista varten sekä asetuksia, joilla pakotetaan tiettyjen attribuuttien evaluoituminen kätkömuistin kirjoittamisen yhtedessä. Attrbuutit ICE:ssä eivät aina tallennu kätkömuistiin, johtuen ICE:n optimoidusta luonteesta, joka poistaa lopputuloksen kannalta epäolennaiset attribuutit muistista.

4.4.2 Verisimulaatio

Verisimulaatio oli jaettu kuvakohtaisesti kahteen eri versioon “Clouds”-mallin “Blood”-pistepilvestä. Kehitin nopeamman järjestelmän ja työnkulun veren ajelehtimiselle lapsivedessä repeytymistä sisältämättömiä kuvia varten.
Ensimmäisessä repimiskuvia varten kehitetyssä em. tavalla “Tearing”-pistepilvestä seulotut ja kopioidut pisteet alustetaan tarvittavilla Lagoa-attribuuteilla “Lagoa Init”-paketissa. Pisteille määritetään väriarvo ja simuloinnin ensimmäistä kuvaruutua varten alustava kiihtyvyys. Samankaltaisella tekniikalla kuin napanuoria simuloidessa, vereen kohdistetaan voima, joka lasketaan “Tearing Null”-primitiivien sijaintien vähennyslaskuna. Tätä voimaa sattumanvaraistetaan ja yhdistettynä “Drag Force”-voimasolmuun sekä materiaaliominaisuuksiin ne yhdessä määrittävät pisteiden liikkeen. Materiaaliasetukset ovat vedelle tyypilliset keskittyen viskositeettiin ja paineeseen. Kätkömuistilla tallennetaan verisimulaation pisteet talteen kovalevylle.



Blood-pistepilvi repeytymistä sisältäviä kuvia varten. ICE-vuokaavio sovelsi Lagoan aidompaa nestesimuloinnin nimissä.



Blood-pistepilvi repimistä sisältämättömiä kuvia varten. Veren kaltainen liike saavutettiin nopeammin kuin Lagoa Multiphysicsillä renderoimalla.

Toinen optimoidumpi pistepilvi nousi aiheelliseksi, kun kävi ilmi, kuinka monessa kuvassa tarvittiin jatkuvuuden kannalta mahdollisimman samalla tavalla ajelehtivaa verta. Lagoa-nestesimulaatiot riittävällä määrällä pisteitä osoittautuivat odotetusti liian hitaiksi mielekästä työstämisnopeutta ajatellen. Lähdin tutkimaan yksinkertaisinta mahdollista ICE-vuokaaviota, joka lisäisi pisteitä määrättyihin kohtiin hahmojen monikulmioverkkoja ja irtoruuminosien monikulmioverkkoja. Tämä onnistui helposti “Emit from Geometry”-paketilla joka synnyttää pisteitä annetun monikulmioverkon pinnalle. Näitä pisteitä seulottiin “Filter by Weight Map”-paketilla, joka etsii lähimmän sijainnin pisteille annetun monikulmioverkon pinnalta ja katsoo on tässä kohden pintaa positiivisia arvoja ns. “Weight Map”-ominaisuudessa. Weightmapit ovat Softimagessa laajalti käytetty toiminto, jolla monikulmioverkkojen vertekseille voidaan määrittää liukulukuarvoja. Kuvassa näkyvälle irtokädelle olin maalannut tämän ominaisuuden ainostaan repeytymiskohdan vertekseille, joista kuvissa vuotaa verta. Pisteiden liike simuloitin toistuvasti erittäin käyttökelpoiseksi osoittautuneella “Curl Noise”[27]-paketilla, joka imitoi savun kaltaista liikettä pistepilvissä, mutta huomattavasti aitoa kaasusimulaatiota nopeammin. Curl Noisen luoma liike oli niin korkealaatuista, että jollei “Drag Force”-hidastuvuusvoimasolmua lasketa, mitään lisävoimia simulaatioon ei tarvittu realistisen ajelehtivan veren aikaansaamiseksi.Edellisten simulaatiovuokaavioiden tapaan saadut tulokset tallennettiin levylle kätkömuistin eli “Cache On File”-solmun avulla

5. Pohdinta ja yhteenveto

Etenkin projektin alussa erilaiset rasitustestit nousivat suureen asemaan, ja niissä olisi tullut olla huolellisempi. Vokselointijärjestelmässä itse pisteiden luonti ei muodostunut pullonkaulaksi, mutta testeissä olisi tullut käyttää raskaampia jo uudelleenjaettuja monikulmioverkkoja, sillä vokseloinnin nopeus oli riippuvainen monikulmioiden määrästä. Näin ollen syntyi aluksi käsitys nopeasta järjestelmästä, kunnes kävi ilmi, että järjestelmän toiminta hidastui lisääntyneiden monikulmioiden myötä nelinkertaiseksi, suoraan verrannollisesti suhteessa uudelleenjaettujen monikulmioiden määrään.

Toinen huomio liittyi lähikuvien arvuutteluun projektin alussa. Mikäli hahmot nähtäisiin todella läheltä, vokseloinnin pistetarkkuutta olisi nostettava. Vokseloitava

monikulmioverkko rajautuisi silti “Clone Mesh”-monikulmioverkkojen ICE-vuokaavioiden “Test Visibility From Camera”-paketin avulla oikein. Lisätarkkuus puolestaan hidastaisi simulointia ja mikä pahinta, muuttaisi simulaation liikettä. Tässä projektissa ei ollut erityisiä lähikuvia, mutta tämän henkinen automaattinen tarkkuuden ja simulaatioasetusten skaalautuminen suhteessa kuvakokoon on ilmeinen jatkokehitystarve.

Yksi lukuisista nopeutuksista projektissa oli napanuorien dynaamiikkojen yksinkertaistaminen. Napanuorat eivät törmäile toisiinsa eivätkä hahmoihin, vaan ne perivät liikkeensä perustuen yhden napanuoraa ohjaavan käyrän ensimmäisen ja viimeisen pisteen “naulaamiseen” vatsan alueelle sekä kuvarajauksen ulkopuolelle. Tämä nopeutti simulointia ja monissa kuvissa tämän realistisen ominaisuuden puute ei haitannut. Mikäli napanuorat ja kohdun ahtaus olisi haluttu nostaa enemmän esille, olisi järjestelmää tullut kehittää pidemmälle. Kohdun voimakkaampi läsnäolo olisi vääjäämättä haitannut hahmojen liikkumavaroja ja referenssinä käyttämämme virtuaalikohtu oli noin kaksi kertaa normaalikohtua suurempi.

Lagoa Multiphysics-pakettien parametrien saattaminen skaalautuviksi vokseloitavien monikulmioverkkojen mittakaavan muuttuessa olisi yksi olennaisimpia jatkokehitystarpeita, mitä tulee järjestelmän uudelleenkäytettävyyteen. Tällöin hahmojen kudosten dynamiikat säilyisivät samoina riippumatta kuinka suuri hahmo olisi Softimage-työtiedostoon. Materiaalien asetuksia voisi hioa ja viedä pidemmälle laskemalla kudosten sisäisiä rakenteita, kuten lihassyitä. Tämä voitaisiin toteuttaa yhdensuuntaistamalla pisteiden voimien laskentaa, jolloin pisteillä olisi naapureita vain lihassyyn suuntaisesti, muttei kaikkiin suuntiin, kuten Lagoassa normaalisti. lihassyiden ja muiden kudosrakenteiden vaikutus tulisi ilmeiseksi toki vasta lähikuvissa.

Twins-animaatiota työstäessä kävi ilmeiseksi, että vokselointi- ja kudossimulointijärjestelmän suurin jatkokehittämisen tarve piili kerronnallisen jatkuvuuden säilyttämisessä. Toisin sanoen kun hahmosta irtosi yhdessä kuvassa pala, seuraavassa kuvassa tuli käyttää hahmosta versiota, jossa vastaava pala oli mallintaen poistettu hahmon monikulmioverkosta . Hahmoversioita jouduttiin tekemään vain kahdeksan, ja niitä oli helppo käyttää referenssimallien resoluutioita

vaihtamalla, mutta mikäli kyseessä olisi ollut suuri määrä monimutkaisia repeämisiä, hahmoversioita ei olisi ollut mielekästä mallintaa jatkuvasti lisää. Tämä jatkuvuusongelma ei onneksi muodostunut kynnyskysymykseksi tuotannon läpiviennissä, mutta ilmenemisen ymmärtäminen ennalta olisi voinut ohjata tutkimustyötä toisille urille.

Täydellinen kerronnallisen jatkuvuuden säilyminen kuvasta toiseen edellyttäisikin monimutkaisemman ja laskennallisesti varsin raskaan järjestelmän. Monikulmioverkkojen sisäistä avaruutta ei vokseloitaisikaan joka ruudussa, vaan ainoastaan hahmon oletusasennossa. Tämä tarkoittaisi sitoutumista tiettyyn vokselointitarkkuuteen, jota ei voisi tarkentaa tai harventaa helposti saati nopeasti perustuen näkyvyyteen kameralle tai sijaintiin leikkaavan kappaleen sisässä . Pistepilvi kokonaisuudessaan painotettaisiin kiinni hahmorikin luiden affiineihin kuvauksiin, mikä itsessään jo edellyttäisi operaation ajamista ohjelmoinnillisesti haastavasti näytöohjaimella ns. verteksivarjostimena, jotta animointi olisi ylipäätään mahdollista hyväksi havaitulla noin 4-8 miljoonan pisteen tarkkuudella. Tämäkään ei olisi välttämättä riittävän nopeaa, ja on mahdollista, että vokselointi pitäisi suorittaa pienemmällä tarkkuudella ja lopuksi interpoloida vokselitarkkuus tarkemmaksi, mistä syntyisi taas vääjäämättä epätarkkuuksia. Ongelmaksi saattaisi myös muodostua ICE-ohjelmien ja näytönohjaimelle CUDA- tai OpenCL-rajapinnoilla ohjelmoitujen ohjelmien välinen pullonkaula, mikäli käytännössä koko järjestelmää ei rakennettaisi ICE:n ulkopuolella.

Työskentely olisi kuitenkin suoraviivaista. Hahmoanimaatioon perustuvassa repeytymissimulaatiossa tallennettaisiin kuvakerronnan kannalta otollisessa ajankohdassa pisteiden sijainnin muutos suhteessa oletusasentoon. Mikäli sijainnin muutos ylittäisi tietyn raja-arvon, eli piste olisi liikkunut riittävän kauaksi alkuperäisestä sijainnistaan suhteessa rikin luihin, tallennettaisiin pisteattribuutti joka yksilöisi muodostuneen uuden kappaleen. Lisäksi voitaisiin tallentaa muita pisteattribuutteja, kuten pisteen tila ja siihen vaikuttavat voimat kudossimulaatiossa. Näin ollen kohtauksen seuraavaa kuvaa työstäessä voitaisiin ladata uusimman hahmoanimaation lisäksi tieto edellisessä kuvassa pistepilveen määritellyistä uusista palasista. Näille palasille voitaisiin sitten palauttaa edellisen kuvan repeytynyt tila, luoda skriptillä uudet manipulaattoriobjektit animaattorin käytettäväksi tai pisteet

voitaisiin siirtää suoraan edelleensimuloitaviksi. Järjestelmä edellyttäisi luultavasti lukkoon lyötyä pistetarkkuutta, mikä olisi merkittävä rajoite sinällään.

Yhteenvetona voidaan todeta, että tutkituista vaihtoehtoisista totetustavoista kokonaisvaltaisesti pistepilviin perustuva järjestelmä tarjosi parhaan uudelleenkäytettävyyden ja työnkulun suhteessa tasa-arvopinnoilla monikulmioverkkoja ja pistepilviä yhdistelevään järjestelmään. Kuten edellä mainittiin, järjestelmälle ilmeni myös monia jatkokehittämismahdollisuuksia.

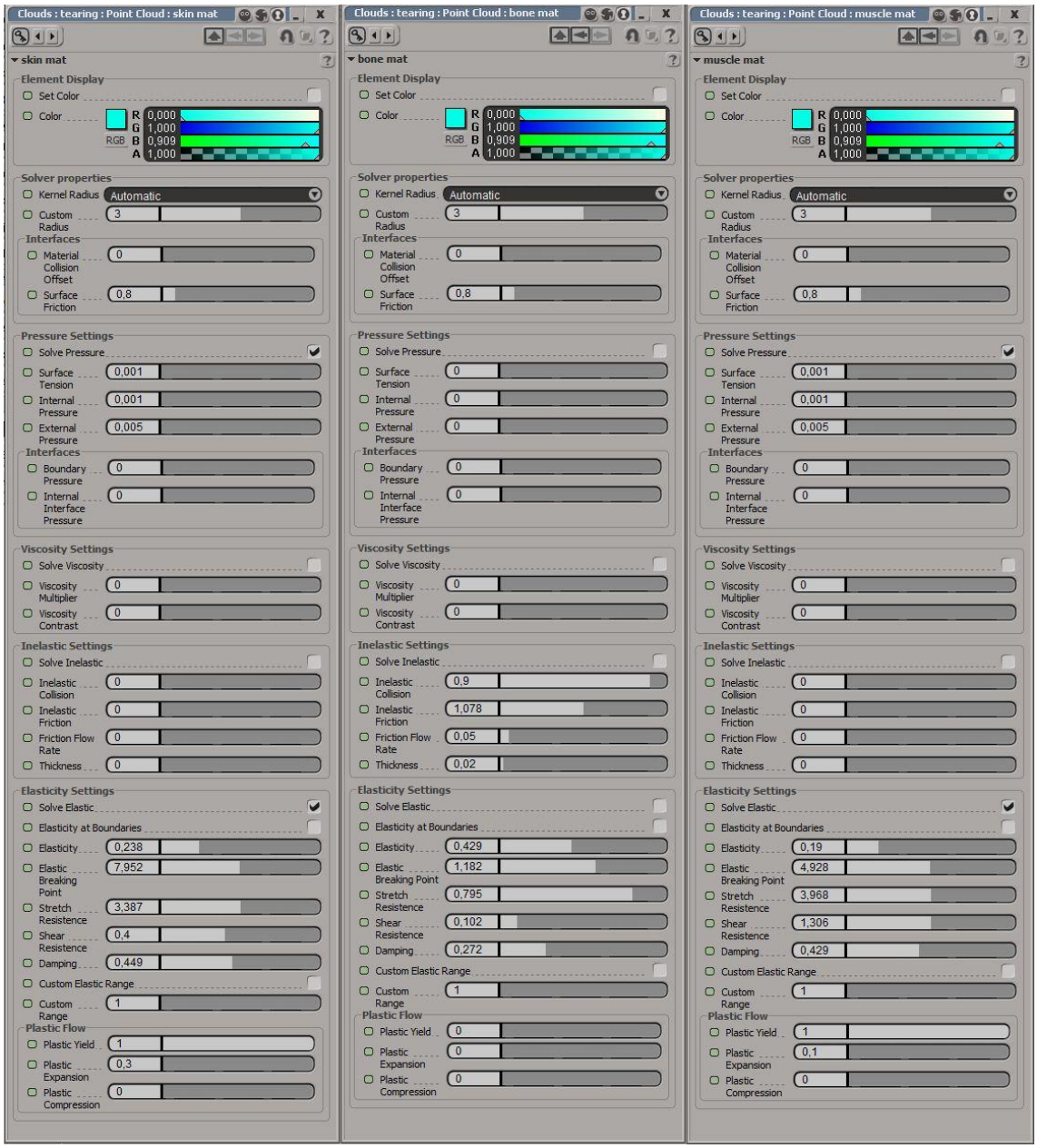
Vaikka vokselointiin perustuvissa työnkuluissa animaatiotuotannoissa on ilmeisiä heikkouksia, ovat ne verrattomia muuttuvan topologian ja tilavuuskuvantamisen saralla. Toivon näkeväni lisää vokseleihin perustuvia työnkulkuja ja ohjelmia kehitettävän alalla lähitulevaisuudessa.

Lähteet

1. Wikipedia, http://en.wikipedia.org/wiki/Point_(geometry, luettu 23.1.2012
2. Wikipedia, http://en.wikipedia.org/wiki/Voxel luettu 23.1.2012
3. Wikipedia, http://en.wikipedia.org/wiki/Point_cloud luettu 23.1.2012
4. Wikipedia, http://en.wikipedia.org/wiki/Polygon_mesh luettu 23.1.2012
5. Wikipedia, http://en.wikipedia.org/wiki/Edge_(geometry) luettu 23.1.2012
6. Wikipedia, http://en.wikipedia.org/wiki/Polygon luettu 23.1.2012
7. Wikipedia, http://en.wikipedia.org/wiki/Vertex_(computer_graphics) luettu 23.1.2012
8. Wikipedia, http://en.wikipedia.org/wiki/ISO_surface luettu 24.4.2012
9. Wikipedia, http://en.wikipedia.org/wiki/Marching_cubes luettu 7.2.2012
10. Wikipedia, http://en.wikipedia.org/wiki/Scalar_(computing) luettu 23.1.2012
11. Wikipedia, http://en.wikipedia.org/wiki/Vector_(geometric) luettu 23.1.2012
12. Wikipedia, http://en.wikipedia.org/wiki/Matrix_(mathematics) luettu 23.1.2012
13. Wikipedia, http://en.wikipedia.org/wiki/Transformation_matrix#Affine_transformations luettu 23.1.2012
14. Wikipedia, http://en.wikipedia.org/wiki/Rendering_(computer_graphics) luettu 24.1.2012
15. Wikipedia, http://en.wikipedia.org/wiki/Ray_tracing_(graphics) luettu 24.1.2012
16. Wikipedia, http://en.wikipedia.org/wiki/Volume_rendering#Splatting luettu 24.1.2012
17. Wikipedia, http://en.wikipedia.org/wiki/Volume_rendering luettu 24.1.2012
18. Wikipedia, http://www.abudhabiartfair.ae/en/programme-2011/exhibitions/Art-Talks-and-Sensations-Island-Artists-1/ luettu 30.1.2012
19. Wikipedia, http://en.wikipedia.org/wiki/Marching_cubes luettu 7.2.2012
20. http://www.sidefx.com/index.php?option=com_content&task=view&id=385&Itemid=190 luettu 20.2.2012
21. Takeuchi Hirotaka, Nonak Ikujiro 1986, Harvard business review, The new new product development game. http://mis.postech.ac.kr/class/MEIE780_AdvMIS/paper/part3/32_The%20new%20product%20development%20game.pdf luettu 20.2.2012

22. Wikipedia, http://en.wikipedia.org/wiki/Level_set_method, luettu 20.4.2012
23. http://en.wikipedia.org/wiki/Signed_distance_function, luettu 20.4.2012
24. Lagoa Material Properties http://download.autodesk.com/global/docs/softimage2013/en_us/userguide/index.html?url=files/lagoa_basics_LagoaMaterialProperties.htm,topicNumber=d30e309018 luettu 25.4.2012
25. Bridson Robert, Hourihan Jim, Nordenstam Marcus, 2007, Curl-Noise for Procedural Fluid Flow, http://www.cs.ubc.ca/~rbridson/docs/bridson-siggraph2007-curlnoise.pdf, luettu 24.4.2012

Liitteet



Liite 1: Lagoa Multiphysics-materiaalien asetukset kudostyypeittäin. Vasemalta oikealle Skin, Bone, Muscle