Java 3D:n hyödyntäminen elementtimenetelmämallinnuksessa

Kimmo Tuppurainen

07.12.2007

Joensuun yliopisto
Tietojenkäsittelytiede
Pro gradu –tutkielma

Tiivistelmä

Tämän tutkimuksen tarkoituksena on selvittää Java 3D-rajapinnan soveltuvuutta akustisen mallintamisen alueeseen kuuluvan elementtimenetelmän toteutukseen. Työllä on tarkoitus antaa valmiuksia elementtimenetelmien ymmärtämiseen ja akustisessa mallintamisessa käytettävän elementtimenetelmän soveltamiseen käytännössä. Tutkimuksen alussa tutkimuskenttään tutustutaan laajemmin kirjallisuuden avulla, jonka jälkeen perehdytään laskentaverkkoihin ja niiden generointimenetelmiin. Tutkimuksessa esitellään Java3D-ohjelmointirajapinta, jonka avulla toteutetaan laskentaverkkojen visualisointityökalu. Tutkimus osoittaa, että Java 3D antaa riittävät työvälineet elementtimenetelmiä koskevien ohjelmistojen ja työvälineiden toteuttamiseen. Koska Java 3D on korkean tason ohjelmointirajapinta, asettaa tämä kuitenkin omat rajoituksensa toteutukselle. Käytännön malliesimerkkinä tutkitaan ääniaallon siroamista kovasta kappaleesta käyttäen elementtimenetelmää.

ACM-luokat (ACM Computing Classification System, version 1998): I.6, I.3.5

Avainsanat: Akustinen mallintaminen, elementtimenetelmä, Java 3D, laskentaverkko

Sisällysluettelo

1. Johdanto

Tämän tutkimuksen tarkoituksena on selvittää Java 3D-rajapinnan soveltuvuutta akustisen mallintamisen alueeseen kuuluvan elementtimenetelmän toteutukseen. Työllä on tarkoitus antaa valmiuksia elementtimenetelmien ymmärtämiseen ja akustisessa mallintamisessa käytettävän elementtimenetelmän soveltamiseen käytännössä.

Akustiikka on fysiikan osa-alue, joka tutkii mekaanista aaltoliikettä kaasussa, nesteessä ja kiinteässä aineessa. Akustista mallintamista käytetään hyvin laajalti aina konserttisalien suunnittelusta ultraääni kirurgian tutkimukseen. Akustisessa mallintamisessa on useita osa-alueita, joista elementtimenetelmää käytetään yhä useammin ratkaisumenetelmänä, koska tietokoneiden laskennallinen tehon nousu on mahdollistanut sen.

Elementtimenetelmiä on käytetty aikaisemmin alun perin aeroakustiikan ongelmien ratkaisemisessa. Viime vuosien aikana akustinen mallintaminen on vallannut alaa myös erilaisilla osa-alueilla kuten teollisuudessa, lääketieteessä ja konetekniikassa. Mallintaminen on oiva työväline tutkimusalueilla, joissa halutaan selvittää aaltoliikkeeseen liittyviä ongelmia. Mallintaminen antaa monipuolisen ratkaisumenetelmän uusien tutkimusalueiden ja tutkimusten toteuttamiseen. Elementtimenetelmää käytettiin ensimmäisen kerran 1950-luvulla, jolloin insinöörit alkoivat ratkaista numeerisesti rakenteellisen mekaniikan ongelmia lentotekniikassa. Elementtimenetelmien käyttö eri sovellusaloilla on kasvanut tasaisesti vuosien saatossa. Nykyisin elementtimenetelmillä voidaan ratkaista myös tavallisia differentiaaliyhtälöitä (DY), integraaliyhtälöitä ja näiden yhdistelmiä integrodifferentiaaliyhtälöitä. Elementtimenetelmiä sovelletaan mm. rakenteiden mekaniikan, virtaustehtävien, lämmönsiirron, akustiikan, kvanttimekaniikan ja sähkömagnetiikan tutkimuksessa. Myös näiden yhdistelmiä voidaan ratkaista elementtimenetelmien avulla.

Elementtimenetelmästä löytyy aikaisempia tutkimuksia, opinnäytetöitä ja lehtiartikkeleita mutta kirjallisuutta rajoitetusti. Tutkimusmateriaalia Java 3D-rajapinnan käytöstä element-

timenetelmissä ei käytännössä löydy. Edellä mainittujen pohjalta on laadittu Java 3D-rajapintaa käyttäen elementtimenetelmään perustuva esikäsittelijä. Java 3D antaa riittävät työvälineet elementtimenetelmiä koskevien ohjelmistojen ja työvälineiden toteuttamiseen. Koska Java 3D on korkean tason ohjelmointirajapinta, asettaa tämä kuitenkin omat rajoituksensa toteutukselle.

Luku 2 perehdyttää lukijan akustiikan, akustisen mallintamisen perusteisiin ja käsitteistöön sekä elementtimenetelmiin. Luvussa 3 tutustutaan laskentaverkkoihin ja niiden generointimenetelmiin. Luvun lopussa luodaan katsaus laskentaverkkojen tiedostoformaatteihin ja olemassa oleviin laskentaverkkogeneraattoreihin. Luku 4 perehdyttää lukijan Java 3D-ohjelmointirajapintaan ja siinä luvussa esitetään konkreettinen esimerkki laskentaverkkojen visualisoinnin toteuttamisesta rajapinnan avulla. Luvun 4 lopussa ratkaistaan malliesimerkki käyttäen elementtimenetelmää. Luku 5 sisältää yhteenvedon painottuen tutkimustulosten loppupäätelmiin.

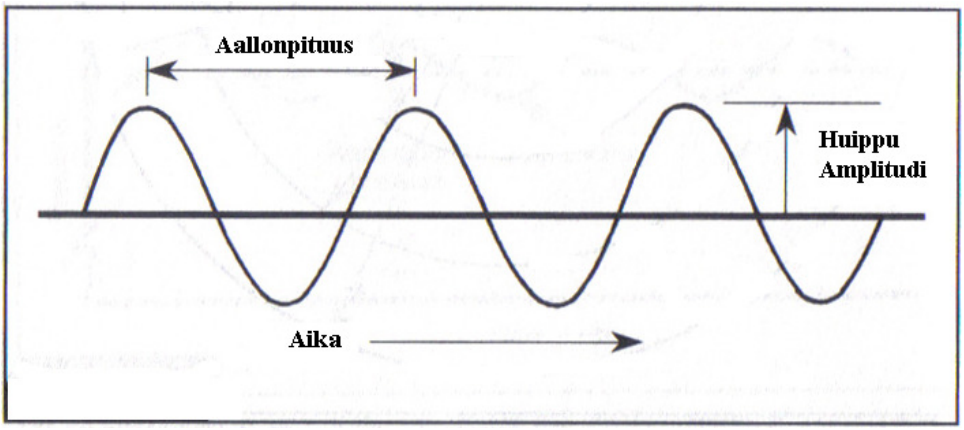
2. Matemaattinen mallintaminen akustiikassa

Matemaattisia malleja käytetään tieteessä ja tekniikassa, kun halutaan tietää, miten luonto tai jokin laite toimii. Näitä malleja ratkaistaan yleensä numeerisesti tietokoneella, koska käytännön tehtävissä analyyttisten ratkaisujen löytäminen on usein mahdotonta. Menetelmällä pyritään luotettavuuteen ja tarkkuuteen. Ratkaisun oikeellisuuteen vaikuttaa paljon se, kuinka menetelmä on ohjelmoitu tietokoneelle. Matemaattinen mallintaminen on haasteellista tutkimusta ja useasti voikin törmätä huonosti toimiviin numeerisiin menetelmiin. Tämä johtuu siitä, että matemaattisen mallin muuntaminen tietokoneessa käsiteltävään muotoon vaatii suurta tarkkuutta. Myös itse ratkaistava ongelma voi asettaa omat rajoituksensa ratkaisemisen onnistumiselle (Haataja & al., 2002).

Tässä luvussa kuvataan akustiikan ja akustisen mallintamisen peruskäsitteet. Luvussa perehdytään myös elementtimenetelmän perusteisiin ja lopuksi luodaan katsaus yleisimmin käytettyihin akustisen mallintamisen työkaluihin.

2.1 Akustiikan ja akustisen mallintamisen perusteet

Ääni voidaan yleisesti ottaen määritellä mekaanisena aaltoliikkeenä joko ilmassa tai muussa väliaineessa. Etenevän aaltoliikkeen värähtely synnyttää paineenvaihteluita, jotka korva aistii äänenä silloin kun värähtely saa tärykalvon liikkumaan. Ääni voidaan havaita mittaamalla, kuuloaistimuksena tai tuntoaistimuksena. Ääneen liittyvät mittayksiköt ovat hertsi ja desibeli ja suureita ovat aallonpituus, taajuus, amplitudi ja vaihe (Everest, 2001).



Kuva 1. Yksinkertainen siniaalto (Everest, 2001).

Kuvassa 1 on esimerkkinä yksinkertainen siniaalto. *Aallonpituus* (wavelength) tarkoittaa välimatkaa, missä aalto kulkee yhden täyden syklin. Sykli voidaan mitata esimerkiksi peräkkäisten aallon huippukohtien välisenä etäisyytenä tai minkä tahansa kahden peräkkäisen toisiaan vastaavan pisteen välisenä etäisyytenä aallossa. *Taajuus* (frequency) kuvaa aallon syklien lukumäärää sekunnissa ja taajuuden yksikkö on *hertsi* (Hz). Taajuus lasketaan jakamalla äänen nopeus aallonpituudella. Esimerkiksi, kun sekuntia kohden on 1 000 värähdystä, taajuus on 1 000 Hz eli 1 kHz. Yleisesti akustiikka liitetään kuultavaan ääneen taajuuksilla 20 - 20 000 Hz, mutta laajemmin ottaen se käsittää myös infraäänet, jotka ovat alle 20 Hz ja ultraäänet, jotka ovat yli 20 000 Hz.

Äänen nopeus ilmassa on noin 344 metriä sekunnissa. Äänen nopeuteen vaikuttavat ilmanlämpötila ja ilmanpaine. *Amplitudi* (amplitude) eli värähdyslaajuus ilmaisee värähdysliikkeen laajuutta. Amplitudi saadaan laskemalla värähtelyssä muodostuvien ääripisteiden etäisyys toisistaan ja jakamalla se kahdella.

Desibeli (dB) on akustiikassa käytetty äänenvoimakkuuden suhteellinen mittayksikkö. Desibeli-asteikon avulla pystytään helposti kuvaamaan kuulon erittäin laajaa herkkyysaluetta. Desibeli on dimensioton yksikkö, joka vertailee suureiden suhteita logaritmisella asteikolla. Yleensä desibelin yhteydessä käytetään 10-kantaista logaritmia. Äänenpaineen mittayksikkö on Pascal ja hiljaisin ääni, jonka ihmiskorva voi kuulla on 20 mikropascalia. Äänenvoi-

makkuus mitataan tutkittavan äänenpaineen p ja vertailupaineen p_0 suhteena. Vertailupaineena käytetään normaalisti 20 mikropascalia. Äänenvoimakkuuden logaritminen mitta saadaan kaavasta

$$L_p = 20\log_{10}(\frac{p}{p_0})$$

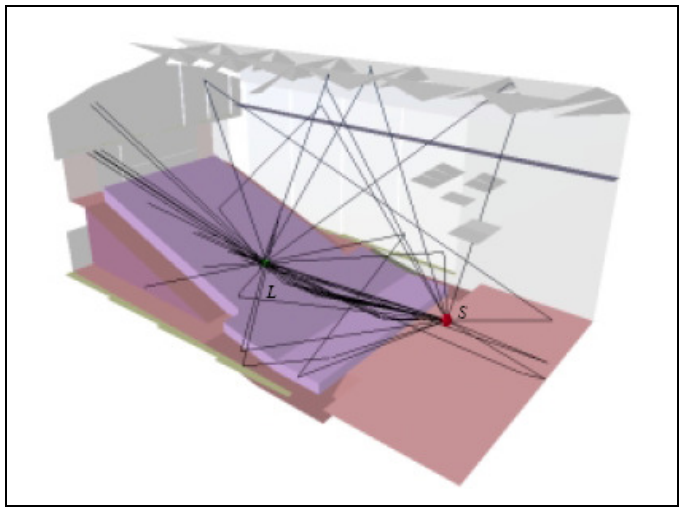
Ihmisen kuulo kykenee käsittelemään äänipainetasoja, jotka ovat välillä noin 0-130 dB. *Vaiheella* (phase) tarkoitetaan äänen aaltomuodon yhtä sykliä, jossa aalto käy nollakohdan molemmilla puolilla palaten nollakohtaan. Vaiheesta puhutaan yleensä asteina välillä 0-360, jossa teoriassa nolla astetta on aallon alkukohta, 90 astetta nousevan aallon korkein kohta eli amplitudi, 270 astetta laskevan aallon alin amplitudi ja 360 astetta jälleen nousevan aallon nollakohta, josta alkaa uusi sykli nolla asteesta. Kaksi samanlaista, mutta eri vaiheessa olevaa siniaaltoa voivat muodostavat yhdistelmän, jonka aallonmuoto voi olla hyvinkin erinäköinen kuin mitä alkuperäisillä oli. Muuttunut aalto voi kuitenkin kuulostaa samalle kuin mitä ennenkin (Everest, 2001).

2.1.1 Akustinen mallintaminen ja sen sovellusalueet

Akustisella mallintamisella tarkoitetaan jonkin tilan tai ilmiön äänimaailman mahdollisimman autenttista konstruointia. Tilojen akustista mallintamista tietokoneohjelmistojen avulla on tutkittu jo 1950-luvulta lähtien (Keränen 2006). Akustisten aaltojen matemaattista mallintamista hyödynnetään monilla fysiikan ja tekniikan osa-alueilla. Akustisen aallon etenemistä voidaan mallintaa esimerkiksi ilmassa, nesteissä ja kiinteissä aineissa. Tunnetuimpia esimerkkejä lienevät konserttisalien akustinen suunnittelu (kuva 2) tai autojen ja työkoneiden melun hallinta (Huttunen, 2004).

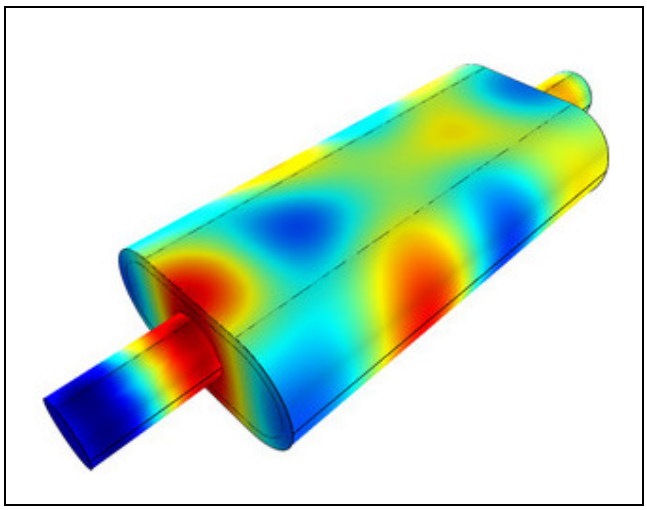
Menetelmän etuna on se, että sitä voidaan käyttää tilan suunnitteluvaiheessa, jolloin tilan soveltuvuus käyttötarkoitukseen nähden voidaan varmistaa ennen rakentamista. Tällöin uudelleensuunnittelu on vielä mahdollista (LMS International, 2007). Tilasta voidaan myös rakentaa pienoismalli, jota pystytään testaamaan visuaalisesti ja akustisesti suhteellisen tar-

kasti, mutta menetelmän kustannukset ja ajan tarve ovat suurempia kuin käytettäessä akustista mallinnusta. Lisäksi akustisen mallintamisen avulla voidaan joustavammin muokata tutkittavaan alueeseen liittyviä parametreja ja analysoida niiden aiheuttamia muutoksia lopputuloksessa (Stettner ja Greenberg, 1989).



Kuva 2. Äänisäteiden eteneminen konserttisalissa. Äänilähdettä kuvaa S ja kuuntelijaa L (Savioja, 1999).

Akustista mallintamista hyödynnetään autoteollisuudessa esimerkiksi auton sisätilojen äänenpainetasojen, moottorin melutason ja pakoputken äänenvaimentimeen liittyvässä kehitystyössä. Äänenvaimentimen mallintamista varten on Comsol Oy esimerkiksi kehittänyt Comsol Multiphysics ohjelmistoonsa mallin, jonka avulla voidaan simuloida ja kolmiulotteisesti visualisoida äänenpainetason etenemistä äänenvaimentimessa (kuva 3).



Kuva 3. Äänenpainetason jakautuminen äänenvaimentimessa (Comsol, 2007).

Meluntorjunnassa käytetään useasti akustista mallinnusta suunniteltaessa meluaitoja, teollisuustilan akustiikkaa tai suuria puhetiloja (Keränen, 2006). Laskennallisessa aeroakustiikassa akustista mallinnusta hyödynnetään muun muassa tutkittaessa lentokoneen turbiineiden aiheuttamaa melua. Muita aloja, joissa akustista mallinnusta hyödynnetään, ovat lääketiede ja akustiikkasensoreiden, kaiutinelementtien, kaikuluotainten ja tutkien suunnittelu (Comsol, 2007).

2.1.2 Aaltoyhtälöpohjaiset menetelmät

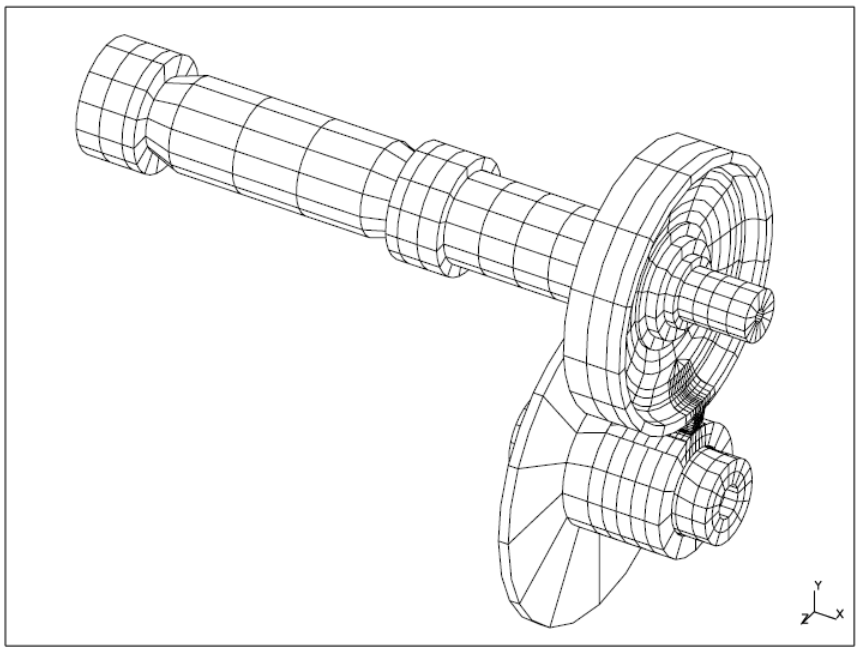
Tutkittaessa fysikaalisia, biologisia ja kemiallisia ilmiöitä tarvitaan useimmiten matemaattisia malleja, jotka sisältävät differentiaaliyhtälöitä. Differentiaaliyhtälöiden avulla voidaan kuvata ilmiöitä, jotka ovat liikkeessä tai muuttuvat ajassa. Fysikaaliset ilmiöt tapahtuvat kolmiulotteisessa avaruudessa, jolloin matemaattiset mallit saattavat sisältää differentiaaliyhtälöitä, joihin vaikuttaa useampi riippumaton muuttuja, kuten esimerkiksi ajan ja paikanmuuttujista. Tällaisia yhtälöitä kutsutaan *osittaisdifferentiaaliyhtälöiksi* (ODY) (Haataja & al., 1993). Osittaisdifferentiaaliyhtälöiden avulla pystytään simuloimaan useita fysikaalisia ilmiöitä.

Matemaattisesti äänen etenemistä kuvataan *aaltoyhtälöllä* (wave equation). Aaltoyhtälö on osittaisdifferentiaaliyhtälö, joka kuvaa aaltojen käyttäytymistä ajan t ja paikan (x,y,z) funktiona (Ballou, 2002). Koska aaltoyhtälössä on mukana aikariippuvuus, kutsutaan sen avulla tehtävää mallinnusta aikatason tarkasteluksi. Jos halutaan tutkia aalto-ongelman yksittäistä taajuuskomponenttia, voidaan aikariippuva aaltoyhtälö muuttaa Helmholtzin yhtälöksi, joka kuvaa ko. taajuuskomponentin käyttäytymistä vain paikan funktiona. Helmholtzin yhtälössä oletetaan siis aikariippuvuuden olevan aika-harmonista eli muotoa $sin(\omega t)$, missä ω on kulmataajuus, joka saadaan aaltokentän taajuudesta f siten että $\omega=2\pi f$. Helmholtzin yhtälöä kutsutaan yleisesti taajuustason aaltoyhtälöksi (Huttunen, 2004).

Aaltoyhtälöt ratkaistaan numeerisesti esimerkiksi käyttäen elementtimenetelmiä. Menetelmällä saadut tulokset ovat hyvin tarkkoja. Menetelmä ei sovellu korkeiden taajuuksien simuloimiseen, koska laskentaverkon elementtien määrä kasvaa huomattavasti äänen taajuuden mukana. Jos n kuvaa simuloitavan äänen taajuutta, tarvitaan $O(n^3)$ elementtiä, jotta laskentaverkko olisi tarpeeksi tiheä. Aikavaativuus ongelman ratkaisemiseen on yleensä $O(n^3 \log n^3)$ ja tästä syystä aaltoyhtälöön perustuvat menetelmät eivät sovellu kuin matalille taajuuksille (Deines & al., 2006).

2.2 Elementtimenetelmä

Numeerisessa mallintamisessa käytettäviä menetelmiä on useita, joista laajimmin yleisesti käytössä ovat *elementtimenetelmä* (Finite Element Method, FEM), *differenssimenetelmä* (finite difference) ja *kontrollitilavuusmenetelmä* (finite volume). Tässä tutkielmassa on tarpeellista keskittyä ainoastaan elementtimenetelmiin. Differenssimenetelmän toimintaperiaate on kuvattu Lewisin & al. (2004) teoksessa ja kontrollitilavuusmenetelmän toimintaperiaate on kuvattu Randalin (2002) teoksessa.



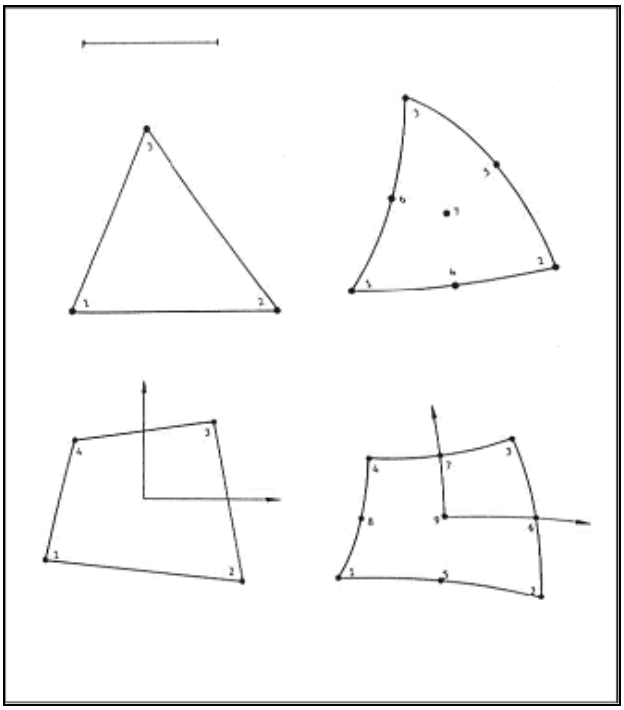
Kuva 4. Hammaspyörästön toiminnan tutkimiseen tarkoitettu laskentaverkko (Haataja & al., 2002).

2.2.1 Elementtimenetelmän perusteet

Elementtimenetelmä kehitettiin pääasiassa osittaisdifferentiaaliyhtälöiden numeeriseen ratkaisemiseen (Haataja & al., 2002). Elementtimenetelmää käytettiin ensimmäisen kerran 1950-luvulla, jolloin insinöörit alkoivat ratkaista numeerisesti rakenteellisen mekaniikan ongelmia lentotekniikassa (Ern ja Guermond, 2004). Lewis & al. (2004) kirjassaan huomauttaa, että elementtimenetelmien historiasta on kirjoitettu lukuisia artikkeleita, jotka sisältävät ristiriitaisia mielipiteitä tämän tekniikan alkuperästä. Elementtimenetelmien käyttö eri sovellusaloilla on kasvanut tasaisesti vuosien saatossa (Ern ja Guermond, 2004). Nykyisin elementtimenetelmillä voidaan ratkaista myös tavallisia differentiaaliyhtälöitä (DY), integraaliyhtälöitä ja näiden yhdistelmiä integrodifferentiaaliyhtälöitä (Haataja & al., 2002). Elementtimenetelmällä ratkaistavan ongelman yksi-, kaksi-, tai kolmiulotteisuus, aikariippuvuus tai epälineaarisuus ei rajoita menetelmää (Hämäläinen ja Järvinen, 2006). Elementtimenetelmiä sovelletaan mm. rakenteiden mekaniikan, virtaustehtävien, lämmönsiirron,

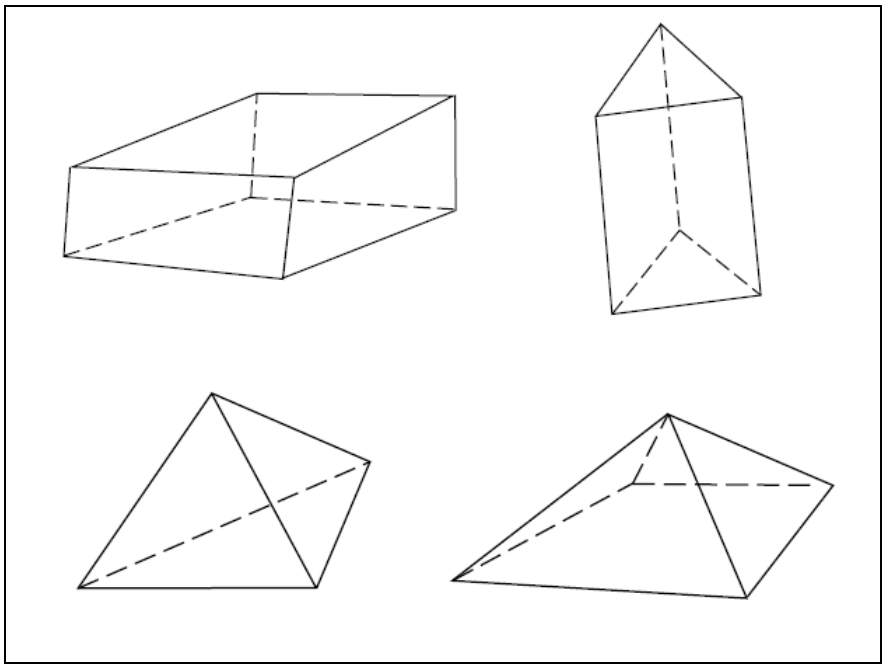
akustiikan, kvanttimekaniikan ja sähkömagnetiikan tutkimuksessa (Haataja & al., 2002). Myös näiden yhdistelmiä voidaan ratkaista elementtimenetelmien avulla (Kärkkäinen, 2003).

Ilmiön simulointia varten tarvitaan malli alueesta, jota kutsutaan *laskenta-alueeksi* (domain). Elementtimenetelmässä laskenta-alue jaetaan pienempiin osiin, joita on aina äärellinen määrä. Näitä pienempiä alueita kutsutaan elementeiksi. Elementteihin jaettua laskenta-aluetta kutsutaan tässä tutkielmassa *laskentaverkoksi* ja jaon muodostamista verkon *generoinniksi* (mesh generation). Kuva 4 sisältää esimerkkinä hammaspyörästön toiminnan tutkimiseen generoidun laskentaverkon. Muita yleisiä käytettyjä termejä kirjallisuudessa laskentaverkolle ovat esimerkiksi elementtiverkko tai verkko (grid). Elementti koostuu *solmupisteistä* (node) ja elementin sivuista tai tahkoista kun kyseessä on kolmiulotteinen elementti (kuva 5). Tässä tutkielmassa solmupisteiksi kutsutaan elementin geometrian määrääviä pisteitä.



Kuva 5. Yksi- ja kaksiulotteisia elementtityyppejä (Hämäläinen ja Järvinen, 2006).

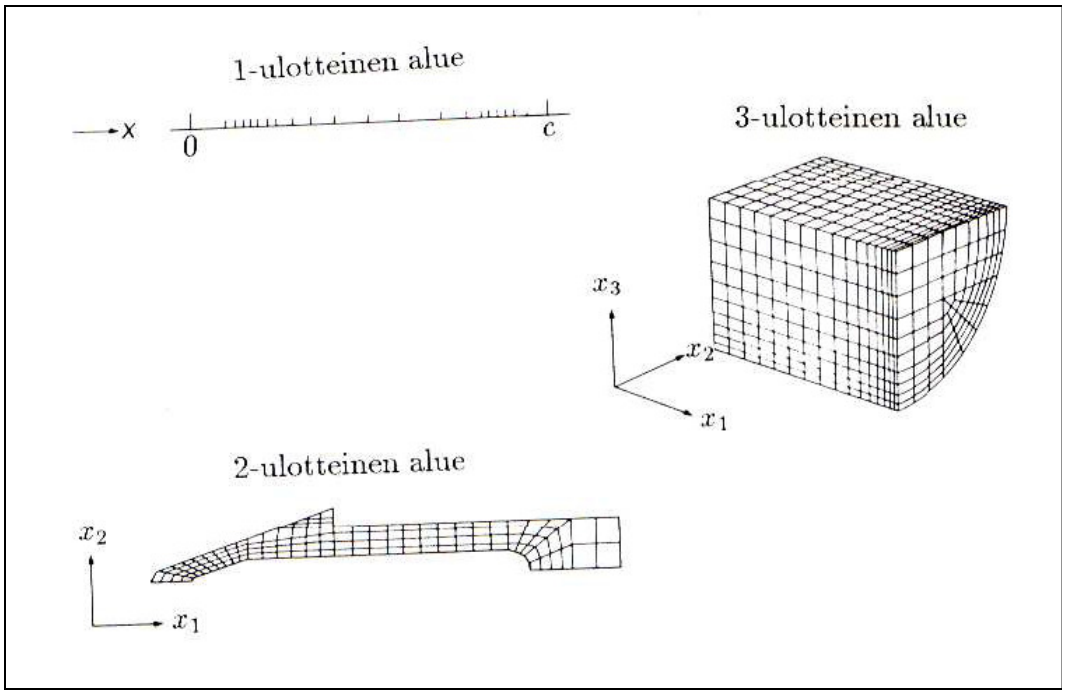
Yksiulotteiset elementit ovat janoja, joiden päätepisteitä ovat solmupisteet. Jana voi sisältää myös solmupisteistä koostuvia sisäpisteitä. Kaksiulotteisessa elementissä solmupisteet sijaitsevat yleensä elementin sivujen kärkipisteissä sekä mahdollisesti sisäpisteissä ja sivujanoilla. Yleisimpiä kaksiulotteisia elementtejä ovat kolmio tai nelikulmio. Kolmiulotteisella elementillä tarkoitetaan kolmiulotteista kappaletta (Hämäläinen ja Järvinen, 2006). Yleisimpiä käytettyjä elementtejä kolmiulotteisessa laskenta-alueessa ovat tetraedri, heksaedri, prisma ja pyramidi (Toivanen, 2004).



Kuva 6. Esimerkkejä kolmiulotteisista elementeistä. Ylärivillä kuusitahokas ja prisma sekä alarivillä nelitahokas ja pyramidi (Toivanen, 2004).

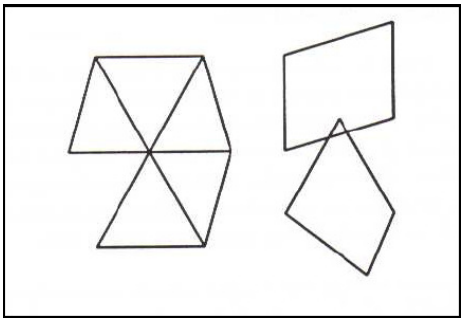
Kuva 6 sisältää esimerkkejä kolmiulotteisista elementtityypeistä. Tetraedri eli nelitahokas on nelitahkoinen geometrinen kappale, joka on yksinkertaisin viidestä mahdollisesta säännöllisestä monitahokkaasta. Heksaedri eli kuusitahokas on kuusitahkoinen geometrinen kappale, jonka tahkot ovat neliöitä. Pyramidi on monitahokas, jonka sivutahkot ovat kolmioita. Tällaisissa elementeissä solmupisteinä ovat pintojen tai tasojen kärkipisteet ja mahdollisesti sisäpisteet (Hämäläinen ja Järvinen, 2006). Kolmiulotteisen laskenta-alueen laatimi-

nen tarvitsee huomattavasti enemmän informaatiota kuin kaksiulotteisen laskenta-alueen laatiminen ja tästä syystä myös laskennallinen työmäärä kasvaa huomattavasti (Lewis & al., 2004).



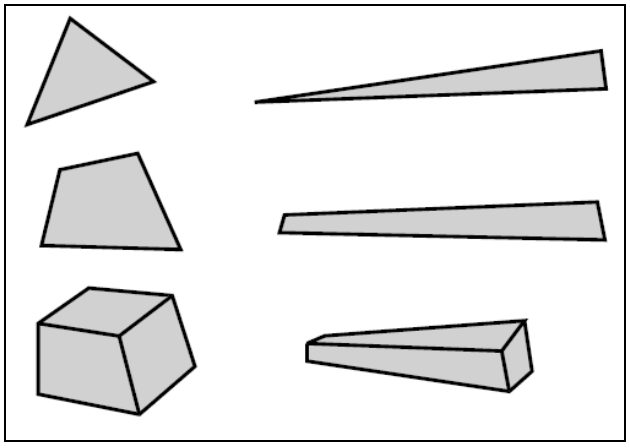
Kuva 7. Yksi-, kaksi-, ja kolmiulotteisia laskentaverkkoja (Hämäläinen ja Järvinen, 2006).

Kuvassa 7 on esimerkkejä yksinkertaisista yksi-, kaksi- ja kolmeulotteisista laskentaverkoista. Laskentaverkko ei saa missään tilanteessa sisältää elementtejä, jotka ovat toistensa sisällä. Kahden vierekkäisen elementin väliin ei myöskään saa jäädä tyhjää tilaa. Laskentaverkon tulee aina olla yhtenäinen ja tiivis (Toivanen, 2004). Kuva 8 sisältää esimerkkejä kielletyistä tilanteista elementeille (Hämäläinen ja Järvinen, 2006).



Kuva 8. Kielletyt tilanteet vierekkäisille elementeille (Hämäläinen ja Järvinen, 2006).

Elementtimenetelmät asettavat käytettävälle verkolle tiettyjä vaatimuksia. Menetelmällä saatujen tulosten oikeellisuuteen vaikuttavat laskentaverkossa käytetty elementtityyppi, laatu ja koko. Elementtiverkkoa suunniteltaessa tulisi aina valita mahdollisimman yksinkertainen elementtigeometria, jota voidaan käyttää ratkaisemaan haluttu ongelma. Myös elementtien kokosuhteisiin tulee kiinnittää huomiota (Toivanen, 2004). Elementin sivusuhteiden tulisi olla lähellä toisiaan eli niissä ei saa esiintyä äärimmäisen pieniä kulmia (kuva 9) (Felippa, 2004).



Kuva 9. Vasemmalla hyviä elementtejä sivusuhteeltaan ja oikealla huonoja (Felippa, 2004).

2.2.2 Elementtimenetelmät akustiikassa

Aaltomekaniikan ongelmia ratkaistaessa elementtimenetelmällä tulee erityisesti ottaa huomioon elementtien koko laskentaverkossa. Laskentaverkon pitää olla riittävän tiheä suhteessa ratkaistavan aallon aallonpituuteen. Normaaleille elementeille nyrkkisääntönä on kymmenen elementtiä per aallonpituus.

Jos mallinnettava aaltokenttä sisältää useita aallonpituuksia, ei nyrkkisääntö ole riittävä vaan laskentaverkon tulee olla tätäkin tiheämpi. Tämä johtuu aaltojen vaihevirheen kumuloitumisesta, joka tunnetaan *numeerisena saastumisena* (numerical pollution) (Ihlenburg, 1998).

Seuraava esimerkki havainnollistaa aallon taajuuden vaikutusta laskentaverkon elementtien lukumäärään. Määritetään, että äänennopeus ilmassa on 340 metriä sekunnissa ja tutkittavan aallon taajuus on 1700 Hz. Laskentaverkon elementtityyppinä käytetään heksaedriä. Aallonpituus voidaan laskea jakamalla äänennopeus aallon taajuudella, eli

$$aallonpituus = \frac{340\frac{m}{s}}{1700\frac{1}{s}} = 0.2m$$

Jos halutaan mallintaa huone, jonka koko on 4x3x2,5 metriä ja tiedetään, että kymmenen elementtiä per aallonpituus on optimaalinen elementtien esiintymistiheys, voidaan yksittäisen elementin koko laskea jakamalla aallonpituus kymmenellä, eli

$$\frac{0.2m}{10} = 2cm$$

Kun yksittäisen elementin koko on 2 cm, mallinnettavan huoneen laskentaverkko tarvitsisi tällöin 3,75 miljoonaa (200 x 150 x 125) heksaedri elementtiä. Tällaisen laskentaverkon

luominen on vaativa tehtävä. Jos tutkittavan aallon taajuus olisi 17000 Hz, tarvittava elementtien lukumäärä kohoaisi 3,75 miljardiin. Jos elementtityyppinä käytettäisiin esimerkiksi nelitahokasta, tarvittava elementtien lukumäärä vähintäänkin kaksinkertaistuisi.

2.3 Mallinnustyökalut

Mallinnustyökalut ovat ohjelmistoja, joita käytetään simuloimaan reaalimaailmassa tapahtuvia ilmiöitä. Ensimmäiset varsinaiset laskentaohjelmistot olivat yhden alueen erikoisohjelmistoja, joilla ratkottiin vaikkapa rakenteiden kestävyyttä, virtausten kulkeutumista tai sähkömagneettisia kenttiä. On kuitenkin olemassa ilmiöitä, joita ei pystytä kuvaamaan toisistaan riippumattomina, vaan ne tapahtuvat yhtä aikaa muiden ilmiöiden kanssa. Tämä synnytti tarpeen useiden ilmiöiden yhtäaikaiseen kuvaukseen soveltuville ohjelmistoille. Tällaisia ohjelmistoja ryhdyttiin kutsumaan nimellä *monifysikaalinen* (multiphysics). Nykyään monifysikaalisia ohjelmistoja on jo olemassa suuri joukko, esimerkiksi Waveller, ELMER ja Comsol Multiphysics (Råback ja Vierinen, 2005).

2.3.1 Waveller

Waveller on Kuava Oy:n kehittämä aaltokenttien laskentaan käytettävä ohjelmisto. Ohjelmalla voidaan simuloida ultraääniin, akustiikkaan ja sähkömagnetiikkaan liittyviä ilmiöitä. Waveller on suunniteltu vähentämään laskentaan kuluvaa aikaa ja muistinkäyttöä. Ohjelman ydin perustuu uuteen numeeriseen menetelmään, ultraheikkoon variaatioformalismiin, joka on edistyksellinen menetelmä aaltokenttien ratkaisemiseksi. Tämä mahdollistaa hyvin tarkat simulaatiot realistisissa laskenta-alueissa. Waveller-ohjelmistoon kuuluu myös kaukokenttien laskentamoduuli, jolla pystytään laskemaan esimerkiksi tutkaheijasteita sekä ihmisen kuuloon tai meluun liittyviä ongelmia (Numerola, 2007). Laskennallisesti suurien ongelmien ratkaisemiseen Waveller-ohjelmistosta on saatavilla myös versio, joka tukee

rinnakkaislaskentaa. Ohjelmistoa voidaan käyttää sekä teknisen laskennan palveluihin että asiakkaille toimitettaviin simulaattoreihin ja muihin ohjelmistoratkaisuihin (Kuava, 2005).

2.3.2 ELMER

ELMER on elementtimenetelmään pohjautuva monifysikaalinen mallinnusohjelmisto, jonka kehitystä on johtanut tieteen tietotekniikan keskus CSC. Rakenteeltaan ELMER koostuu kolmesta komponentista; *esikäsittelijästä* (Front), *ratkaisijasta* (Solver) ja *jälkikäsittelijästä* (Post) (CSC, 2007).

Kehitystyöhön osallistui joukko Suomen yliopistoja, tutkimuslaboratorioita ja teollisuuden yrityksiä. Vuonna 1995 aloitettu kehitystyö keskittyi pääasiassa virtauslaskentaan, mutta myöhemmin ominaisuuksia laajennettiin kattamaan muita fysiikan osa-alueita (CSC, 2000a). Nykyään ELMER-mallinnusohjelmisto sisältää lähes kaksikymmentä lämmönsiirtoon, sähkömagnetiikkaan, virtausmekaniikkaan, akustiikkaan, rakenneanalyysiin ja kvanttimekaniikkaan liittyvää fysikaalista mallia.

Malleja kuvataan osittaisdifferentiaaliyhtälöillä, jotka ratkotaan elementtimenetelmällä. ELMER-jälkikäsittelijää voidaan käyttää yleisenä jälkikäsittelijänä elementtimenetelmällä tuotettujen tulosten visualisointiin. ELMER-mallinnusohjelmistoa voidaan käyttää pöytäkoneilla, MPI-pohjaisilla rinnakkaislaskentajärjestelmillä sekä grid laskenta-alustoilla. Ohjelmisto sisältää yleisimmät elementtimenetelmään liittyvät peruspalvelut omina kirjastoinaan. Ohjelmiston heikkona puolena voidaan mainita käyttöliittymän vaatimaton taso. Vuonna 2005 tieteen tietotekniikan laitos julkaisi ELMER-mallinnusohjelmiston GPL avoimen koodin lisenssin alla. Tarkoituksena oli lisätä ohjelmiston käyttöä ja parantaa ohjelmiston käännettävyyttä eri alustoille (Råback ja Vierinen, 2005). ELMER toimii Unix, Linux ja Windows ympäristöissä (Haataja & al., 2002).

Esikäsittelijällä voidaan generoida laskentaverkkoja, rakentaa matemaattisia malleja graafisesti ja tuottaa tarvittava informaatio ratkaisijaa varten. ELMER-ratkaisija prosessoi matemaattiset mallit diskreettiin muotoon, hoitaa ongelman laskennalliseen puoleen liittyvät rutiinit ja tuottaa jälkikäsittelijälle tulokset. Jälkikäsittelijä visualisoi numeeriset tulokset (CSC, 2000b).

2.3.3 COMSOL Multiphysics

Comsol Multiphysic on kehitetty osittaisdifferentiaaliyhtälöiden numeeriseen ratkaisemiseen tarkoitettu ohjelma. Ohjelmisto oli alun perin nimeltään FEMLAB, mutta se muutettiin Comsol Multiphysics nimiseksi version 3.2 julkaisun ohella. Ohjelmiston avulla voidaan mallintaa eri sovellusalojen ongelmia itsenäisinä tehtävinä tai kytkeä eri sovelluksia toisiinsa. Yhtälöiden numeeriseen ratkaisemiseen käytetään elementtimenetelmää. Ohjelmisto sisältää malleja moniin eri sovelluksiin, kuten esimerkiksi aaltoyhtälöihin, sähkömagnetiikkaan, diffuusioilmiöihin, lämmönsiirtoon, virtausmekaniikkaan ja rakenneanalyysiin. Comsol tarjoaa Multiphysics ohjelmistolleen myös lisämoduuleita, jotka ovat räätälöity tukemaan eri tieteenalojen tutkimustyötä. Tällaisia moduuleita on julkaistu useita tukemaan esimerkiksi kemian tai akustiikan osa-alueita (Comsol, 2006).

Comsol Multiphysics tarjoaa myös erillistä CAD-moduulia ohjelmistoonsa, jonka avulla ohjelmistoon voidaan tuoda malleja useista eri CAD-tiedostoformaateista. Moduuli tukee IGES, SAT, Parasolid ja Step tiedostoformaatteja (Comsol, 2006).

3. Laskentaverkot

Akustisen mallintamisen ytimenä on aina laskenta-alue, joka kuvaa ongelman fyysistä tilavuutta. Tietokoneavusteisen suunnittelun avulla luodaan suurin osa akustisessa mallintamisessa käytetyistä laskenta-alueista. Jotta elementtimenetelmän avulla voidaan ratkaista haluttu ongelma, muodostetaan laskenta-alueesta laskentaverkko. Laskentaverkon muodostamiseen eli generoimiseen on kehitetty lukuisia ohjelmistoja ja algoritmeja.

Tässä luvussa luodaan katsaus siihen kuinka laskenta-alueita muodostetaan. Tämän jälkeen kuvataan laskentaverkkoihin liittyvät peruskäsitteet. Seuraavaksi tässä luvussa kuvataan yleisimmät laskentaverkkojen generointimenetelmät ja esitellään kaksi yleistä laskentaverkkotiedostoformaattia. Lopuksi luodaan katsaus laskentaverkkojen generointi ohjelmistoihin.

3.1 Tietokoneavusteinen suunnittelu elementtimenetelmien apuvälineenä

Tietokoneavusteinen suunnittelu eli CAD (Computer Aided Design) on tietokoneen hyödyntämistä apuvälineenä etenkin insinöörien ja arkkitehtien harjoittamassa suunnittelutyössä. Erilaisia ja eri tarkoituksiin käytettäviä CAD-ohjelmistoja löytyy markkinoilta nykyään kymmeniä. Melkein kaikki CAD-ohjelmistot ovat kehittyneet käyttämään samankaltaisia sisäisiä esitystapoja malleilleen.

Yksinkertaiset laskenta-alueet voidaan tuottaa suoraan suunnittelijoiden piirustuksista, mutta monimutkaisempien laskenta-alueiden tuottamiseen joudutaan käyttämään muita menetelmiä. CAD-ohjelmistojen avulla tuotetaan suuri osa laskenta-alueista, joita käytetään elementtimenetelmissä (Thompson & al., 1999). Laadukkaan *laskentaverkon generointi* (mesh generation) laskenta-alueesta on yksi tärkeimmistä tehtävistä elementtimenetelmässä (George, 1992). Laskentaverkon generoinnin merkittävä ongelma on, kuinka CAD-

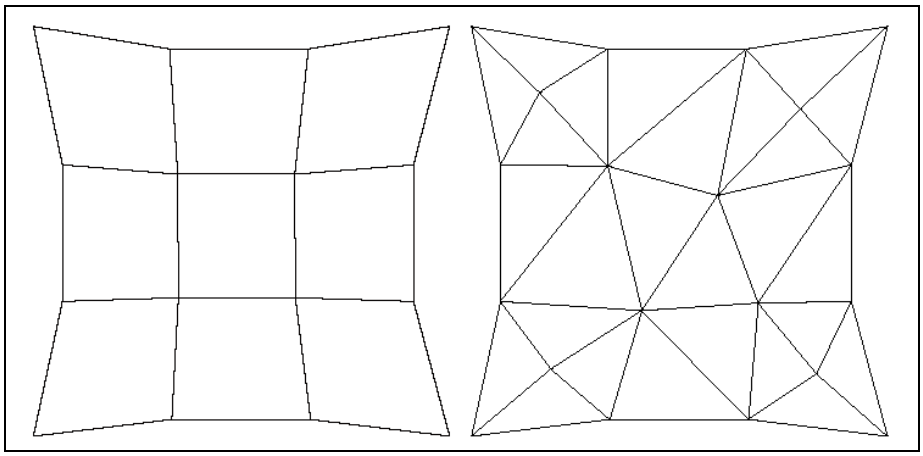
geometriaa pystytään käsittelemään tehokkaasti ja tarkasti. CAD-ohjelmistolla mallinnetun kolmiulotteisen mallin muuntaminen laskentaverkkorakenteeksi on pitkään ollut haasteellinen prosessi (Beall & al., 2003).

Tietokoneavusteisessa suunnittelussa geometristen elementtien muoto on ensiarvoisessa asemassa suunnittelijoille ja insinööreille. Laskentaverkkoja saadaan erilaisista lähteistä, kuten esimerkiksi kolmiulotteisista skannereista, mallinnusohjelmistoista tai tuotettuna tietokone- algoritmeillä. Vaikkakin nämä laskentaverkot voivat olla geometrialtaan tarkkoja, voivat niiden laadulliset kriteerit olla kaukana ideaalisesta. CAD-ohjelmistolla luodut kolmiulotteiset mallit sisältävät tasaisilla pinnoilla yleensä suuria määriä soveltumattomia ns. pitkiä kolmioita. Monet numeerisen laskennan menetelmät, kuten elementtimenetelmä, tarvitsevat suhteellisen yhdenmukaisia elementtejä sisältävän laskentaverkon, jotta voidaan minimoida numeeriset virheet (Yue & al., 2007).

3.2 Laskentaverkon perusteet

Laskentaverkkojen generointi on kehittynyt tasaista tahtia viimeiset kolme vuosikymmentä. Laskentaverkkojen generoinnista on kehittynyt oma tieteenala, johon on vaikuttanut vahvasti matematiikka ja tietojenkäsittelytiede. Mallien ja laskentamenetelmien kehittyminen on johtanut myös mallinnettavien geometrioiden monimutkaistumiseen (Thompson & al., 1999). Erilaisia verkon generointiohjelmistoja ja algoritmeja on nykyisin olemassa kymmeniä (Owen, 1998). Laskentaverkon generoinnissa tulee kiinnittää huomiota seuraaviin asioihin. Generoitu laskentaverkko tulee olla tarpeeksi tiheä, jotta numeerinen ratkaisu on tarkka. Mikäli generoitu laskentaverkko on liian tiheä, voi ratkaisun saaminen osoittautua epäkäytännölliseksi. Tarvittaessa voidaan laskentaverkon osa-alueita tihentää numeerisen ratkaisun tarkentamiseksi. Useat elementtimenetelmään perustuvat ohjelmistot osaavat automaattisesti tihentää laskentaverkkoa laskennan kannalta kriittisillä osa-alueilla, esimerkiksi kohdissa, joissa on nopea virtaus. Laskentaverkkoa generoitaessa tulee kiinnittää myös huomiota ratkaisun laskennalliseen tehokkuuteen (Thompson & al., 1999).

Laskentaverkkoja on kahta perustyyppiä, *rakenteellisia* (structured) ja *rakenteettomia* (unstructured). Rakenteellisilla laskentaverkoilla on toistuva ja säännöllinen rakenne (Toivanen, 2004). Rakenteellisen laskentaverkon tunnistaa siitä, että kaikilla sisäisillä solmuilla on yhtä monta vierekkäistä elementtiä (Owen, 1998). Kuvassa 10 on esimerkkinä kaksiulotteinen rakenteellinen ja rakenteeton laskentaverkko.



Kuva 10. Esimerkki laskenta-alueesta, joka on verkotettuna rakenteellisella nelikulmio- ja rakenteettomalla kolmioverkolla (Toivanen, 2004).

Rakenteellisilla laskentaverkoilla on tiettyjä etuja verrattuna rakenteettomiin laskentaverkkoihin. Ne ovat yksinkertaisempia ja käytännöllisempiä käytettäessä yksinkertaisia elementtimenetelmiä. Rakenteelliset laskentaverkot tarvitsevat vähemmän laskennassa käytettävää muistia, koska elementtien koordinaatit voidaan laskea, eikä niitä tarvitse erikseen säilöä. Rakenteellisten laskentaverkkojen elementtien kokoa ja geometrista muotoa voidaan kontrolloida paremmin kuin rakenteettomien. Heikkoutena rakenteellisilla laskentaverkoilla on niiden huono soveltuvuus monimutkaisten geometrioiden esittämiseen (Du ja Hwang, 1992). Rakenteettomassa laskentaverkossa yksittäisellä solmulla voi olla mielivaltainen määrä vierekkäisiä elementtejä (Owen, 1998).

Useasta rakenteellisesta osasta muodostuvaa laskentaverkkoa kutsutaan *blokeittain rakenteelliseksi* (block-structured) ja rakenteellisia sekä rakenteettomia osia sisältäviä laskentaverkkoja *hybridiverkoiksi* (hybrid grid) (Thompson & al., 1999). Tässä tutkielmassa ei käsitellä blokeittain rakenteellisia laskentaverkkoja tai hybridiverkkoja.

3.3 Laskentaverkon generointimenetelmiä

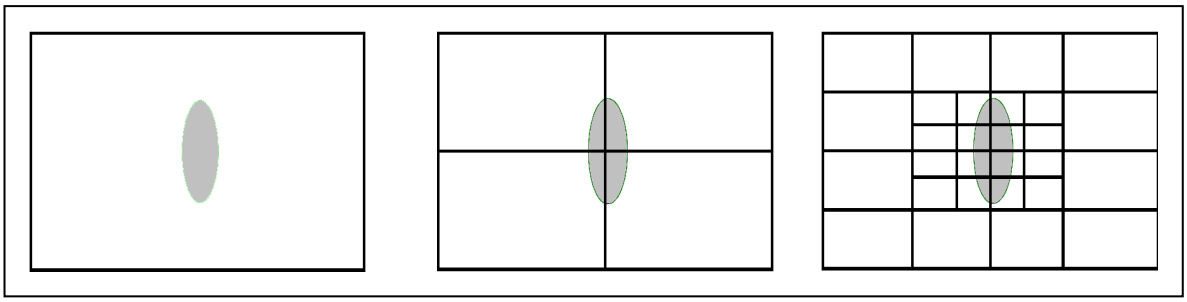
Erilaisia laskentaverkon generointimenetelmiä ja niiden variaatioita on kehitetty lukuisia (Du ja Hwang, 1992). Osa menetelmistä soveltuu esimerkiksi pelkästään rakenteettomien kolmio- tai nelitahokasverkkojen generoimiseen (Owen, 1998). Thompsonin mukaan lukuisia artikkeleja on esitetty, joissa puhutaan automaattisesta menetelmästä, mutta laskentaverkkojen generointimenetelmät eivät ole riittävän automaattisia kaikissa tilanteissa (Thompson & al., 1999).

Tässä tutkimuksessa keskitytään yleisimpiin rakenteettomien laskentaverkkojen generointimenetelmiin. Yleisimmin käytetty elementtityyppi rakenteettoman laskentaverkon generoimisessa on kolmio tai nelitahokas (Owen, 1998). Rakenteettoman laskentaverkon generoinnissa voidaan yleisesti ottaen kuitenkin käyttää mitä elementtityyppiä tahansa (Thompson & al., 1999). Tällaiset menetelmät voidaan jakaa pääpiirteittäin kolmeen kategoriaan; *Octree*, *Delaunay* ja *Advancing Front*. Vaikkakin selviä eroavaisuuksia on kaksi- ja kolmiulotteisten laskentaverkkojen kompleksisuudessa, alla kuvatut menetelmät soveltuvat pääperiaatteiltaan niin kolmio- kuin nelitahokasverkkojen generointiin (Owen, 1998).

3.3.1 Octree

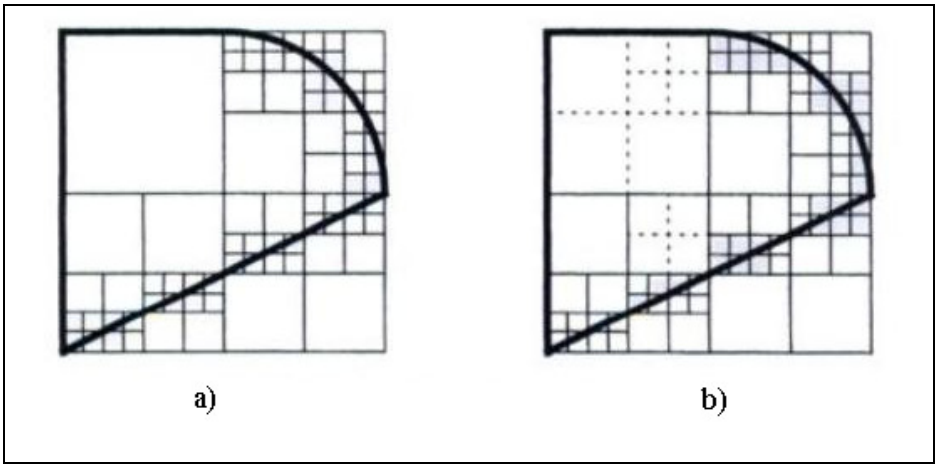
Octree-menetelmän kehitti Mark Shephard ryhmineen 1980-luvulla (Owen, 1998). Menetelmä perustuu verkotettavan laskenta-alueen peittämiseen säännöllisellä muodolla. Kaksiulotteisen laskenta-alueen peittämisessä käytetään suorakulmiota ja kolmiulotteisen lasken-

ta-alueen peittämisessä käytetään kuusitahokasta (Thompson & al., 1999). Kuvan 11 mukaisesti laskenta-alue jaetaan rekursiivisesti pienempiin osiin, kunnes haluttu lopetusehto täyttyy (Owen, 1998). Rekursiivisesti jaettuja osia kutsutaan soluiksi. Rekursiivinen jakaminen tehdään ainoastaan niille soluille, jotka sisältävät verkotettavan laskenta-alueen reunaa (Lee, 1984). Lopetusehdoksi voidaan määritellä esimerkiksi rekursion tuleva syvyys. Kaksiulotteinen laskenta-alue jaetaan aina neljään samanlaiseen osaan ja kolmiulotteinen laskenta-alue kahdeksaan osaan (Toivanen, 2004).



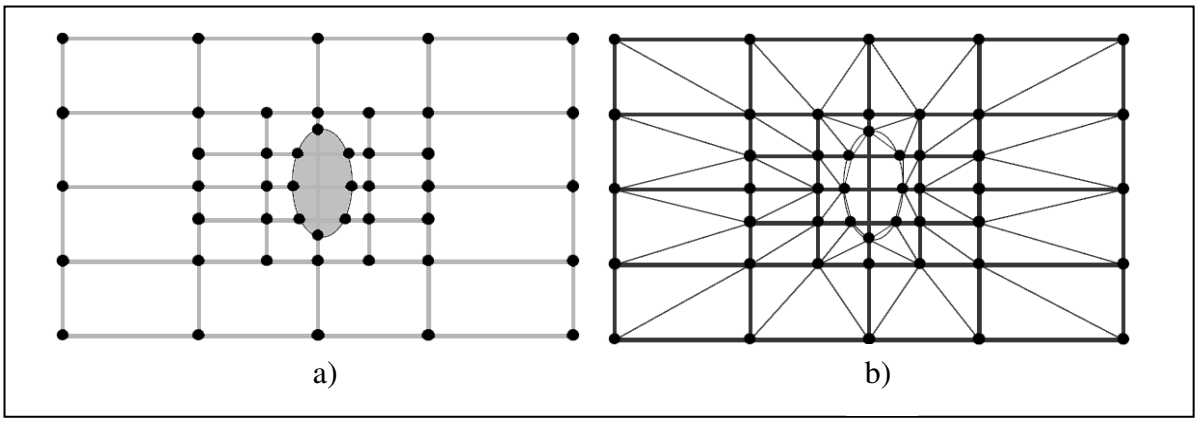
Kuva 11. Rekursiivinen kaksiulotteisen laskenta-alueen jakaminen (Wolf, 2005).

Useimmat Octree-pohjaiset menetelmät toteuttavat lopuksi niin kutsutun syvyystasojen tasapainottamisen. Prosessissa jatketaan solujen jakamista niin kauan, että vierekkäisten solujen syvyydet eroavat toisistaan korkeintaan yhdellä. Prosessin tarkoituksena on parantaa elementtien koon ja muodon kontrolloitavuutta. Kuvan 12 kohta a sisältää laskenta-alueen, jossa vierekkäisien solujen syvyysero on suurempi kuin yksi. Kuvan 12 kohta b sisältää laskenta-alueen, jossa on katkoviivoilla visualisoitu toteutettu syvyystasojen tasapainotus (Thompson & al., 1999).



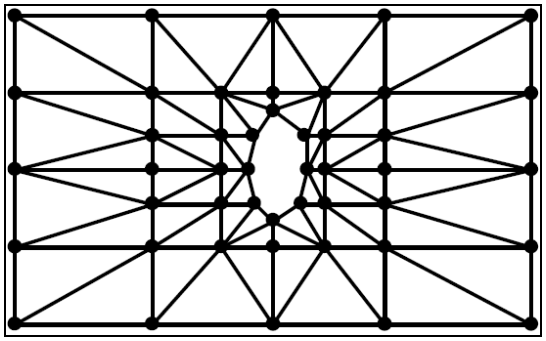
Kuva 12. Esimerkki, joka havainnollistaa laskentaverkon syvyystasojen tasapainotusta. Kuvassa laskenta-alue on rajattuna tummalla viivalla (Thompson & al., 1999).

Kun rekursiivinen jako on valmis, verkotettavan laskenta-alueen ulkopuolelle jäävät solut poistetaan ja jäljelle jäävät solut kolmioidaan. Kolmioiden solmupisteiksi tulevat jäljelle jääneiden solujen nurkkapisteet (kuva 13 kohta a). Soluihin generoitavien kolmioiden muotoon vaikuttaa montako solmua solun reunoilla on ja kuinka ne sijoittuvat toisiinsa nähden (kuva 13 kohta b). Reunoilla olevat solut vaativat erityiskäsittelyä, jotta reunojen lähistölle ei syntyisi huonolaatuisia elementtejä (Toivanen, 2004). Elementtien generointi laskenta-alueeseen voidaan myös tehdä käyttäen hieman muunneltua versiota Delaunay -menetelmästä, joka on esitelty kohdassa 3.3.2.1 (Thompson & al., 1999).



Kuva 13. Kohdassa a on laskenta-alueen solmupisteiden muodostaminen ja kohdassa b on laskenta-alueen solmupisteiden kolmiointi (Wolf, 2005).

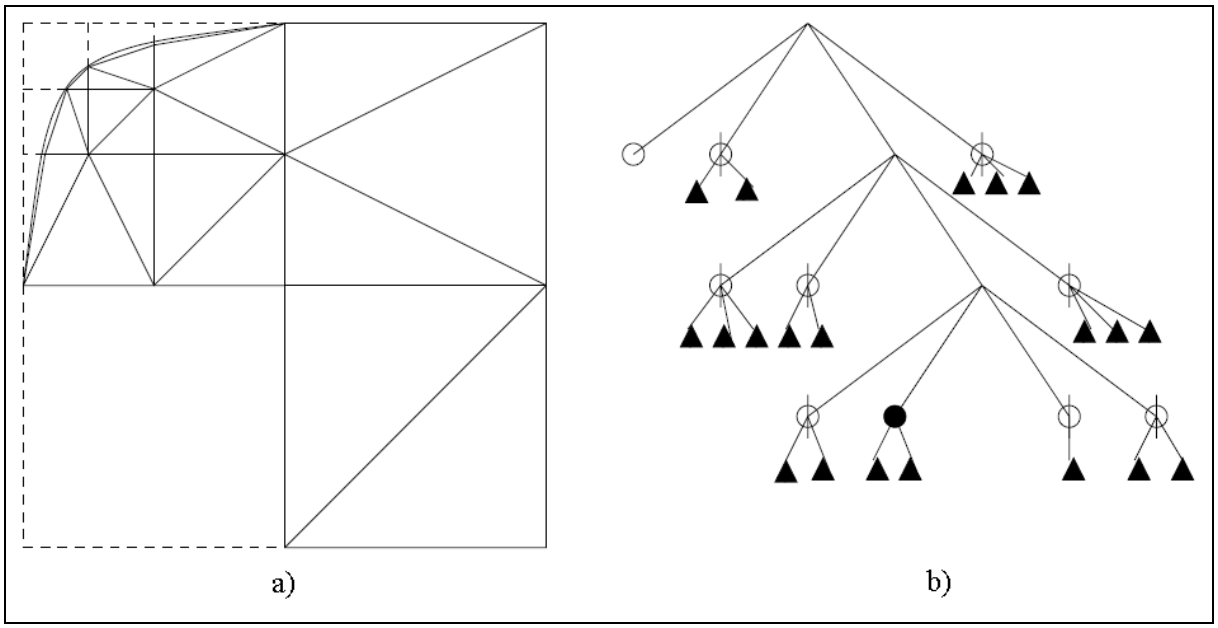
Menetelmä voi tuottaa myös hyvin erikokoisia elementtejä, mutta näiden laadullisia kriteerejä voidaan lopuksi parantaa käyttäen erilaisia tekniikoita kuten *siloittaminen* (smoothing) tai *uudelleen sijoittamalla solmupisteitä* (node point repositioning) (Thompson & al., 1999). Kuva 14 esittää valmista Octree-menetelmällä generoitua laskentaverkkoa.



Kuva 14. Octree-menetelmällä generoitu valmis laskentaverkko (Wolf, 2005).

Octree-menetelmän huonona puolena voidaan pitää sitä, että laskentaverkon lopputulos riippuu vahvasti siitä, missä asennossa kappale koordinaatistossa esitetään. Octree-menetelmällä toteutettu jako voidaan helposti esittää puurakenteena. Puun juurena on laskenta-alueen peittävä suorakulmio tai kuusitahokas. Puun juuren lapsia ovat jakamisessa syntyneet solut. Kaksiulotteisen laskenta-alueen jakamisessa lapsia on aina neljä ja kolmi-

ulotteisen laskenta-alueen jakamisessa kahdeksan. Kuvan 15 kohdassa a on kaksiulotteinen laskenta-alue, johon on generoitu laskentaverkko Octree-menetelmällä ja kohdassa b vastaava verkko esitettynä puurakenteena. Puurakennetta voidaan hyödyntää laskentaan liittyvissä operaatioissa, joissa esimerkiksi tarvitaan tietoa tietyn osa-alueen soluista tai tietyn solun naapureista (Toivanen, 2004).



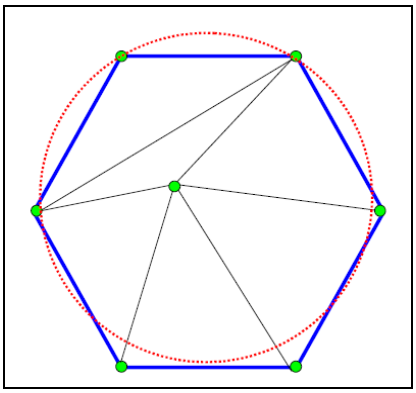
Kuva 15. Kohdassa a on Octree-menetelmällä generoitu laskentaverkko ja kohdassa b sama laskentaverkko on esitettynä puurakenteena (Flaherty ja Eaton, 2000).

3.3.2 Delaunay

Suosituimpia kolmio- ja nelitahokaslaskentaverkkojen generointimenetelmiä ovat ne, jotka toteuttavat *Delaunayn ehdon* (delaunay criterion) (Owen, 1998). Tässä kappaleessa esimerkkinä käytetään kaksiulotteista solmupistejoukkoa, johon kolmioelementeistä koostuva laskentaverkko halutaan generoida.

3.3.2.1 Toimintaperiaate

Delaunayn ehto määrittää, että jokainen elementti ympyröidään ympyrällä, jonka kehä kulkee tämän elementin kaikkien solmupisteiden kautta. Laskentaverkko toteuttaa Delaunayn ehdon, mikäli yksikään elementtejä ympäröivistä ympyröistä ei sisällä pelkästään toiselle elementille kuuluvaa solmupistettä (Owen, 1998). Tämä menetelmä mahdollistaa korkealaatuisten laskentaverkkojen generoinnin suhteellisen nopeassa ajassa (Ito & al., 2004). Kuvassa 16 on esimerkkinä generoitu laskentaverkko, joka ei täytä Delaunayn ehtoa.



Kuva 16. Laskentaverkon melkein keskikohdassa oleva solmupiste rikkoo Delaunay ehtoa (Wolf, 2005).

Delaunayn ehto itsessään ei ole laskentaverkon generointialgoritmi, vaan pelkästään kriteeristö, kuinka yhdistetään olemassa olevat solmupisteet. Siksi tarvitaan metodi solmupisteiden generoimiseksi laskenta-alueeseen. Tyypillinen lähestymistapa on luoda laskenta-alueen reunoille ensimmäiset solmupisteet. Nämä solmupisteet kolmioidaan Delaunayn ehdon mukaisesti. Seuraavaksi solmupisteitä lisätään inkrementaalisti laskenta-alueeseen siten, että jokainen uusi solmupiste säilyttää Delaunayn ehdon. Se, millä periaatteella uusia solmupisteitä lisätään laskenta-alueeseen, erottaa Delaunay algoritmit toisistaan.

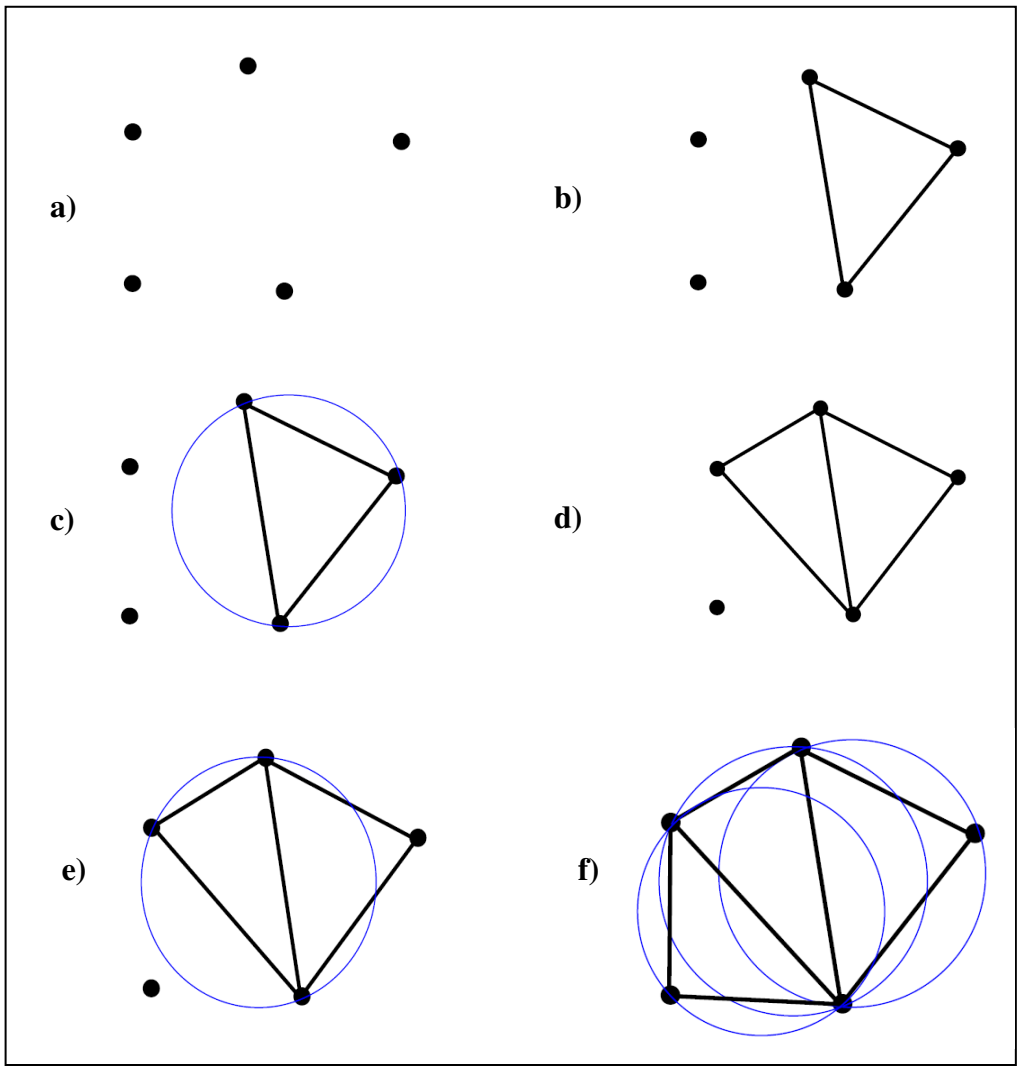
Kuva 17 havainnollistaa laskentaverkon generointia Delaunay -menetelmällä. Kuvan 17 kohdassa a on laskenta-alueena mielivaltainen solmujoukko. Kuvan 17 kohdan b mukaises-

ti laskentaverkon generointi aloitetaan valitsemalla kolme solmupistettä, joista luodaan ensimmäinen kolmio- elementti. Seuraavaksi tarkistetaan, että kolmio täyttää Delaunayn ehdon kuvan 17 c mukaisesti ympyröimällä kolmio. Mikäli ympyrän sisälle ei jää kolmiolle kuulumattomia solmupisteitä, kolmio täyttää Delaunayn ehdon ja seuraava kolmio luodaan kuvan 17 kohdan d mukaisesti. Kuvan 17 kohdan e mukaisesti tarkistetaan, että luotu elementti toteuttaa Delaunayn ehdon ja samalla periaatteella jatketaan niin kauan, kunnes kaikki joukon solmupisteet on kolmioitu. Kuvan 17 kohdasta f nähdään, että kaikki luodut kolmiot täyttävät Delaunayn ehdon ja laskentaverkko on valmis.

3.3.2.2 Solmupisteen lisääminen

Yksinkertaisin solmupisteiden lisäysmenetelmä on määrittää säännöllinen pisteverkko, joka peittää laskenta-alueen tietyllä pistetiheydellä. Jotta menetelmä pystyisi tuottamaan vaihtelevan kokoisia elementtejä, voidaan määritellä erillinen elementtikokofunktio, jonka avulla laskenta-alueeseen lisätään pisteitä niin kauan, kunnes funktion asettamat ehdot elementtien koolle täyttyvät. Toinen lähestymistapa on rekursiivisesti lisätä solmupisteitä nykyisten kolmioiden keskipisteisiin. Weatherill esittää algoritmin solmupisteiden lisäämiseen kolmioiden keskipisteeseen siten, että määriteltyä elementtikoko funktion sääntöä ei rikota (Weatherill ja Hassan, 1994).

Tunnetumpi solmupisteiden lisäysmenetelmä on *etenevän rintaman* –menetelmä (advancing front approach). Solmupisteet lisätään inkrementaalisesti, mutta laskenta-alueen ulkoreunalta lähtien tasaisesti kohti keskustaa. Menetelmässä jokainen kolmio käydään läpi tutkimalla, mihin kohtaan neljäs solmupiste olisi optimaalisinta sijoittaa laskenta-alueen sisällä. Kohdan löydyttyä solmupiste lisätään ja lokaalikolmiointi suoritetaan. Tämä menetelmä tuottaa yleensä laskentaverkkoja, jotka näyttävät rakenteellisille ja sisältävät hyvälaatuisia elementtejä laskenta-alueen reunoilla (Owen, 1998).



Kuva 17. a) Solmujoukko. b) Luodaan ensimmäinen kolmio elementti. c) Tarkistetaan Delaunayn ehto. d) Luodaan uusi kolmio elementti. e) Tarkistetaan Delaunayn ehto. f) Valmis Delaunayn ehdon toteuttava laskentaverkko (Wolf, 2005).

Chew (1989) ja Ruppert (1992) esittävät vaihtoehtoisen menetelmän uusien solmupisteiden lisäämiseen. Menetelmässä uusi solmupiste lisätään elementtiä ympyröivän ympyrän keskipisteeseen. Kun solmupisteet lisätään tarkoin määritetyssä järjestyksessä tekniikka tunnetaan nimellä *taattu laatu* (guaranteed quality).

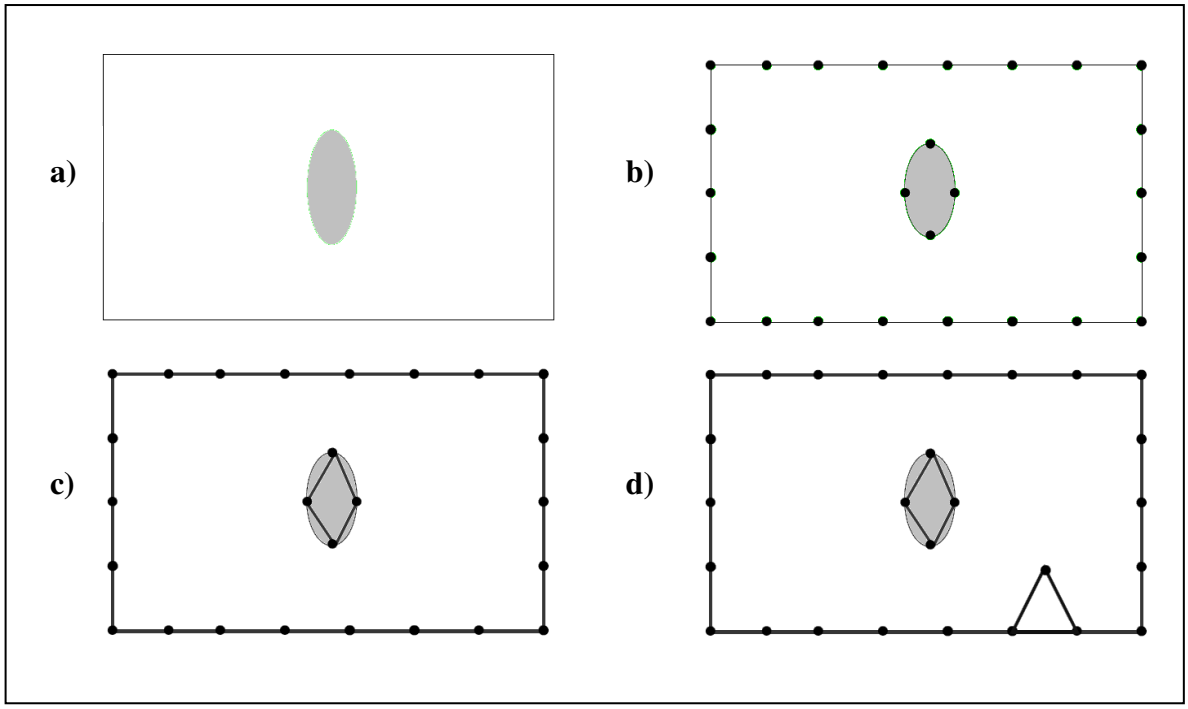
Samantyyppinen ympyröivän ympyrän solmupisteen lisäysmenetelmä on nimeltään *Voroinoi-segmentti solmupisteenlisäysmenetelmä* (Voronoi-segment point insertion method). Voronoi-segmentti voidaan määritellä viivasegmenttinä, joka koostuu kahden vierekkäistä kolmiota ympyröivien ympyröiden keskipisteistä. Uusi solmupiste lisätään Voronoi-segmentin sisään siten, että säilytetään optimaalinen elementin kokokriteeri. Tämä menetelmä tuottaa rakenteellisen näköisiä laskentaverkkoja, jossa jokaiseen sisäiseen solmupisteeseen liittyy kuusi kolmiota (Owen, 1998).

3.3.3 Advancing Front

Toinen hyvin suosittu menetelmä laskentaverkkojen generoimisessa on *etenevän rintaman*–menetelmä (Advancing Front). Menetelmästä on olemassa useita variaatioita. Menetelmän pääkehittäjät ovat Rainald Lohner ja S. H. Lo (Owen, 1998). Etenevän rintaman–menetelmän tuloksena syntyy rakenteettomia laskentaverkkoja. Menetelmällä voidaan generoida laskentaverkkoja, jotka sisältävät kolmio-, nelitahokas-, kuusitahokas- tai nelikulmioelementtejä. Aluksi menetelmän avulla pystyttiin generoimaan vain kaksiulotteisia laskentaverkkoja, mutta myöhemmin menetelmää laajennettiin kattamaan myös kolmiulotteisen laskenta-alueen verkottaminen. Lisäksi laskentaverkon generoinnin aikavaativuutta ja solmupisteen lisäysalgoritmia on optimoitu (Thompson & al., 1999).

Menetelmässä verkotettavaan laskenta-alueeseen rakennetaan elementtejä askeleittain edeten progressiivisesti ulkoreunasta sisäänpäin. Kuvan 18 kohta a sisältää laskenta-alueen, johon generoidaan laskentaverkko etenevän rintaman-menetelmällä. Ensimmäisenä luodaan verkotettavan laskenta-alueen reunoille solmupisteitä kuvan 18 kohdan b mukaisesti. Reunasolmupisteiden etäisyydet tulee olla yhdenmukaiset ja oikeassa suhteessa verkotettavan laskenta-alueen kokoon. Luodut solmupisteet yhdistetään viivasegmenteiksi (kuva 18, kohta c). Viivasegmentit ovat menetelmän aloitusrintama, josta elementtien generointi aloite-

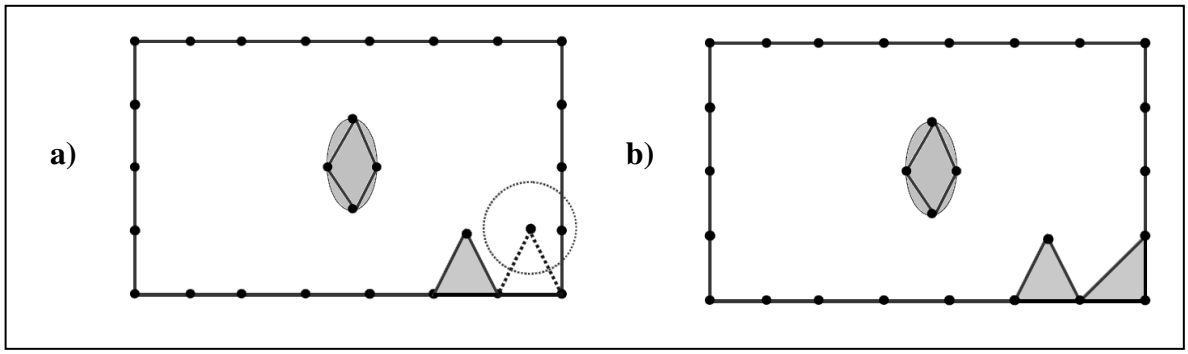
taan. Uusi kolmio luodaan lisäämällä uusi solmupiste laskenta-alueeseen tai käyttäen olemassa olevaa solmupistettä kuvan 18 kohdan d mukaisesti.



Kuva 18. a) Laskenta-alue johon laskentaverkko generoidaan. b) Luodaan laskenta-alueen reunoilla solmupisteet. c) Yhdistetään solmupisteet viiva-segmenteiksi. d) Luodaan ensimmäinen elementti laskenta-alueeseen (Wolf, 2005).

Kuvan 19 kohdassa a on tilanne, jossa uuden solmupisteen sijoituskohdan läheisyydestä löytyy myös toinen solmupiste. Menetelmässä valitaan kolmion solmupisteeksi se, joka tuottaa laadullisesti parhaimman kolmioelementin (kuva 19, kohta b). Aina, kun uusi elementti lisätään laskenta-verkkoon, tulee varmistua siitä, ettei se ole päällekkäin minkään muun elementin kanssa. Tarvittaessa voidaan määritellä koko funktio kontrolloimaan elementtien kokoa saman periaatteen mukaisesti kuin Delaunay –menetelmässäkin (Owen, 1998).

Aina kun uusi elementti luodaan laskentaverkkoon, rintama päivitetään. Rintama on dynaaminen tietorakenne, joka alkutilanteessa sisältää tiedon viivasegmenteistä. Kun uusi elementti luodaan laskentaverkkoon, poistetaan kolmion generoimisessa käytetty viiva segmentti tietorakenteesta. Mikäli kolmion muodostamisessa luotiin uusi solmupiste, lisätään tämä solmupiste rintaman tietorakenteeseen (Thompson & al., 1999).



Kuva 19. Kohdassa a laskenta-alue sisältää usean solmupiste vaihtoehdon kolmiolle. Kohdassa b valitaan laadultaan paras solmupiste kolmiolle (Wolf, 2005).

Laskentaverkon generointi on valmis, kun rintamaa kuvaava tietorakenne on tyhjä (Thompson & al., 1999). Ito:n mukaan menetelmässä on kaksi haittapuolta. Menetelmän laskennallinen nopeus on hidas johtuen lukuisista suoritettavista geometriahauista ja valmiin laskentaverkon laadukkuus voi vaihdella (Ito & al., 2004).

3.4 Laskentaverkkotiedostoformaatti

Laskentaverkkotiedostoformaatteja on käytössä lukuisia. Yhteistä näille formaateille on kuitenkin se, että ne ovat pääpiirteiltään hyvin samanlaisia. Tässä kappaleessa esitellään kaksi tunnettua laskentaverkkotiedostoformaattia.

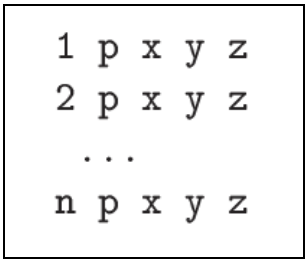
3.4.1 ELMER laskentaverkkotiedostoformaatti

ELMER-laskentaverkko koostuu neljästä erillisestä tiedostosta; mesh.header, mesh.nodes, mesh.elements ja mesh.boundary. Ensimmäinen mesh.header tiedoston rivi sisältää laskentaverkon solmupisteiden, elementtien ja reunaelementtien lukumäärän. Laskentaverkossa käytettyjen erilaisten elementtityyppien lukumäärä on ilmoitettu toisella rivillä. Tämän jälkeen ilmoitetaan riveittäin yksitellen elementtityypit ja niiden lukumäärät. ELMER-laskentaverkko tiedostoformaatissa erilaiset elementtityypit ilmaistaan numerokoodeina. Kuva 20 havainnollistaa mesh.header tiedoston sisältöä. Laskentaverkko sisältää 300 solmupistettä, 261 elementtiä ja 76 reunaelementtiä. Laskentaverkossa käytetään kahta erilaista elementtityyppiä, joiden tyyppikoodit ovat 404 ja 202 (CSC, 2000b).

```
300 261 76
2
404 261
202 76
```

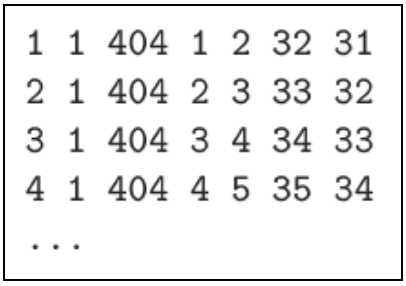
Kuva 20. Tiedoston mesh.header kuvaus (CSC, 2000b).

Tiedosto mesh.nodes sisältää laskentaverkon solmupisteet (kuva 21). Tiedoston jokainen rivi sisältää viisi lukua, joista kaksi ensimmäistä ovat kokonaislukuja ja kolme viimeistä reaalilukuja. Ensimmäinen luku ilmoittaa solmupisteen järjestysnumeron. Solmupisteiden ei välttämättä tarvitse olla nousevassa järjestyksessä. Rivin toinen luku on laskentaverkon ositusindeksi, jota käytetään rinnakkaislaskennassa. Ositus indeksin arvoksi voidaan asettaa -1 tai 1, jos sitä ei tarvita. Lopuksi rivi sisältää solmupisteen x, y, ja z koordinaatit. Huomioitavaa on, että kaikki kolme koordinaattia tulee antaa vaikka laskentaverkko olisi yksi- tai kaksiulotteinen (CSC, 2000b).



Kuva 21. Tiedoston mesh.nodes kuvaus (CSC, 2000b).

Elementit on kuvattu mesh.elements tiedostossa (kuva 22). Tiedoston jokainen rivi noudattaa samaa rakennetta. Ensimmäinen luku määrää elementin järjestysnumeron ja toinen numero määrää mihin osa-alueeseen elementti kuuluu. Kolmantena on elementtityypin koodi, jota seuraa solmupisteindeksit, joiden perusteella elementin geometria muodostuu (CSC, 2000b).



Kuva 22. Tiedoston mesh.elements kuvaus (CSC, 2000b).

Reunaelementtien tiedosto mesh.boundary on rakenteeltaan hyvin samankaltainen kuin mesh.elements tiedosto. Ensimmäisenä on reunaelementin järjestysnumero ja toisena luku, joka määrää, mihin reuna-alueeseen elementti kuuluu. Kaksi seuraavaa lukua viittaavat mesh.elements tiedoston elementtien järjestyslukuihin. Viitatut elementit ovat tämän reunaelementin vanhempia. Mikäli reunaelementti kuuluu laskenta-verkon ulkoreunaan, sillä on tällöin vain yksi vanhempi ja toinen vanhempi on arvoltaan nolla, kuten kuvassa 23 jokaisen rivin neljäs numero on asetettu nollaksi. Lopuksi ovat solmupisteindeksit, joiden perusteella elementin geometria muodostuu (CSC, 2000b).

```
2 1 735 0 404 175 1686 1685 174
3 1 791 0 404 176 1687 1686 175
4 1 847 0 404 177 1688 1687 176
5 1 903 0 404 178 1689 1688 177
...
```

Kuva 23. Tiedoston mesh.boundary kuvaus (CSC, 2000b).

3.4.2 UWVF laskentaverkkoformaatti

Waveller Preprocessor -ohjelmassa käytetty UWVF (Ultra Weak Variational Formulation) laskentaverkon tietorakenne noudattelee pääpiirteittäin samaa rakennetta kuin mitä käytetään yleisestikin FEM-ohjelmissa. Tyypilliset FEM-laskentaverkkotiedostot sisältävät listan kärkipiste- (vertex) koordinaatteja ja listan elementtejä. Jokainen elementti koostuu listasta kärkipisteindeksejä, jotka muodostavat elementin, esimerkiksi nelitahokkaan (Kuava, 2006). Koska ultraheikkoon variaatioformalismiin perustuvan menetelmän keskeisenä osana ovat laskentaverkon reunojen yli laskettavat integraalit, käyttää UWVF laskentaverkkotallennusformaatti peruselementtityyppinä kolmioita (Huttunen, 2004).

UWVF laskentaverkkotallennusformaatti sisältää myös listan kärkipisteiden koordinaatteja, mutta elementtejä ei muodosteta suoraan kärkipisteiden avulla, vaan erillinen listaus elementtien tahkoja (faces) on tallennettuna mukana. Esimerkiksi nelitahokas muodostetaan neljästä erillisestä kolmiotahkosta. Matriisi (matrix), joka sisältää elementin tiedot, sisältää myös indeksit tämän elementin vierekkäisiin elementteihin ja lopuksi indeksin, joka määrittelee *osa-alueen* (subdomain) elementille. Jokaisen tahkon listaus sisältää myöskin *raja-* (boundary) tyypin indeksin, jota tarvitaan erottamaan erilaiset *rajatilat* (condition) ulkoisessa tilavuudessa tai valinnaisesti osoittamaan laskentaverkon sisäiset sidokset.

Kuva 24 havainnollistaa laskentaverkkotiedoston rakennetta. Ensimmäinen laskentaverkkotiedoston rivi sisältää kolme lukua; kärkipisteiden, tahkojen ja elementtien lukumäärän.

```
   7575 77887 37715
     5     6     9    956     0     0     2     6     1
     7     8     9  50641     3     4     1     7     1
     1     2     7     44     0     0     2     9     1
     3     4     8    177     0     0     2     0     1
   ...
   ...
   ...
    1       4   3559       3
    1       4   3571       1
    1      13   3559       3

     ...
     ...
     ...
-5.000000000000e-01 -0.000000000000e+00  5.000000000000e-01
-5.000000000000e-01  1.900000000000e+00  5.000000000000e-01
-5.000000000000e-01  2.000000000000e+00  5.000000000000e-01
-5.000000000000e-01  0.000000000000e+00  4.000000000000e-01
    ...
    ...
    ...
```

Kuva 24. Esimerkki UWVF laskentaverkkotiedoston sisällöstä (Kuava, 2006).

Ensimmäisen rivin jälkeen alkavat elementtien listaukset. Sarakkeet 1-4 sisältävät elementin tahkojen indeksit ja nämä luvut tulee olla nousevassa järjestyksessä riveittäin. Sarakkeet 5-8 sisältävät elementin jokaisen tahkon viereisen elementin indeksin. Mikäli elementin jollain tahkolla ei ole viereistä elementtiä eli elementti on laskentaverkon ulkoreunaa, on elementin indeksi nolla. Viimeinen yhdeksäs sarake määrittää osa-alueindeksin, jota tarvitaan määritettäessä materiaaliparametrit elementille. Osa-alueindeksit tulee olla kasvavassa järjestyksessä alkaen yhdestä.

Elementtilistausten jälkeen tiedostoformaatti sisältää tahkojen listaukset. Sarakkeet 1-3 sisältävät tahkon koordinaattien indeksit ja neljäs sarake sisältää tahkon tyypin. Sarakkeiden 1-3 luvut tulee olla nousevassa järjestyksessä. Sisempien elementtien tahkojen indeksi on nolla ja laskentaverkon ulkoreunan tahkojen indeksi tulee olla aina eri suuri kuin nolla.

Tahkojen tyyppi-indeksit numeroidaan numerosta yksi ylöspäin. Esimerkiksi jos laskentaverkko sisältää viisi erilaista tahkotyyppiä, tulee indeksien olla yksi, kaksi, kolme, neljä ja viisi.

Viimeisenä UWVF laskentaverkkoformaatti sisältää kärkipisteiden koordinaatit. Jokainen kärkipiste sisältää x, y ja z arvot. Laskentaverkkotiedostojen koordinaatit ovat aina *tieteellisellä merkintätavalla* (scientific notation) ilmoitettuna. Yleisesti ottaen tieteellisellä merkintätavalla luku 1230000 voidaan kirjoittaa muodossa $1{,}23 \cdot 10^6$. Tieteellistä merkintätapaa käytetään suurten ja pienten lukujen kirjoittamisessa (Kuava**,** 2006).

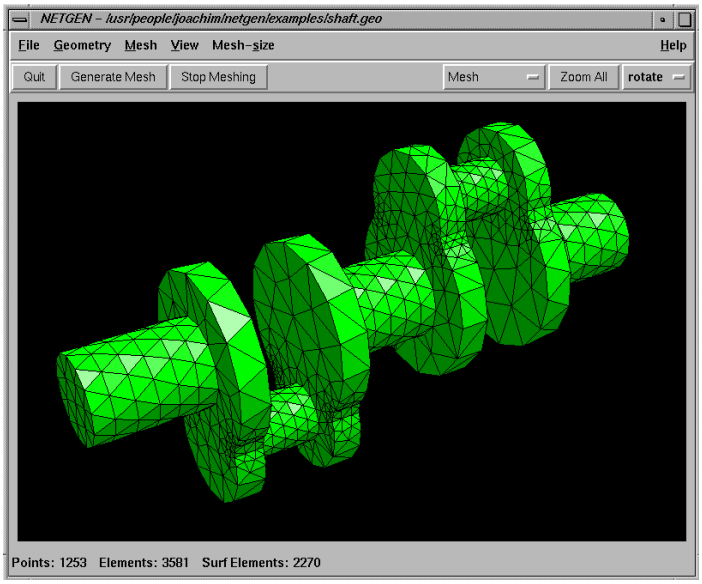
3.5 Laskentaverkon generointiohjelmia

Laskentaverkon generointiohjelmia on julkaistu lukuisia, joista suurin osa käyttää elementtityyppinään kolmiota. Toiseksi eniten käytetty elementtityyppi laskentaverkkogeneraattoreissa on nelitahokas. Suurin osa näistä laskentaverkkogeneraattoreista on kaupallisia (Owen, 1998). Seuraavissa kappaleissa tutustutaan NETGEN -laskentaverkkogeneraattoriin, joka pohjautuu avoimeen lähdekoodiin ja Comsol Multiphysics-ohjelmiston mukana tulevaan kaupalliseen laskentaverkkogeneraattoriin.

3.5.1 NETGEN

NETGEN on automaattinen kolmiulotteinen nelitahokaslaskentaverkkogeneraattori, jonka Joachim Schöberl kehitti kumppaneineen Johannes Kepler-yliopistossa (NETGEN, 2007). Ohjelma hyväksyy syötteenä CSG (constructive solid geometry) tiedostoformaatin tai BRep (boundary representation) rakenteen STL tiedostoformaatissa. Ohjelmalla voidaan laskentaverkko visualisoida monipuolisesti esittäen se esimerkiksi rautalankana tai päällystettynä materiaalilla (kuva 25). Lisäksi laskentaverkkoa voidaan leikata, jotta voidaan tutkia generoituja elementtejä laskentaverkon sisäosissa.

NETGEN käyttää Delaunay-pohjaista menetelmää laskentaverkon generoimiseksi. Arviolta noin 98 % laskentaverkon elementeistä luodaan käyttäen tätä menetelmää ja jäljelle jäävien elementtien generointiin sovelletaan *takaisinpäin etenevää algoritmia* (back-tracking rule-based method). NETGEN sisältää moduulit laskentaverkon optimointiin ja hierarkkiseen laskentaverkon jalostamiseen. Generoitu laskentaverkko voidaan tallentaa lukuisissa eri tiedostomuodoissa, kuten Abagus, Fluent, STL, VRML, Surface Mesh Format ja FEPP. NETGEN pohjautuu avoimeen lähdekoodiin LGPL lisenssin alla. Ohjelma on saatavilla Unix, Linux ja Windows käyttöjärjestelmille (Schöberl, 2004).



Kuva 25. NETGEN-ohjelmistolla generoitu laskentaverkko (NETGEN, 2007).

3.5.2 Comsol Multiphysics laskentaverkkogeneraattori

Comsol Multiphysics ohjelmisto sisältää laskentaverkon generointityökalun, jonka avulla voidaan generoida yksi-, kaksi- tai kolmiulotteisia laskentaverkkoja (Comsol, 2006). Ohjelma tuottaa rakenteettomia laskentaverkkoja. Kaksiulotteisen laskentaverkon generoimisessa voidaan käyttää elementtityyppinä joko kolmiota tai nelikulmiota. Kolmiulotteisen laskentaverkon elementtityyppinä voidaan käyttää nelitahokasta, kuusitahokasta tai pris-

maa. Laskentaverkkogeneraattorin avulla voidaan muun muassa kontrolloida viereisten elementtien kokoeroja, laskentaverkon kaarevuutta ja elementtien kokoa verrattuna laskenta-alueeseen. Comsol Multiphysics laskentaverkkogeneraattori sisältää myös lukuisia menetelmiä generoidun laskentaverkon jälkikäsittelyyn, joiden avulla voidaan muokata laskentaverkkoon generoituja elementtejä.

4. Java3D ja laskentaverkkojen visualisointi

Tässä luvussa perehdytään Java3D-ohjelmointirajapintaan ja esitetään kuinka rajapinnan avulla luodaan kolmiulotteisia objekteja. Tämän jälkeen perehdytään laskentaverkkojen visualisointiin Java3D-ohjelmointirajapinnan avulla. Lopuksi esitetään malliesimerkki, jossa ratkaistaan ääniaallon sironta kovasta kappaleesta.

4.1 Johdatus Java 3D-ohjelmointirajapintaan

Sun Microsystemsin kehittämä *Java 3D API* on ohjelmointirajapinta sellaisten ohjelmien toteuttamiseen, jotka esittävät tai ovat vuorovaikutuksessa kolmiulotteisen grafiikan kanssa. Ohjelmointirajapinta (Application Programming Interface, API) on käyttöliittymä, jolla eri ohjelmat voivat tehdä pyyntöjä ja vaihtaa tietoja. Java 3D on standardilaajennus Java 2 JDK kehitysympäristöä ja se tarjoaa kokoelman korkean tason rajapintoja kolmiulotteisen geometrian luomiseen, manipulointiin ja grafiikan renderöintiin eli kuvan luomiseen mallista. Java 3D -rajapinnan kehitys aloitettiin vuonna 1996, jolloin Sun julkaisi suunnitelmansa sen kehittämisestä. Java 3D 1.1 API oli rajapinnan ensimmäinen versio ja se julkaistiin vuoden 1998 lopussa. Tällä hetkellä uusin saatavilla oleva versio Java 3D -rajapinnasta on 1.5.1 (Bouvier, 2001).

Java 3D sisältää funktiot kuvankäsittelyyn, visualisointiin, animaatioiden tuottamiseen ja interaktiivisten 3D graafisten ohjelmien toteuttamiseen. Java 3D pohjautuu olemassa olevilla alemman tason 3D-rajapinnoille, esim. OpenGL, Direct3D ja QuickDraw3D. Java-ohjelmointikielen tavoin Java 3D -ohjelmat ovat ajettavissa useimmissa laiteympäristöissä.

Vaikkakin Sun Microsystems on jo pitkään tukenut interaktiivista 3D-renderöintiä, se ei ole johtava toimija 3D-grafiikan saralla. Yleisin käytetty standardi interaktiiviselle 3D renderöinnille on OpenGL, joka on Silicon Graphics:n (SGI) toteuttama rajapinta. OpenGL

suunniteltiin alustariippumattomaksi renderöintiarkkitehtuuriksi ja sitä tukee laaja kirjo käyttöjärjestelmiä, näytönohjainvalmistajia ja ohjelmistoja. OpenGL rajapinta on toteutettu C-ohjelmointikielellä ja tästä syystä sitä ei voi kutsua suoraan Java-ohjelmointikielellä. Koska Java 3D on korkean tason ohjelmointirajapinta, ei sillä välttämättä päästä yhtä korkeaan suorituskykyyn kuin mihin taitava kehittäjä pystyisi OpenGL rajapintaa käyttäessä. Mikäli mahdollisimman korkea renderöinnin suorituskyky on pääkriteerinä toteutusta suunniteltaessa, Java 3D ei todennäköisesti ole optimaalisin ratkaisu rajapinnaksi. Java 3D ohjelmat voidaan kirjoittaa itsenäisiksi ohjelmiksi tai selaimessa ajettaviksi appleteiksi. Java 3D ohjelmia ajettaessa selaimessa tulee olla tuki Java 3D-rajapinnalle (Selman, 2002).

4.1.1 Java 3D luokkahierarkia

Jokainen Java 3D-ohjelma koostuu ainakin osaksi olioista, jotka kuuluvat Java 3D luokan hierarkiaan. Tämä kokoelma olioita kuvaa virtuaalisen universumin, joka renderöidään. Java 3D API määrittelee yli sata luokkaa, jotka ovat osa *javax.media.j3d* pakettia. Nämä luokat tunnetaan yleisesti nimellä Java 3D *ydinluokat* (core classes). Ydinluokat sisältävät satoja metodeja, mutta vain muutamakin riittänee luomaan yksinkertaisen virtuaalisen universumin.

Java 3D sisältää muitakin paketteja ydinpaketin lisäksi. Yksi näistä on *com.sun.j3d.utils* -paketti, joka yleisesti tunnetaan nimellä *työkaluluokat* (utility classes). Työkaluluokat ovat kätevä ja tehokas lisä ydinpaketille. Työkaluluokat jaottuvat neljään laajempaan alakategoriaan. *Sisällön lataaja* (content loaders) -apuluokka sisältää muun muassa geometrioiden lataamisen esimerkiksi 3D-mallinnusohjelmista. *Luomisapu* -luokka (construction aids) sisältää metodeja mallin rakentamiseen. *Geometria*-apuluokka (geometry classes) sisältää valmiita geometrisiä objekteja ja peruselementtejä ja *kätevyys* (convenience utilities) apuluokka sisältää yksinkertaistettuja metodeja esimerkiksi virtuaalisen universumin luomiseen. Työkaluluokkien käyttö vähentää huomattavasti tarvittavan kirjoitetun koodin määrää. Java 3D ydin- ja apupakettien lisäksi jokainen Java 3D ohjelma käyttää *java.awt* ja *ja-*

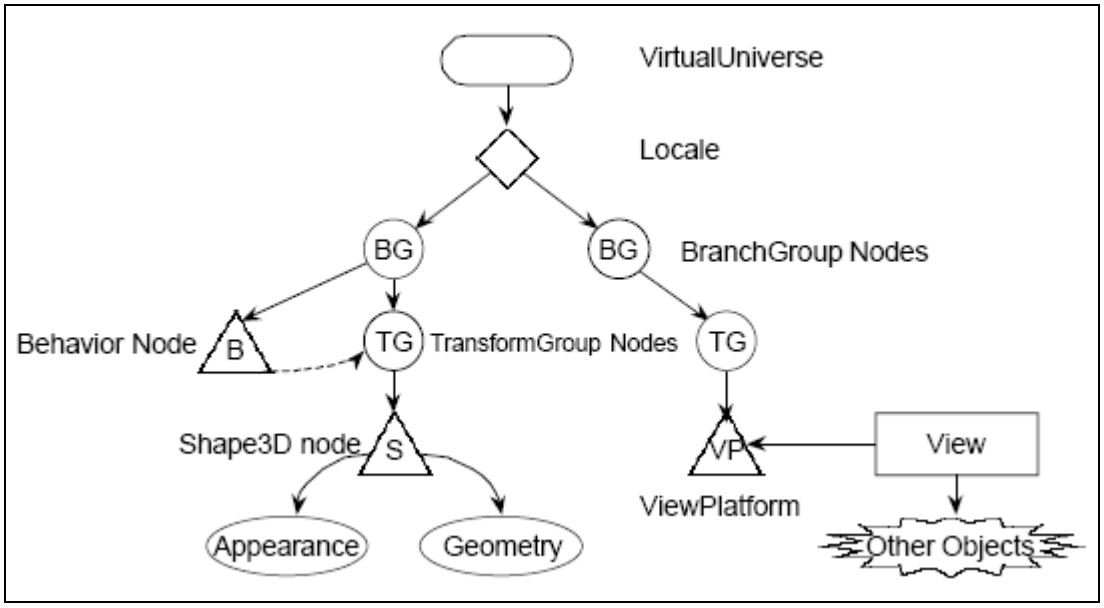
vax.vecmath -paketteja. *Java.awt* -paketti sisältää luokat renderöinnin esittämiseen. *Javax.vecmath* -paketti määrittelee luokat pisteiden, vektoreiden, matriisien ja muiden matemaattisten olioiden esittämiselle (Bouvier, 2001).

4.1.2 Maisemagraafin rakentaminen

Java 3D API on suunniteltu tukemaan joustavuutta rakennettaessa eri kokoisia virtuaalisia universumeita. Virtuaaliset universumit voivat vaihdella kooltaan astronomisesta atomitasolle. Vaikka Java 3D API sisältää huomattavan määrän toiminnallisuuksia, on sen käyttö kuitenkin suoraviivaista. Ohjelmoijan ei tarvitse tietää renderöintiin liittyviä yksityiskohtia vaan nämä hoidetaan automaattisesti. Java 3D kykenee rinnakkaiseen renderöintiin Java-säikeiden avulla. Java 3D renderöijä pystyy myös automaattisesti optimoimaan renderöinnin suorituskykyä paremmaksi

Java 3D:n virtuaalinen universumi rakentuu *maisemagraafista* (scene graph), joka luodaan käyttäen Java 3D luokista luotuja olioita. Maisemagraafi on puurakenne, joka sisältää kaiken virtuaalisen universumin renderöintiin liittyvät tiedot (Kuva 26.). Yleinen määritelmä *graafille* on tietorakenne, joka koostuu solmuista ja kaarista. Solmu on tietoelementti ja kaari on yhteys kahden tietoelementin välillä. Maisemagraafin solmut ovat Java 3D luokkien *ilmentymiä* (instances). Maisemagraafin juurisolmuna on *VirtualUniverse*-luokan olio, joka toimii *säiliönä* (container) kaikille mallien puurakenteille. Jokainen maisemagraafi voi sisältää vain yhden VirtualUniverse objektin. Maisemagraafi rakennetaan olioista, jotka määrittelevät geometrian, valon, äänen, sijainnin, suunnan, ulkonäön visuaalisille ja ääni objekteille. VirtualUniverse objekti koostuu joukosta *Locale*-objekteja, jotka toimivat säiliöinä *BranchGroup*-solmuolioille. BranchGroup-solmuoliot ovat juuria alipuulle, jossa määritellään esimerkiksi mallin graafisten objektien geometriat, ulkonäöt, sijainnit, toiminnallisuudet, valaistukset ja äänet.

Lähes kaikki luokat maisemagraafissa perivät *SceneGraphObject*-luokan, lukuunottamatta VirtualUniverse- ja Locale- luokkia. SceneGraphObject on yliluokka lähes kaikille Java 3D:n sisältämille ydin- ja työkaluluokille. SceneGraphObject-luokka sisältää kaksi aliluokkaa, jotka ovat *Node* ja *NodeComponent.* Node-luokka on *Group*- ja *Leaf*-luokkien abstrakti yläluokka. Group-luokkaa käytetään määrittämään visuaalisten objektien sijainnit ja suuntautuneisuus. Leaf-luokkaa käytetään määrittämään visuaalisten objektien muoto, ääni ja käyttäytyminen (behaviour) virtuaalisessa universumissa. Joitain Leaf-luokan aliluokkia ovat esimerkiksi *Shape3D*, *Light*, *Behaviour* ja *Sound*. NodeComponent-luokan avulla määritetään Shape3D-solmun geometria, ulkoasu, tekstuuri ja materiaali ominaisuudet (Bouvier, 2001).



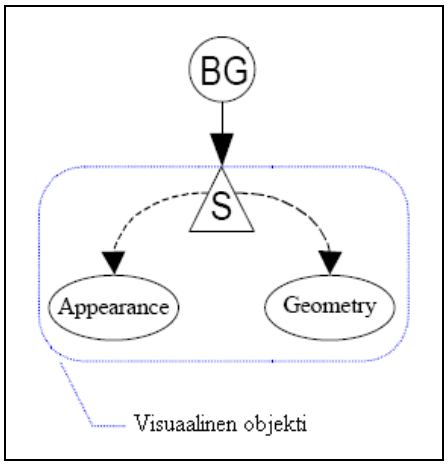
Kuva 26. Maisemagraafi esitettynä puurakenteena (Bouvier, 2001).

4.1.3 Geometrian luominen

Java 3D sisältää *com.sun.j3d.utils.geometry* –paketin, joka tarjoaa valmiita luokkia geometristen objektien luomiseen, kuten Box-, ColorCube-, Cone-, Cylinder-, Sphere- ja Text2D –luokat, joiden avulla voidaan luoda yksinkertaisia geometria objekteja helposti. Java 3D:n

c*om.sun.j3d.utils.geometry* –paketti sisältää myös c*om.sun.j3d.utils.geometry.primitives* alipaketin, joka sisältää pääpiirteiltään samat geometrialuokat kuin yläluokkansa sillä erotuksella, että tämän paketin avulla luotuja geometria objekteja voidaan modifioida monipuolisemmin. Nämä valmiit geometrian luomiseen käytettävissä olevat luokat eivät välttämättä riitä monimutkaisempien geometriaobjektien luomiseen.

Monipuolisempien geometristen objektien luomiseen Java 3D:ssä on Shape3D-luokka. Shape3D on maisemagraafin solmu, joka määrittelee visuaalisen objektin. Shape3D-olio ei itsessään sisällä informaatiota visuaalisen objektin muodosta tai väristä, vaan nämä tiedot on sisällytetty NodeComponent-solmuihin, joihin Shape3D-olio voi viitata. Shape3D-olio voi viitata yhteen geometria (Geometry) NodeComponent-solmuun ja yhteen ulkoasu (Appearance) NodeComponent-solmuun (Kuva 27).



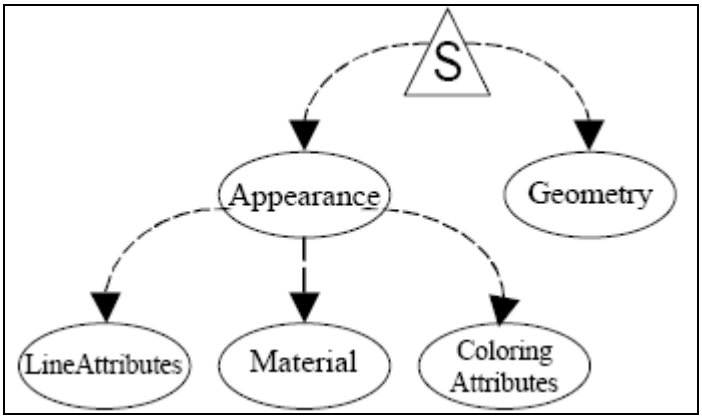
Kuva 27. Shape3D-olion sisältämät solmut (Bouvier, 2001).

Geometrialuokan perivät *Compressed Geometry, GeometryArray, Raster* ja *Text3D* -luokat. CompressedGeometry-luokkaa käytetään säilyttämään geometriainformaatio pakattuna. Raster-luokan avulla voidaan piirtää rasterikuva haluttuun kohtaan virtuaalisessa universumissa. Text3D-luokan avulla voidaan merkkijono muuntaa kolmiulotteiseksi geometriaksi. GeometryArray-luokka sisältää aliluokat *GeometryStripArray, IndexedGeometryArray, LineArray, PointArray, QuadArray* ja *TriangleArray.* Näiden luokkien avulla voidaan luoda

primitiivisiä geometrioita kuten viivoja, kolmioita, pisteitä ja nelikulmioita, joita voidaan hyödyntää monimutkaisempien geometriaobjektien luomisessa (Bouvier, 2001).

4.1.4 Geometrian ulkoasu

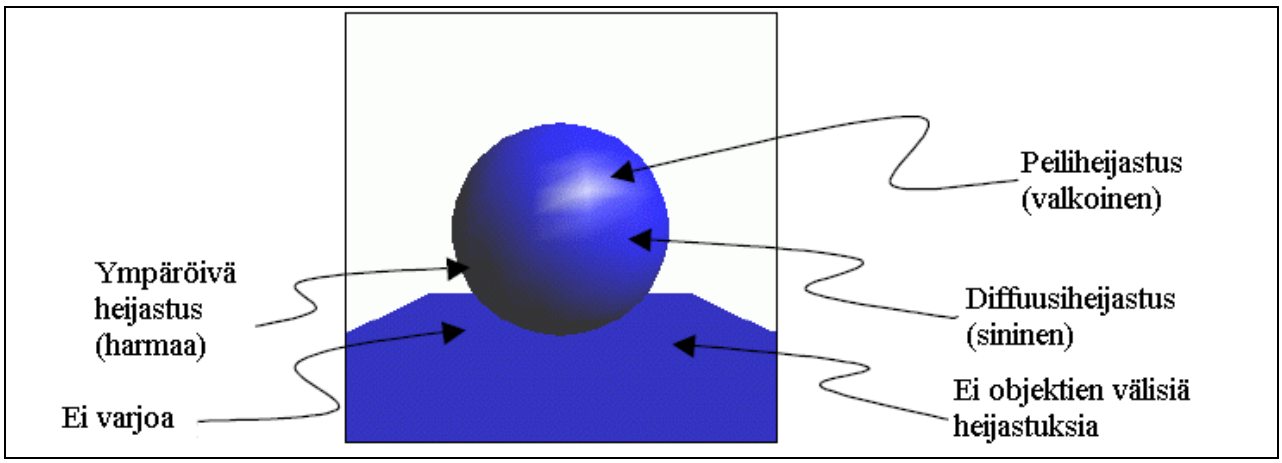
Appearance-luokka määrittää, kuinka Shape3D-olion geometriaobjekti esitetään. Appearance-olio ei itsessään sisällä informaatiota, mille Shape3D objekti tulisi näyttää, vaan se sisältää viittauksen ulkoasuun liittyvään informaatioon (kuva 28). Samaan Appearance-olioon voidaan viitata useista Shape3D-olioista.



Kuva 28. Shape3D-olion sisältämän Appearance-olion rakenne (Bouvier, 2001).

Appearance-luokka sisältää useita aliluokkia, joita kutsutaan *ulkoasuominaisuusobjekteiksi* (appearance attribute objects). Appearance-luokan sisältämät aliluokat ovat *PointAttributes, LineAttributes, PolygonAttributes, ColoringAttributes, TransparencyAttributes, RenderingAttributes, Material, TextureAttributes, Texture* ja *TexCoordGeneration.* Keskitymme kuuteen ensimmäiseen, koska ne ovat tämän tutkimuksen kannalta oleellisimmat. PointAttributes-olio sisältää tiedon, kuinka pisteprimitiivit renderöidään. Tämän olion avulla voidaan muuttaa esimerkiksi renderöitävän pisteen kokoa tai käyttää *reunanpehmennystä* (antialiasing).

LineAttributes-luokan avulla voidaan muuttaa viivan renderöintityyliä. Viivan renderöinti ominaisuuksissa voidaan vaikuttaa esimerkiksi viivan paksuuteen, muuttaa viivan piirtorakennetta, ja käyttää reunan pehmennystä. PolygonAttributes-luokan avulla määritetään miten polygoni renderöidään. ColoringAttributes-olio kontrolloi, kuinka objekti väritetään ja varjostetaan (kuva 29). Tämän luokan oliolle voidaan antaa parametreina väri- ja varjostusmalli, jolla objektin pinta varjostetaan. Väriparametri annetaan *Color3f* väri-luokan oliolla. Mahdollisia vaihtoehtoja objektin pinnan varjostusmalleiksi ovat nopein varjostus (FASTEST), laadukkain varjostus (NICEST), tasavarjostus (SHADE_FLAT) ja Gouraud-varjostus (SHADE_GOURAUD). Käytettäessä nopein tai laadukkain -varjostusmallia Java 3D määrittää laitteistosta ja Java 3D ympäristöstä riippuen varjostusmallin tyypin. TransparencyAttributes-luokan avulla voidaan geometrinen objekti asettaa *läpinäkyväksi* (transparency). Luokka sisältää erilaisia läpinäkyvyys malleja ja sen avulla voidaan kontrolloida läpinäkyvyyden voimakkuutta. Material-olion avulla voidaan määrittää mille geometrinen objekti näyttää valaistuna. Olion avulla voidaan määritellä geometrisen objektin diffuusisti heijastama väri (diffuseColor), spekulaariheijastuksen väri ja voimakkuus (specularColor ja shineness) sekä objektin läpinäkyvyyden aste (transparency) (Bouvier, 2001).



Kuva 29. Esimerkki erilaisista heijastumisista objektin pinnalla (Bouvier, 2001).

4.1.5 Valot

Virtuaalisen universumin valaisemiseen Java 3D API tarjoaa *javax.media.j3d.Light* –paketin, joka sisältää neljä erilaista valotyyppiä, joita käytetään virtuaalisen universumin valaisemiseen. Jokaiselle valotyypille voidaan määrittää muun muassa valon väri, vaikutusalue ja onko valo päällä. AmbientLight–luokan avulla voidaan luoda yleisvalo, joka valaisee kaikkialta yhtä voimakkaasti. Valon voimakkuuteen ei voi vaikuttaa. Suunnattuja valoja voidaan luoda DirectionalLight-luokan avulla. Valaisusuunta määritetään Vector3f-luokan vektorilla. PointLight-luokka mahdollistaa vaimenevaa valoa säteilevän objektin sijoittamisen virtuaaliseen universumiin, joka säteilee kaikkialle poispäin valonlähteestä. SpotLight-luokka on PointLight-luokan aliluokka, jonka avulla voidaan vaikuttaa valon valaisusuuntaan (Bouvier, 2001).

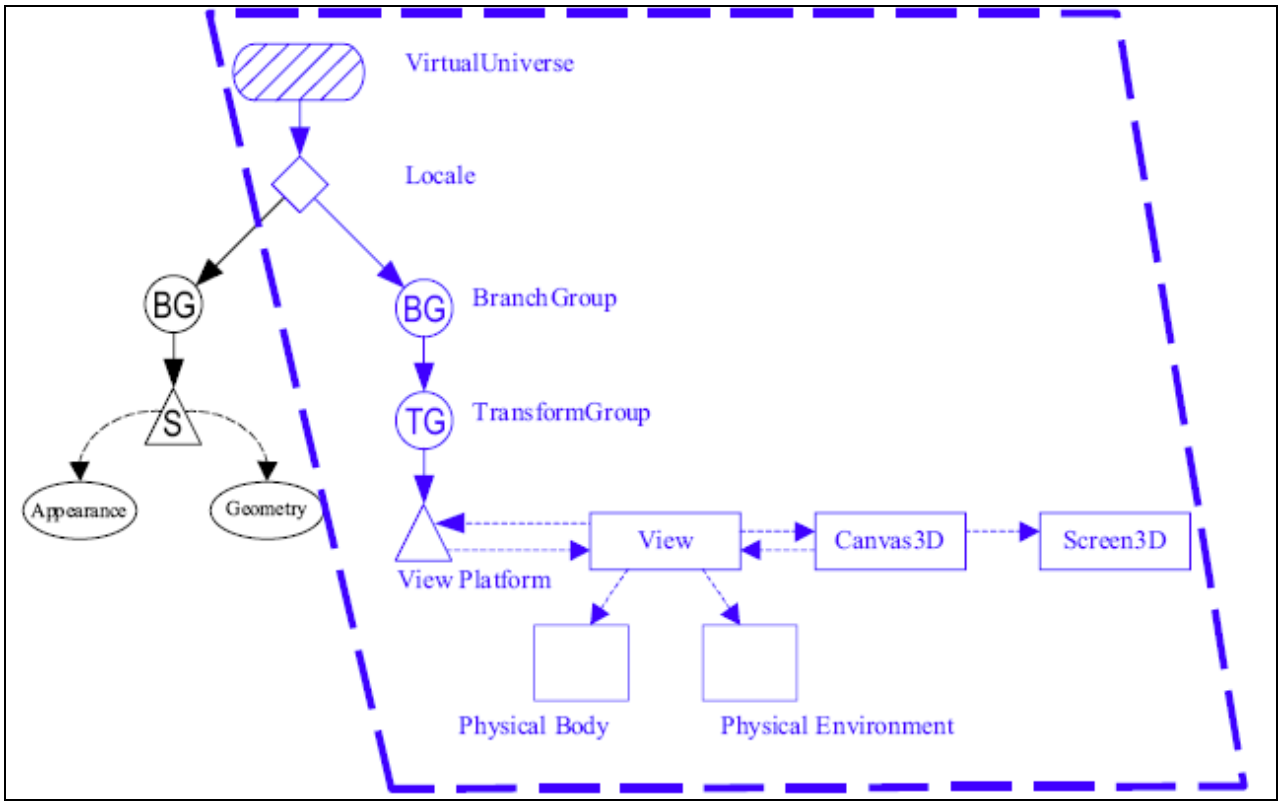
4.2 Waveller-esikäsittelijän toteutus

Esikäsittelijän (Preprocessor) tarkoituksena on havainnollistaa käyttäjälle UWVF-laskentaverkon sisältämiä tilavuuksia ja tarjota mahdollisuus asettaa laskennassa käytettäviä parametreja UWVF-laskentaverkolle. Näiden asetettujen parametrien avulla voidaan haluttu ongelma ratkaista laskennallisesti Waveller ohjelmalla.

Tärkeää on saada visualisoitua käyttäjälle selvästi eri laskentaverkon sisältämät tilavuudet, jotta parametrit voidaan asettaa oikein. Tilavuusgeometrioiden esitystapaa tulee pystyä muokkaamaan ohjelmalla. Esikäsittelijän käyttöliittymällä tulee pystyä valitsemaan laskentaverkosta yksittäisiä tilavuuksia hiirellä valiten. Lisäksi tulee toteuttaa laskentaverkon geometrian rotaatio, tarkentaminen, loitontaminen ja laskentaverkon siirtäminen virtuaalisessa universumissa. Ohjelmalla tulee voida asettaa parametreja eri tilavuuksille ja myös laskentaverkolle yleisesti. Laskentaverkon visualisointi toteutettiin Java 3D-rajapinnan avulla ja tarvittavat toiminnallisuudet Java-ohjelmointikielellä. Toteutuksessa käytetyt versiot olivat Java 3D 1.5.0 ja Java 1.5.0_06.

4.2.1 Virtuaalisen universumin luominen

Java 3D sisältää SimpleUniverse-luokan, jonka avulla on mahdollista luoda nopeasti ja helposti ominaisuuksiltaan minimalistinen ympäristö kolmiulotteiselle mallille. SimpleUniverse-olion kontruktori luo maisemagraafin, joka sisältää *VirtualUniverse* ja *Locale* –oliot, sekä valmiin *näkymäpuurakenteen* (view branch graph) (kuva 30).



Kuva 30. SimpleUniverse-olion avulla luotu virtuaalinen universumi, joka on merkitty katkoviivalla (Bouvier, 2001).

Canvas3D-luokka sisältää *piirtoalustan* (canvas) kolmiulotteiselle renderöinnille. SimpleUniverse-luokan metodin *getPreferredConfiguration()* avulla saadaan optimaaliset grafiikan renderöintiin liittyvät asetukset järjestelmästä, jossa sovellusta suoritetaan. Virtuaalisen universumin luominen käyttäen SimpleUniverse-luokkaa on esitetty kuvassa 31.

```
SimpleUniverse simpleUniverse;
GraphicsConfiguration config =
                SimpleUniverse.getPreferredConfiguration();

Canvas3D canvas3D = new Canvas3D(config);
simpleUniverse = new SimpleUniverse(canvas3D);
```

Kuva 31. Virtuaalisen universumin luominen käyttäen optimaalisia järjestelmän asetuksia.

4.2.2 Geometrian lataaminen

Java 3D rajapinnalle on toteutettu kymmeniä valmiita kirjastoja, joiden avulla voidaan virtuaaliseen ympäristöön ladata sisältöä (content loader) mallinnusohjelmien tiedostoformaateista, kuten esimerkiksi CAD, Milkshape 3D ja 3D-studio (Wright, 2007). UWVF-laskentaverkko formaatille ei valmista sisällön latauspakettia ollut saatavilla, joten se tuli toteuttaa ensimmäisenä. UWVF-laskentaverkkoformaatti sisältää pääasiassa listauksen kolmioita. Formaatti sisältää myös tiedon, kuinka näistä kolmioista muodostuu laskentaverkon tilavuudet.

Toteutuksessa käytettiin GeometryInfo-luokkaa säilyttämään geometria informaatioita. Valintaan vaikutti se, että tämän luokan avulla geometriaa pystytään optimoimaan monipuolisemmin kuin muilla tekniikoilla. GeometryInfo-luokka sisältää työkaluja muun muassa käyttämättömän ja käyttäjälle näkymättömän informaation poistoon geometriasta. Koska esikäsittelijän tulee pystyä käsittelemään laskentaverkkoja, joiden geometrinen rakenne koostuu jopa miljoonista kolmioista, korostuu optimoinnin tarve toteutuksessa.

Geometriset tilavuudet ovat jaoteltu omiin GeometryInfo-olioihinsa. Geometria ladataan GeometryInfo-olioon samalla periaatteella kuin käytettäessä GeometryArray-metodeja. GeometryInfo-olion konstruktorille annetaan parametrina *lippu* (flag), joka määrittää millaista primitiivistä geometriaa olioon ladataan (Bouvier, 2001). Koska UWVF-laskentaverkko koostuu kolmioista, konstruktori saa syötteenä *GeometryIn-*

fo.TRIANGLE_ARRAY lipun. Striptifier-luokkaa käytetään optimoimaan kolmiot yhdeksi yhtenäiseksi geometriseksi rakenteeksi (kuva 32). Striptifier-luokan avulla voidaan geometriasta poistaa täysin identtiset rakenteet ja näin vähentää visualisoinnissa tarvittavaa laskentatehoa. Jokaiselle visualisoitavalle geometrialle luodaan oma GeometryArray-olio. Yksittäisten tilavuuksien sisältämät solmupisteet kerätään UWVF-tiedostosta omiin tietorakenteisiinsa. Tietorakenne sisältää alkioita, jotka puolestaan sisältävät x, y ja z koordinaatit reaalilukuina. Metodi setCoordinates() saa parametrinaan listauksen kolmioiden solmupisteitä, joiden perusteella geometria muodostetaan. Metodi getGeometryArray() palauttaa prosessoidun geometrian GeometryArray-oliona.

```
GeometryInfo gi = new GeometryInfo(GeometryInfo.TRIANGLE_ARRAY);
gi.setCoordinates(verticesPoints);
// stripify
Stripifier st = new Stripifier();
st.stripify(gi);
GeometryArray result = gi.getGeometryArray();
```

Kuva 32. Luodaan haluttu geometria solmujoukolla, jolle kutsutaan striptify –metodia optimoimaan geometrian rakenne. Lopuksi luodaan GeometryArray olio, johon geometria asetetaan.

Tilavuusgeometriat asetetaan Shape3D-olioihin, jotka lisätään Switch-solmun lapsiksi. Switch-luokan avulla voidaan kontrolloida, mitkä sen lapsisolmuista renderöidään. Tämän avulla tilavuusgeometrioita voidaan asettaa tarvittaessa näkymättömiksi. Switch-solmu puolestaan lisätään *TransformGroup*-solmun lapseksi. TransformGroup-luokan avulla voidaan lapsi-solmujen geometrioita skaalata, siirtää ja kääntää. Metodilla setCapability() voidaan kontrolloida, mitä operaatioita solmulle voidaan suorittaa. TransformGroup-solmu asetetaan lopuksi BranchGroup-solmun lapseksi.

```
TransformGroup objSpin = new TransformGroup();
objSpin.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);
objSpin.setCapability(TransformGroup.ALLOW_TRANSFORM_READ);
// switchGroup-solmun lapsina ovat Shape3D-oliot, jotka sisältävät
// laskentaverkon tilavuusgeometriat.
objSpin.addChild(switchGroup);

BranchGroup objRoot = new BranchGroup();
objRoot.addChild(objSpin);

objRoot.compile();
simpleUniverse.addBranchGraph(objRoot);
```

Kuva 33. Esimerkki geometrioiden liittämiseen virtuaaliseen universumiin.

BranchGroup-solmun sisältämä puurakenne voidaan optimoida renderöintiä varten compile() metodin avulla. Lopuksi BranchGroup-solmu liitetään virtuaaliseen universumiin, jonka jälkeen puurakenteen sisältämät geometriat automaattisesti renderöidään (kuva 33). Piirtoalusta voidaan liittää esimerkiksi *javax.swing* -paketin sisältämään JPanel-komponenttiin.

4.2.3 Interaktio virtuaalisessa universumissa

Interaktio kolmiulotteisten objektien kanssa voidaan toteuttaa helposti käyttämällä *com.sun.j3d.utils.behaviors.vp* –paketin sisältämää OrbitBehavior-luokkaa. OrbitBehavior-luokka sisältää valmiin implementoinnin perusinteraktioille, kuten tarkastelupisteen siirtämiseen kauemmaksi tai lähemmäksi objektista ja tarkastelupisteen rotaatiolle objektin ympärillä. OrbitBehavior-olion konstruktorissa määritetään piirtoalusta (Canvas3D), johon olio kiinnitetään. Tämän jälkeen OrbitBehavior-olio kiinnitetään tarkastelupisteeseen (viewingPlatform), jonka jälkeen tarkastelupisteen sijaintia voi muuttaa hiiren tai näppäimistön avulla (kuva 34).

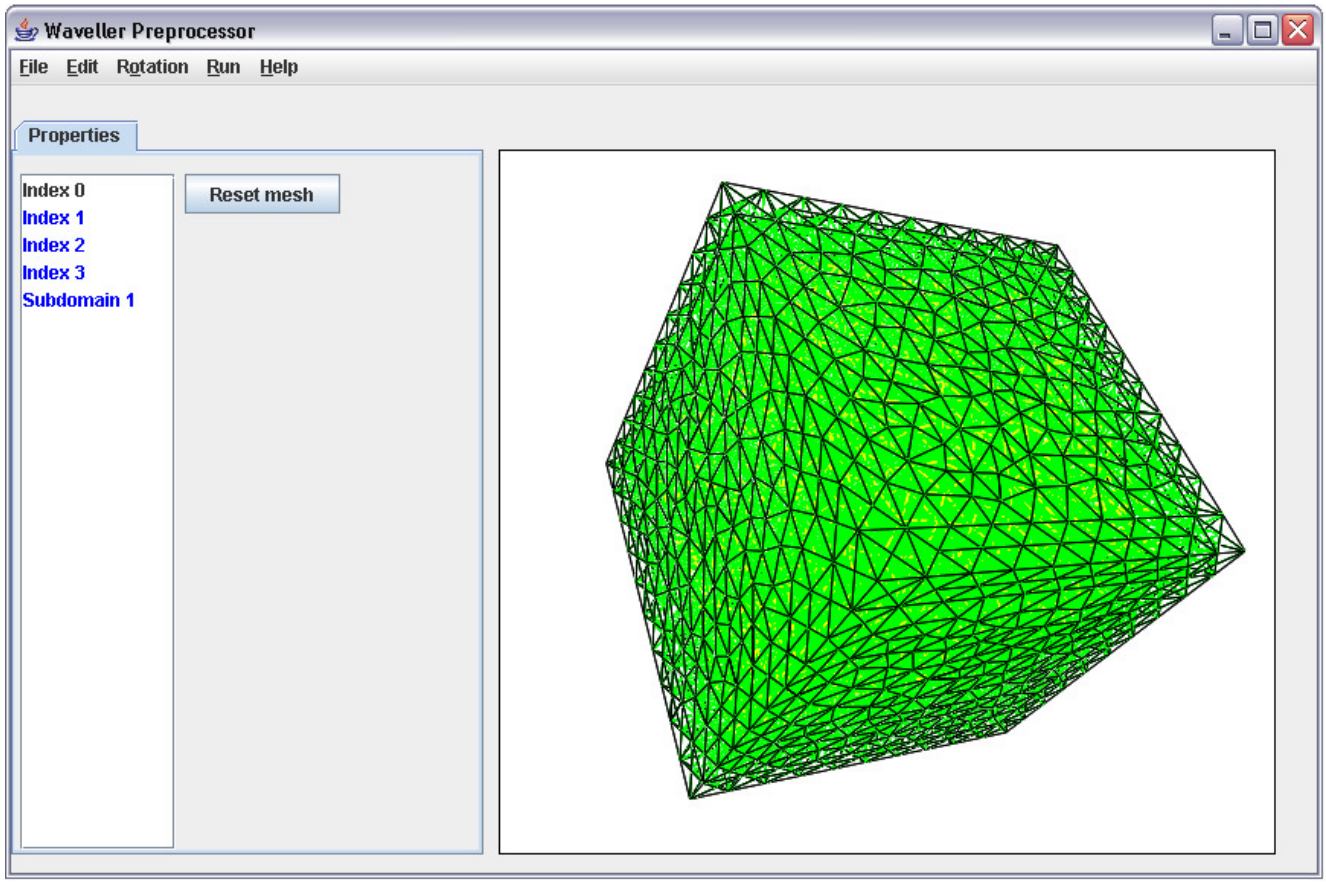
```
BoundingSphere bounds = new BoundingSphere(new Point3d(), 1000.0);
OrbitBehavior orbit =
     new OrbitBehavior(canvas3D, OrbitBehavior.REVERSE_ALL);
orbit.setSchedulingBounds(bounds);

ViewingPlatform viewingPlatform = simpleUniverse.getViewingPlatform();
viewingPlatform.setViewPlatformBehavior(orbit);
```

Kuva 34. Esimerkki OrbitBehavior-olion luomisesta, joka kiinnitetään ViewingPlatform alustaan.

4.3 Malliesimerkki

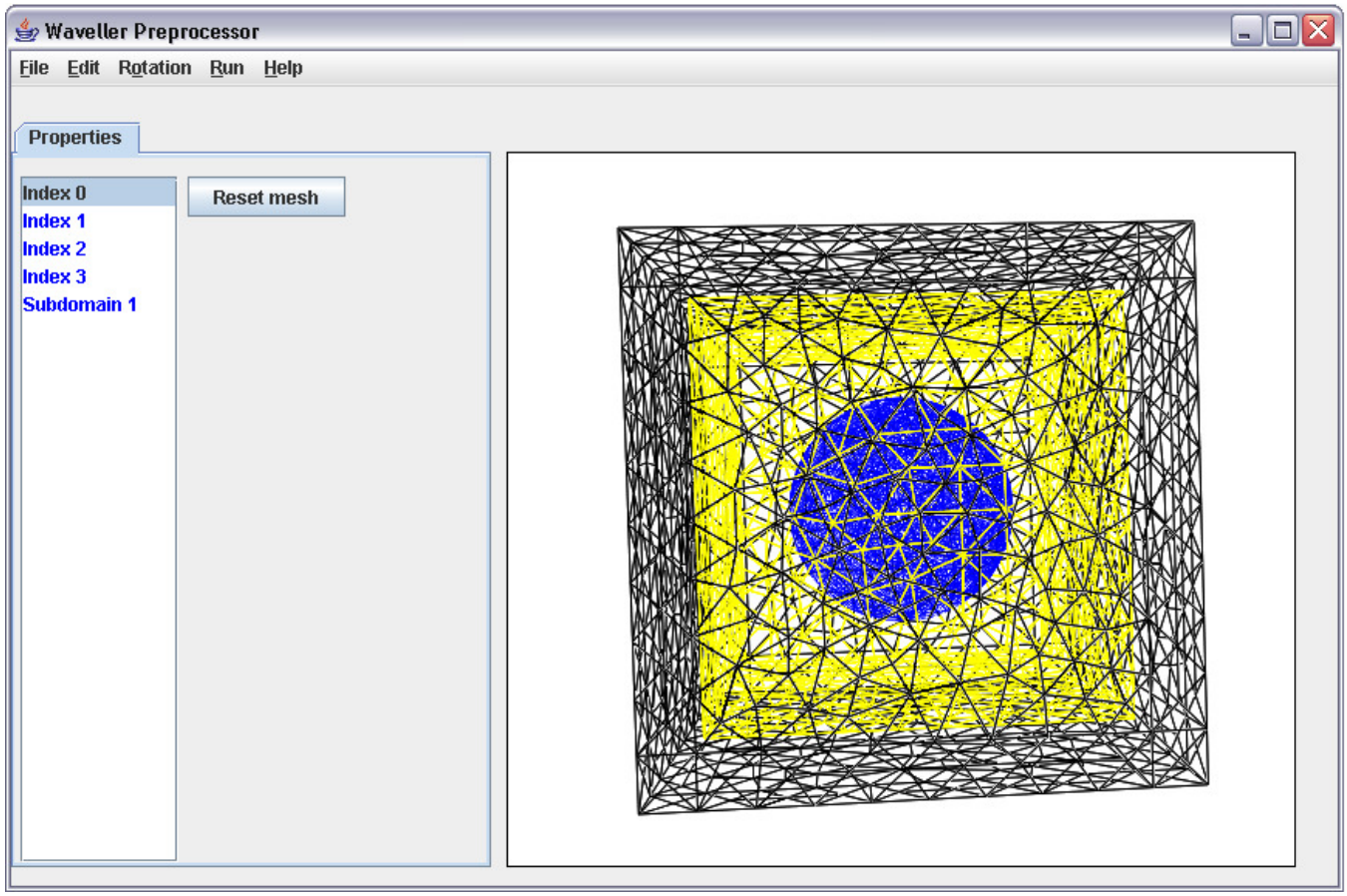
Tutkimuksen esimerkkiongelmana halutaan ratkaista, miten ääniaalto käyttäytyy, kun se osuu kovaan kappaleeseen. Ääniaallon taajuutena on 10 000 Hz ja väliaineena ilma. Esimerkkiongelmassa äänennopeus on 340 metriä sekunnissa ja ilman tiheys 1,2 Kilogrammaa per kuutiometri. Ongelmanratkaisussa ei huomioida ääniaallon vaimenemista.

Ongelmanratkaisemiseen käytetään elementtimenetelmää ja tuloksena halutaan saada selville, kuinka ääniaalto siroaa akustisesti kovasta kappaleesta. Esimerkissä käytettävä laskentaverkko on generoitu käyttäen Comsol Multiphysics –ohjelmistoa, jonka jälkeen se on konvertoitu UWVF-laskentaverkkotiedostoformaattiin, jotta Waveller-esikäsittelijä pystyy käsittelemään laskentaverkkoa. Laskentaverkko on tyypiltään rakenteeton ja elementtityyppinä on nelitahokas. Laskentaverkko sisältää 37 591 nelitahokasta, eli yhteensä 150 364 kolmioelementtiä. Kuvassa 35 Waveller esikäsittelijään on ladattuna tutkittavan ongelman laskentaverkko.



Kuva 35. Waveller-esikäsittelijä, johon on ladattuna ongelmanratkaisuun käytettävä laskentaverkko.

Tutkittava laskentaverkko sisältää neljä pintaa ja yhden osatilavuuden. Pinta nolla sisältää kaikki pinnat, jotka eivät ole laskentaverkon ulkopinnalla. Ongelmanratkaisemisen kannalta pinta nolla on irrelevantti. Tämä pinta on kuvassa 35 vihreällä visualisoitu geometria. Pinta numero yksi on laskentaverkon ulkoreunan pinta, joka on kuvassa 36 visualisoitu mustana. Kuvassa 36 pinta numero kaksi on visualisoitu sinisellä ja pinta numero kolme keltaisella värityksellä. Eri pintojen visualisoinnin helpottamiseksi pinta nolla on poistettu näkyvistä kuvan 36 laskentaverkosta.



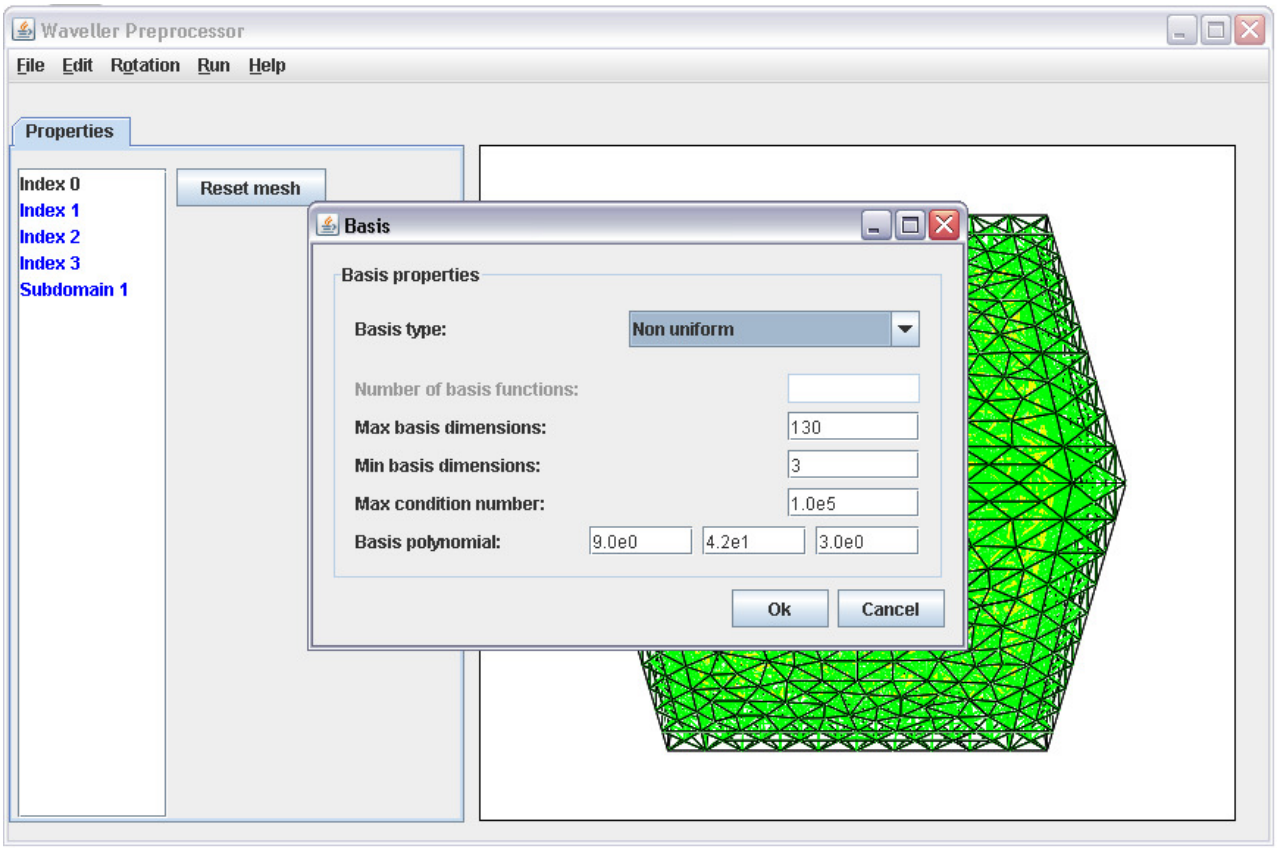
Kuva 36. Laskentaverkon pinnat. Laskentaverkon visualisoinnista on poistettuna pinta numero nolla.

Pinta numero kaksi on ongelman kova kappale, joka tässä tapauksessa on pallo. Pallon halkaisija on 50cm. Osatilavuuksia (subdomain) ei tässä tapauksessa ole kuin yksi. Osatilavuudet kuvaavat yleensä jotain tiettyä fyysistä aluetta laskenta-alueessa. Tässä tapauksessa osatilavuus on laskentaverkon ulkoreuna pinta.

4.3.1 Ongelmanratkaisemiseen asetettavat parametrit

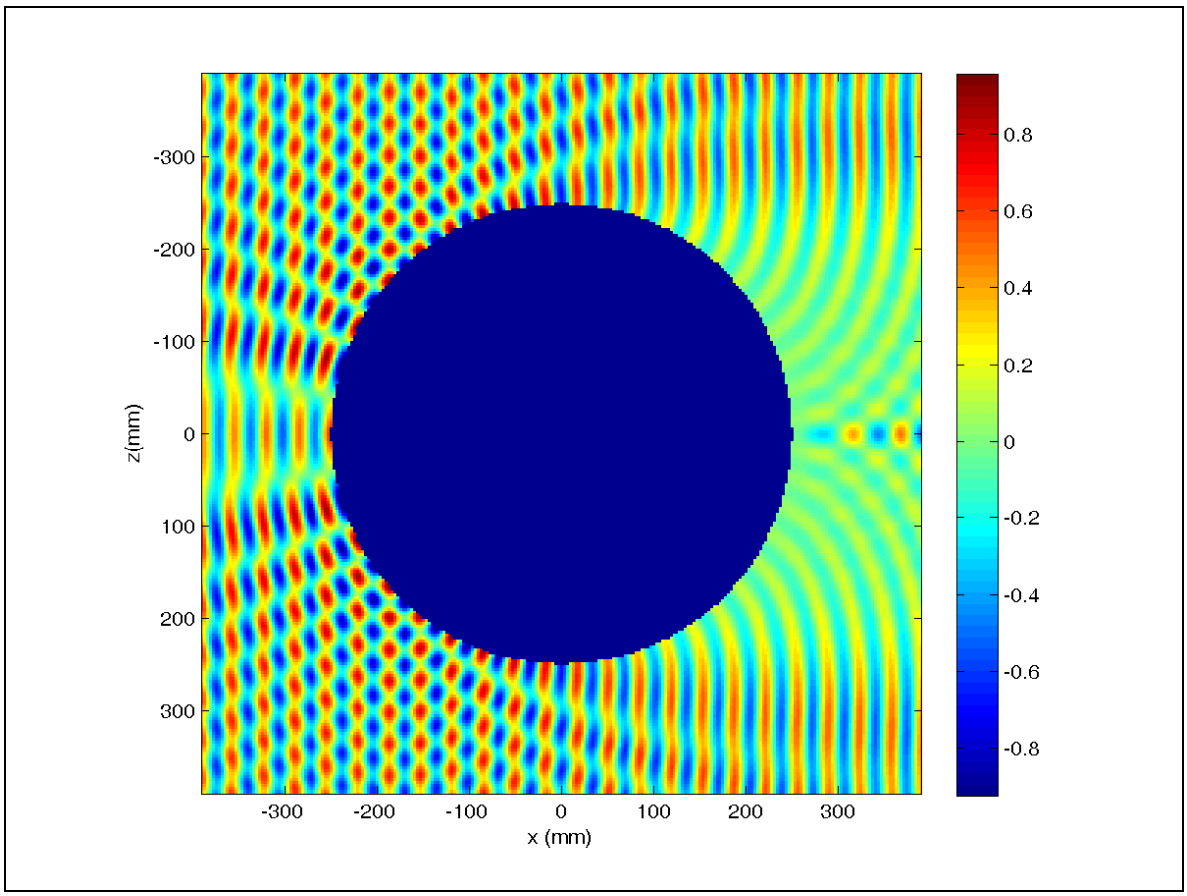
Esikäsittelijän avulla asetetaan ongelmanratkaisemiseen tarvittavat parametrit laskentaverkolle. Kantafunktio-parametrit sisältävät ratkaisijan käyttämän UWVF menetelmän tasoaaltokantafunktioiden lukumäärän (kuva 37). Tulostevirta-parametrit sisältävät määritelmät

ulostulotiedostoille ja laskennan etenemisen aikana ruudulle tulostettavan tiedon määrän kontrolloinnin. PML-parametrit (perfectly matched layer) sisältävät ominaisuudet laskennalliselle vaimennuskerrokselle, jota käytetään suurien laskenta-alueiden rajoittamiseksi pienemmäksi. Ratkaisijaan liittyvät parametrit (solver parameters) määrittelevät numeerisen menetelmän, jolla ratkaistaan UWVF-menetelmän matriisiyhtälö.



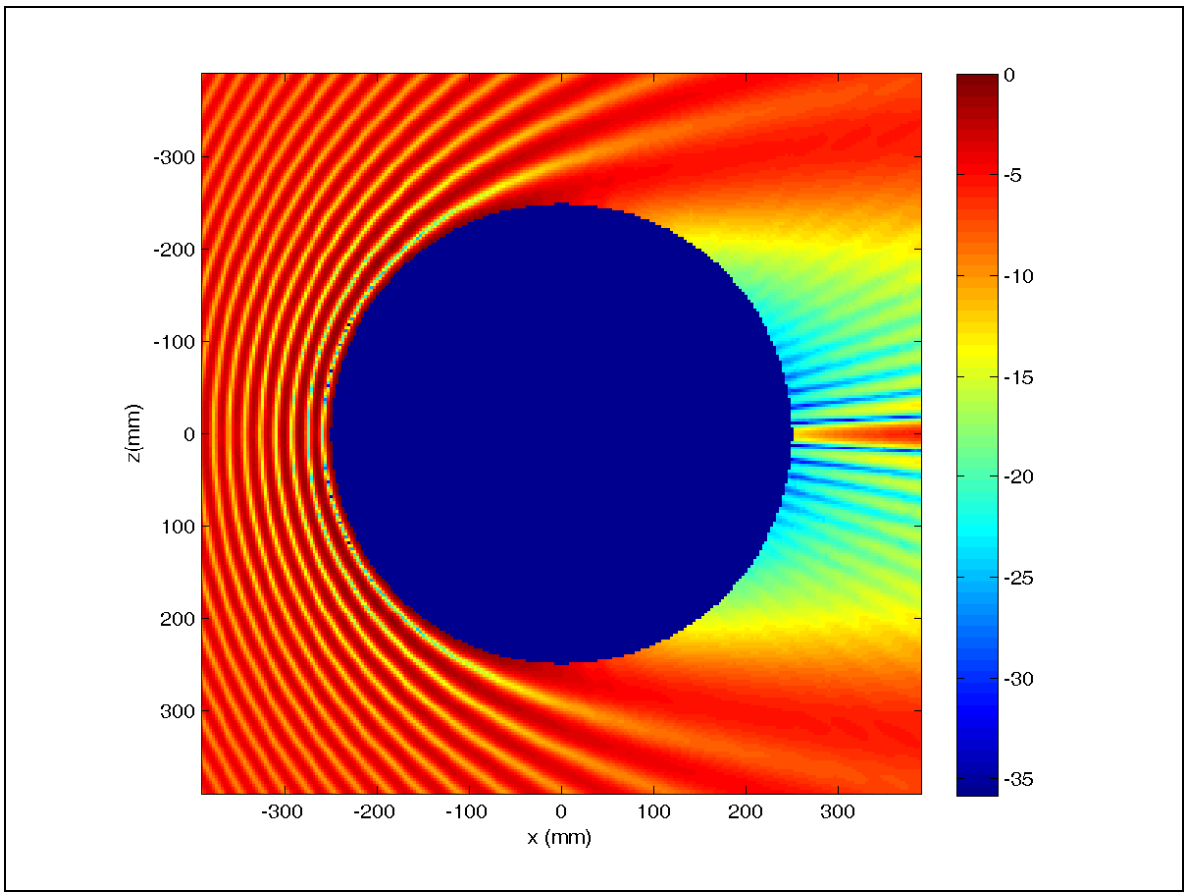
Kuva 37. Kantafunktioiden määrittely laskentaverkolle Waveller-esikäsittelijällä.

Esimerkkitapauksessa reunaehtoja on kaksi: akustisesti kova reunaehto pallon pinnalle ja niin kutsuttu absorpoiva reunaehto laskenta-alueen ulkoreunalle. Jälkimmäisen tarkoitus on mallintaa tilannetta jossa aallot pääsevät etenemään vapaasti alueesta ulos. Ongelmanratkaisemisessa käytetyt arvot löytyvät tarkemmin liitteestä 1. Ongelmanratkaisemiseen käytettiin 24 tietokoneen klusteria. Tietokoneet sisälsivät Pentium 4-suorittimet. Laskentaan kului 790 sekuntia.



Kuva 38. Kovan kappaleen ympärille muodostuva äänenpaine.

Kuvat 38 ja 39 esittävät akustisen tasoaallon siroamista kovasta kappaleesta. Kuva 38 havainnollistaa kovan kappaleen ympärille muodostuvaa äänenpainetta ja kuva 39 paineamplitudia desibeliasteikolla.



Kuva 39. Kovan kappaleen ympärille muodostuva paineamplitudi desibeliasteikolla.

Tämän tyyppinen ongelma tulee ratkaista esimerkiksi kun mallinnetaan akustista painekenttää ihmisen pään ympärillä (Kirkeby & al., 2007). Kuvista nähdään äänikentän taittuminen kovan kappaleen ympäristössä sekä hiljaisemman, nk. varjoalueen, muodostuminen kappaleen taakse. Desibeliasteikolla esitetyssä kuvassa voidaan varjoalueessa kuitenkin havainta diffraktiokuvio, jonka erityispiirteenä on voimakas paineamplitudimaksimi kappaleen takapuolella.

5. Yhteenveto

Tutkimuksen tavoitteena oli selvittää Java3D-ohjelmointirajapinnan soveltuvuutta akustisen mallintamisen alueeseen kuuluvan elementtimenetelmän toteutukseen. Aihetta ei ole aikaisemmin tutkittu, joten sitä voidaan pitää tieteellisessä mielessä mielenkiintoisena. Teoriaosuudessa käsittelin akustiikan, akustisen mallintamisen ja elementtimenetelmien perusteita käyttäen havainnollistavia esimerkkejä. Lisäksi tutkielmassani perehdyttiin laskentaverkkoihin ja laskentaverkkojen generointimenetelmiin.

Käytännön osuudessa toteutettiin Waveller-esikäsittelijä käyttäen Java3D-ohjelmointirajapintaa. Tutkielmassa tutustuttiin Java3D-ohjelmointirajapinnan tarjoamiin kirjastoihin, rajapinnan toimintaperiaatteisiin ja näiden avulla toteutettiin Waveller-esikäsittelijä. Waveller-esikäsittelijän tarkoituksena on havainnollistaa käyttäjälle UWVF-laskentaverkon sisältämiä tilavuuksia ja tarjota mahdollisuus asettaa laskennassa käytettäviä parametreja UWVF-laskentaverkolle.

Tutkimus osoitti, että Java3D-ohjelmointirajapinta soveltuu riittävän hyvin laskentaverkkojen visualisoimiseen ja antaa tarvittavat työkalut esimerkiksi interaktiivisuuden toteuttamiseen suhteellisen vaivattomasti. Mielestäni Java3D-ohjelmointirajapinta ei sovellu kaikkiin elementtimenetelmissä liittyviin toteutuksiin johtuen siitä, että se on korkean tason rajapinta. Rajapinnan avulla on esimerkiksi hyvin hankalaa toteuttaa tilanne, jossa yksittäinen geometria pitäisi leikata halki tarkastelua varten. Lisäksi muistihallinta ja rajapinnan käyttämä muistin määrä asettaa omat rajoituksensa laskentaverkkojen visualisoinnille. Uskon kuitenkin, että rajapinnan kehittyessä eteenpäin saadaan muistin käyttöön liittyvät ongelmat hallintaan ja rajapintaa pystytään tulevaisuudessa käyttämään monipuolisempiin sovelluksiin kuin mitä nykypäivänä.

VIITELUETTELO

Ballou G.M. (2002) *Handbook for Sound Engineers Third Edition.* Focal Press, United States of America.

Beall., Mark W., Walsh J., Shephard M.S. (2003) Accessing CAD geometry for mesh generation. *12th International Meshing Roundtable*, Sandia National Lab, 33-42.

Bouvier D.J (2001) *Java 3D API Tutorial.* Sun Microsystems. Myös WWW-sivusto, http://java.sun.com/developer/onlineTraining/java3d/ (20.10.2007).

Chew P.L. (1989) *Guaranteed-Quality Triangular Meshes*, Department of Computer Science Tech Report 89-983, Cornell University, Ithaca, New York.

Comsol. (2006) *Comsol Multiphysics User's Guide*, COMSOL AB.

Comsol. (2007) *Comsol OY*. WWW-sivusto, http://www.comsol.fi/ (2.5.2007).

CSC. (2000a) *ELMER Front's User Guide.* http://www.nic.funet.fi/pub/sci/physics/elmer/doc/ElmerFrontUserGuide.pdf (23.6.2007).

CSC. (2000b) *ELMER User's Guide.* http://www.nic.funet.fi/pub/sci/physics/elmer/doc/OldElmerTutorial.pdf (23.6.2007).

CSC. (2007) *Overview of Elmer.* http://www.nic.funet.fi/pub/sci/physics/elmer/doc/ElmerOverview.pdf (23.6.2007).

Deines E., Bertram M., Mohring J., Jegorovs J., Michel F., Hagen H., Nielson G.M (2006) Comparative Visualization for Wave-based and Geometric Acoustics. *IEEE Transactions on Visualization and Computer Graphics*, **12**(5), 1173-1180.

Du D., Hwang F. (1992) *Computing in Euclidean Geometry (Lecture Notes Series on Computing, Vol 1).* World Scientific Publishing Co., Inc, New Jersey.

Ern A., Guermond J.L. (2004) *Theory and Practice of Finite Elements*. Springer Verlag, New York.

Everest F.A (2001) *Master Handbook of acoustics*. McGraw-Hill, R.R. Donnelley & Sons Company.

Felippa C.A (2004). *Introduction to Finite Element Methods*. http://www.colorado.edu/CAS/courses.d/IFEM.d/IFEM.Ch07.d/IFEM.Ch07.pdf (12.6.2007).

Flaherty J.E., Eaton A. (2000) *Finite Element Analysis, Lecture notes.* Rensselaer Polytechnic Institute, New York.

George, P.L. (1992) *Automatic Mesh Generation: Applications to Finite Element Methods*. John Wiley & Sons, Inc, New York.

Haataja J., Käpyaho J., Rahola J. (1993) *Numeeriset menetelmät.* CSC-Tieteellinen laskenta Oy, Yliopistopaino.

Haataja J., Heikonen J., Leino Y., Rahola J., Ruokolainen J., Savolainen V. (2002) *Numeeriset menetelmät käytännössä*. Picaset Oy, Helsinki.

Huttunen T. (2004) *The ultra weak variational formulation for ultrasound transmission problems.* Väitöskirja, Kuopion yliopisto, fysiikan laitos.

Hämäläinen J., Järvinen J. (2006) *Elementtimenetelmä virtauslaskennassa.* CSC - Tieteellinen laskenta Oy, Espoo.

Ihlenburg F. (1998) *Finite Element Analysis of Acoustic Scattering.* Springer, United States of America.

Ito Y., Shih A.M., Soni B.K (2004) Reliable isotropic tetrahedral mesh generation based on an advancing front method. *Proceedings, 13th International Meshing Roundtable,* Sandia National Laboratories Williamsburg, VA, 95-106.

Keränen J. (2006) *Akustinen mallintaminen työpaikkojen meluntorjuntasuunnittelussa.* Lisensiaatintyö, Turun yliopisto, fysiikan laitos.

Kirkeby O., Seppälä E., Kärkkäinen A., Kärkkäinen L., Huttunen T. (2007) Some Effects of The Torso on Head-Related Transfer Functions. *122nd Audio Engineering Society Convention* (toim. Audio Engineering Society), Journal of the Audio Engineering Society, 1-8.

Kuava. (2005) Waveller Acoustics User's guide, Kuava Inc.

Kuava. (2006) Waveller Acoustics Command Line Manual, Kuava Inc.

Kärkkäinen L. (2003) *Akustiikan Sovelluksia – elementtimenetelmät*. http://www.physics.helsinki.fi/common/kurssienkoti/akustiikka/luku6.pdf (2.9.2007).

Lee Y.T. (1984) Automatic Finite-Element Mesh Generation from Geometric Models-A Point-Based Approach. *ACM Transactions on Graphics,* **3**(4*),* 287-311.

Lewis R.W., Nithiarasu P., Seetharamu K.N. (2004) *Fundamentals of the Finite Element Method for Heat and Fluid Flow*. John Wiley & Sons, Ltd, Chichester.

LMS International (2007) *LMS Virtual.Lab*. WWW-sivusto, http://www.lmsintl.com/simulation/Lmsvirtuallab (6.7.2007).

NETGEN. (2007) *NETGEN.* WWW-sivusto, http://www.hpfem.jku.at/netgen/ (22.6.2007).

Numerola. (2007) *Waveller.* WWW-sivusto, http://www.numerola.fi/fi/waveller.php?corp=kuava (18.5.2007).

Owen S.J. (1998) A survey of unstructured mesh generation technology. *7th International Meshing Roundtable*, Sandia National Lab, 239–267.

Randal J.L. (2002) *Finite Volume Methods for Hyperbolic Problems*. Press syndicate of the university of Cambridge, Cambridge.

Ruppert J. (1992) A New and Simple Algorithm for Quality 2-Dimensional Mesh Generation. *Technical Report UCB/CSD 92/694*, University of California at Berkely, Berkely California

Råback P., Vierinen J. (2005) Monifysikaalisen mallinnusohjelmiston kehitystarina. *Infors,* **2005**(2), 9-11.

Savioja L. (1999) *Modeling Techniques for Virtual Acoustics.* Picaset, Espoo.

Schöberl J. (2004) *NETGEN - 4.3*, WWW-sivusto, http://www.hpfem.jku.at/netgen/ (22.6.2007).

Selman D. (2002) *Java 3D programming.* Manning Publications Co, Greenwich.

Stettner A., Greenberg D.P (1989) Computer Graphics Visualization For Acoustic Simulation. *Computer Graphics,* **23**(3), 195-206.

Thompson J.F., Soni B.K., Weatherill N.P. (1999) *Handbook of Grid Generation.* CRC Press LLC, United States of America.

Toivanen J.I. (2004) *3D-verkon generointi hybridimenetelmällä*. Tietotekniikan pro gradu -tutkielma, Jyväskylän yliopisto.

Weatherill N.P., Hassan O. (1994) Efficient Three-dimensional Delaunay Triangulation with Automatic Point Creation and Imposed Boundary Constraints, *International Journal for Numerical Methods in Engineering*, (37), 2005-2039.

Wolf M.M. (2005) *Intro to Mesh Generation.* http://www.cs.uiuc.edu/homes/mmwolf/presentations/CS591MH/CS591MH_20050420.pdf (10.9.2007).

Wright J. (2007) *File Loader Archives*. WWW-sivusto, http://java3d.j3d.org/utilities/loaders.html (7.8.2007).

Yue W., Guo Q., Zhang J., Wang G. (2007). 3D triangular mesh optimization in geometry processing for CAD. *Proceedings of the 2007 ACM Symposium on Solid and Physical Modeling* (toim. Bruno Levy), ACM Press, 23-33.

LIITE 1: Esimerkkiongelman ratkaisussa käytetyt parametri tiedostot

Materiaali parametrit:

```
340.0d0             !  c (real)  for material index 1
0.0d0               !  alpha1 (real) for material index 1
0.0d0               !  alpha0 (real) for material index 1
1.2d0               !  rho (real) for material index 1
```

Äänilähteet ja reunaehdot:

```
3500.0d0, 0.0d0, 3500.0d0      ! frequency
1                    ! surf 1: passive absorbing surface
(0.0d0,0.0d0)        ! Q for surf 1
1.0d0                ! sigma for surf 1: 1.0d0  -> kappa/rho
2             ! surf 2: rigid surface emits the plane wave
(1.0d0,0.0d0)        ! Q for surf 2
1.0d0                ! sigma for surf 2: 1.0d0  -> kappa/rho
1.0d0,0.0d0,0.0d0    ! (dx,dy,dx)-direction of the plane wave
1.0d0                ! amplitude of the plane wave
0.0d0                ! phase of the plane wave
0                    ! surf 3: internal surface
0,0               ! number and type of volume (point)sources
```

Kantafunktioiden määrittely:

```
2             !basis type: 2=non uniform
1.0d5         !max condition number
3             !min basis dimension
130           !max basis dimension
```

```
9.0d0,42.0d0,3.0d0   !initial basis polynomial
```

Määritelmät numeeriselle menetelmälle jolla ratkaistaan UWVF-menetelmän matriisiyhtälö:

```
1               !solver: 1=bicg
1.0d-6          !termination tolerance
2000            !maximum number of iter.
```

Määrittelyt laskennalliselle vaimennuskerrokselle:

```
0.4d0   ! |x|=0.4
3.0d0   ! x-decay parameter
0.1     ! PML thickness in x-dir.
0.4d0   ! |y|
3.0d0   ! y-decay parameter
0.1     ! PML thickness in y-dir.
0.4d0   ! |z|
3.0d0   ! z-decay parameter
0.1     ! PML thickness in z-dir.
```

Määritelmät ratkaisijan antamille tulostevirroille:

```
2     ! print progress: 0=none, 1=some, 2=all
1     ! print infofile: 0=no, 1=yes
1     ! print X-file; 0=no, 1=yes
1     ! print P-file; 0=no, 1=yes
1     ! print convergence-file; 0=no, 1=yes
0     ! compute solution at nodes
'0'   ! solution points: '0'=no or filename
```

```
'0'   ! compute the far-field pattern:  '0'=no or filename
```