Virusmallin pintaesityksen yksinkertaistaminen

Jan Lindström (jplindst@cs.helsinki.fi)

Algoritmisia ongelmia rakennebiologiassa ja geenitekniikassa

Helsingin yliopisto, Tietojenkäsittelytieteen laitos

Ohjaaja: Teemu Kivioja

1 Johdanto

Tämä työ on osa projektia, jossa toteutetaan menetelmiä Viikin Biokeskuksessa. Visualisointityökalua tarvitaan avustamaan virusten analysointia. Tämän työn tavoitteena on kehittää menetelmä, jolla viruksen visualisointia voitaisiin yksinkertaistaa. Päätavoitteena on luoda yksinkertaistettu kolmiulotteinen malli viruksesta.

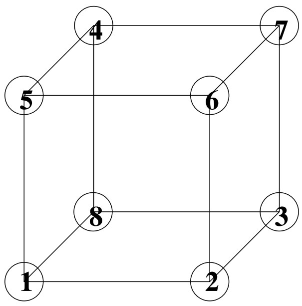
Virusmalli esitetään tiheyskarttana eli kolmiulotteisena taulukkona, jonka jokainen alkio on tiheysarvo. Valmiiden virusmallien visualisoinnissa tärkeä menetelmä on niin sanotun *tasa-arvopinnan* (isosurface) laskeminen [LE87, WW92]. Tasa-arvopinnan laskeminen on yksi menetelmä muuntaa tiheyskartta sellaiseen muotoon, että sitä voidaan tarkastella näytöllä. Menetelmässä muodostetaan tiheyskartasta pintoja, joiden sisällä tiheys on käyttäjän antamaa raja-arvoa suurempi. Biologiset päätelmät virusmalleista tehdään usein juuri tällaisista pintaesityksistä.

Virusmalleista muodostetuissa tasa-arvopinnoista tulee usein hyvin monimutkaisia: pinta muodostuu sadoista tuhansista tai jopa miljoonista kolmioista. Tällaisen esityksen käsittely, esimerkiksi lataaminen verkosta, on hankalaa. Pintaesityksen kolmioiden lukumäärää tulisi radikaalisti pienentää siten, että kohteen oleelliset piirteet säilyvät.

2 Pintaesityksen luonti

VVV-ohjelmisto lukee tiheyskartan mrc-tiedostosta ja muodostaa tiheyskartasta pintaesityksen eli tasa-arvopinnan Marching Cubes -algoritmilla [LE87, WW92].

Kahdeksasta naapuriarvosta luodaan looginen kuutio, jolla tiheyskartta läpikäydään (Kuva 1). Kuution jokaisessa kulmassa olevaa tiheysarvoa verrataan käyttäjän antamaan raja-arvoon. Kuution kulma on pinnan sisällä tai pinnassa, jos tiheysarvo käsiteltävässä kulmassa on sama tai ylittää käyttäjän antaman raja-arvon. Kuution kulmat, joiden tiheysarvo on raja-arvoa pienempi ovat pinnan ulkopuollella. Pinta leikkaa niitä kuution särmiä, joissa yksi kulma on pinnan ulkopuolella ja toinen kulma on pinnan sisäpuolella. Pinnan leikkauspiste kuution särmässä selvitetään lineaarisella interpolaatiolla.



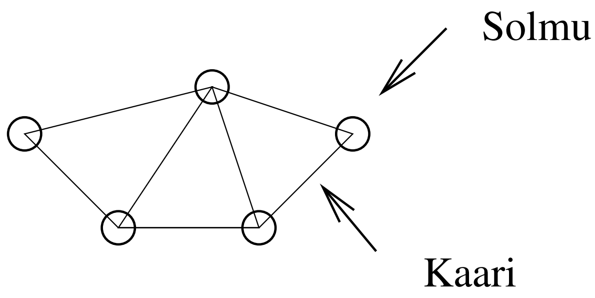
Kuva 1: Looginen kuutio.

Kuutiolla on kahdeksan kulmaa ja jokaisella särmällä on kaksi mahdollista tilaa, joten pinta voi leikata kuutiota 256 eri tavalla. Käyttämällä komplementti- ja kiertämissymmetrioita voidaan eri tapausten lukumäärä vähentää 14 malliin. Permutoimalla näitä 14 perusmallia voidaan luoda kaikki 256 tapausta taulukkoon. Taulukkoon on talletettu myös tarvittava pinnan kolmiointi.

Käsittelyssä olevaa kuutiota verrataan taulukkoon generoituihin tapauksiin. Käsiteltävä kuutio kolmioidaan taulukosta löytyvän mallin perusteella. Tuloksena syntyy kolmioista muodostettu pintaesitys. Pintaesitys voi koostua suuresta määrästä kolmioita.

3 Pintaesityksen yksinkertaistaminen

Pintaesitys koostuu *solmuista* (vertex) ja niiden välisistä *kaarista* (edge) (Kuva 2). Tavoitteena on vähentää pintaesitykseen tarvittavien kolmioiden lukumäärää, jolloin pintaesitys yksinkertaistuu.



Kuva 2: Pintaesitys.

Pintojen yksinkertaistamiseen on kirjallisuudessa esitetty lukuisia menetelmiä [HG97, SZL92, GH97]. Yleisesti yksinkertaistamismenetelmistä voidaan muodostaa seuraava luokitus:

- Solmuyksinkertaistamismenetelmät (*vertex decimation methods*) poistavat solmun ja uudelleenkolmioivat solmun naapuriston.
- Kaariyksinkertaistamismenetelmät (*edge decimation methods*) poistavat yhden kaaren, kaksi kolmiota ja yhdistävät kaksi solmua.
- Kolmioyksinkertaistamismenetelmät (*triangle decimation methods*) poistavat yhden kolmion, kolme kaarta, yhdistävät kolme solmua ja uudelleenkolmioivat naapuriston.
- Paikkayksinkertaistamismenetelmät (*patch decimation methods*) poistavat useita vierekkäisiä kolmioita ja uudelleenkolmioivat niiden rajan.

Schroeder ym. [SZL92] kehittivät yleisen solmuyksinkertaistamismenetelmän tieteelliseen visualisointiin. Menetelmälle annetaan syötteeksi kolmioitu pinta. Algoritmi käsittelee syötteen useita kertoja kunnes haluttu virhetaso on saavutettu. Jokaisella käsittelykerralla kaikki solmut, jotka eivät ole rajalla tai joiden virhe on halutun raja-arvon alapuolella poistetaan ja ympäröivät kolmiot uudelleenkolmioidaan. Kyseinen menetelmä on osa VTK (The Visualization

Toolkit) [SML96] järjestelmää. Järjestelmä on hyvin suuri ja monimutkainen, mikä käytännössä esti järjestelmän tai järjestelmän osan käyttämisen tässä työssä. Järjestelmää kannattaa kuitenkin harkita jatkossa.

JADE (Just Another DEcimator) [CCMS97] yksinkertaistamismenetelmä perustuu verkkoyksinkertaistukseen (*mesh decimation*), joka edellä mainitussa luokittelussa olisi solmuyksinkertaistamismenetelmä. Järjestelmästä on saatavilla vain SGI binäärit, joten järjestelmää ei pystytty edes testaamaan.

Algoritmit kuten Simplification envelopes [CVM$^+$96] ja Mesh Optimization [HDD$^+$93] ja Memoryless Simplification [LT98] olisivat olleet myös kiinnostavia testata, mutta joko toteutus ei ollut julkisesti saatavilla, toteutus oli saatavilla vain SGI koneeseen tai toteutuksen muokkaaminen sopivaan muotoon olisi ollut tälle työlle liian työlästä.

Lindstrom [LT98] on esittänyt Metro-ohjelmalla [CCMS96] tehtyjä mittaustuloksia eri menetelmien välillä. Havaitsimme, että osa algoritmeista oli työn tarkoitukseen aivan liian hitaita. Ehdokkaiksi jäi käytännössä vain QSlim [GH97], JADE ja Memoryless Simplification. Valitettavasti Metro-työkalusta löytyy vain SGI binäärit, joten Metroa ei voitu käyttää tässä työssä.

Menetelmäksi valittiin QSlim v1.0 -työkaluun, koska siihen oli löydettävissä helposti julkisesti jaettava lähdekoodi. QSlim oli mittaustuloksissa osoittautunut nopeaksi ja tuotti muihin verrattuna hyvin kilpailukykyisiä tuloksia. Järjestelmän liittäminen VVV-ohjelmistoon arvioitiin myös työn aikataulun kannalta kaikkein sopivimmaksi. QSlim-järjestelmän huonoja puolia on se, että mallin lokaali topologia saattaa muuttua.

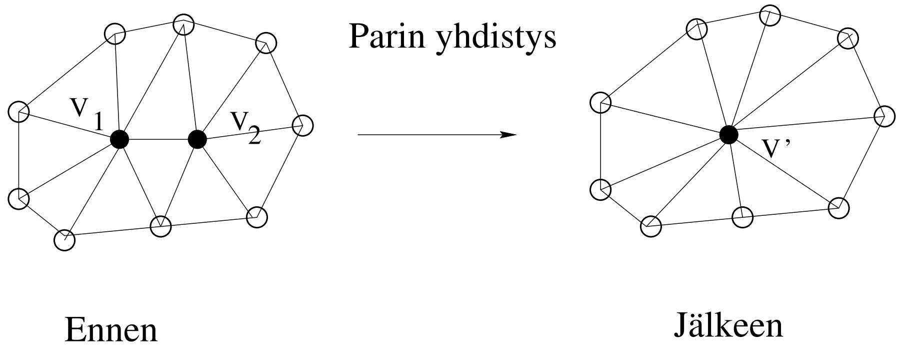
4 QSlim-menetelmä

Lopullinen toteutus perustuu QSlim v2.0 -pakkauksen MixKit v1.0 -kirjastoon, joka ilmestyi 17.3.1999 eli kesken toteutuksen. Uusi versio osoittautui helpoksi sisällyttää VVV-järjestelmään, nopeaksi ja selvästi vähemmän muistia vieväksi kuin ensimmäinen versio.

QSlim on kaariyksinkertaistamismenetelmä, joka perustuu iteratiiviseen solmujen yhdistämiseen. *Parin yhdistys (pair contraction)* $(\mathbf{v_1}, \mathbf{v_2}) \rightarrow \overline{\mathbf{v}}$ siirtää solmut $\mathbf{v_1}$ ja $\mathbf{v_2}$ uuteen paikkaan

$\overline{\mathbf{v}}$, yhdistää kaikki naapurisolmut solmuun $\mathbf{v_1}$ ja poistaa solmun $\mathbf{v_2}$.

Pari $(\mathbf{v_1}, \mathbf{v_2})$ voidaan yhdistää jos $(\mathbf{v_1}, \mathbf{v_2})$ on kaari (Kuva 3). Edelleen kaikki epämuodostuneet solmut tai pinnat poistetaan. Yhdistämisen vaikutus on pieni ja paikallinen. Tämä määritelmä on itseasiassa aika yleinen, koska joukko solmuja voidaan yhdistää yhteen solmuun $(\mathbf{v_1}, \mathbf{v_2}, ..., \mathbf{v_k}) \rightarrow \overline{\mathbf{v}}$. QSlim algoritmi kuitenkin suorittaa yhdistämisen atomisena operaationa. Yhdistämisessä mallin topologia saattaa muuttua. Jos alkuperäisessä mallissa oli reikä, saattaa reikä umpeutua yksinkertaistetussa mallissa.



Kuva 3: Parin yhdistäminen.

Alkumallista M_n lähtien jono parien yhdistämisiä suoritetaan kunnes haluttu käyttäjän antama kolmioiden lukumäärä saavutetaan ja lopullinen yksinkertaistettu malli M_g on valmis.

Yhdistettävä pari on valittava jollain menetelmällä. Tätä varten tarvitaan yhdistämiselle hinta. Hinta on virhe, joka muodostuu yhdistämisessä. Jokaiseen solmuun liitetään symmetrinen 4x4 matriisi $\mathbf{Q}$. Matriisi kuvaa tasot, jotka leikkaavat solmussa $\mathbf{v}$. Matriisia tarvitaan virheen laskemiseen, kun suoritetaan yhdistys $(\mathbf{v_1}, \mathbf{v_2}) \rightarrow \overline{\mathbf{v}}$. Jokaiseen solmuun voidaan liittää joukko tasoja. Solmuun liitetään ne tasot, jotka voidaan piirtää solmussa leikkaavien kolmioiden kautta. Tällöin solmun virhe $\Delta(\mathbf{v})$ voidaan määritellä solmun etäisyyksien neliösummaksi tasoista:

$$\Delta(\mathbf{v}) = \Delta([\mathbf{v_x}, \mathbf{v_y}, \mathbf{v_z}, \mathbf{1}]^{\mathbf{T}}) = \sum_{\mathbf{p} \in \mathbf{planes}(\mathbf{v})} (\mathbf{p}^{\mathbf{T}}\mathbf{v})^{\mathbf{2}} \tag{1}$$

,missä $\mathbf{p} = [\mathbf{abcd}]^{\mathbf{T}}$ kuvaa tasoa, joka on määritelty lausekkeesta $ax + bx + cz + d$, missä $a^2 + b^2 + c^2 = 1$. Jättäen teknisiä yksityiskohtia väliin voidaan (1) kirjoittaa $\mathbf{Q}$ matriisia käyttäen muotoon:

$$\Delta(\mathbf{v}) = \mathbf{v^T Q v}. \tag{2}$$

Näin matriisin $\mathbf{Q}$ avulla voidaan laskea virhe $\Delta(\mathbf{v})$ jokaiselle solmulle $\mathbf{v}$. Jokaiselle yhdistämiselle $(\mathbf{v_1}, \mathbf{v_2}) \rightarrow \overline{\mathbf{v}}$ on laskettava uusi matriisi $\overline{\mathbf{Q}}$, joka kuvaa solmun $\overline{\mathbf{v}}$ virhettä. Tämä voidaan tehdä laskemalla matriisit yhteen eli $\overline{\mathbf{Q}} = (\mathbf{Q_1} + \mathbf{Q_2})$. Koko algoritmi voidaan esittää tiivistetysti seuraavasti:

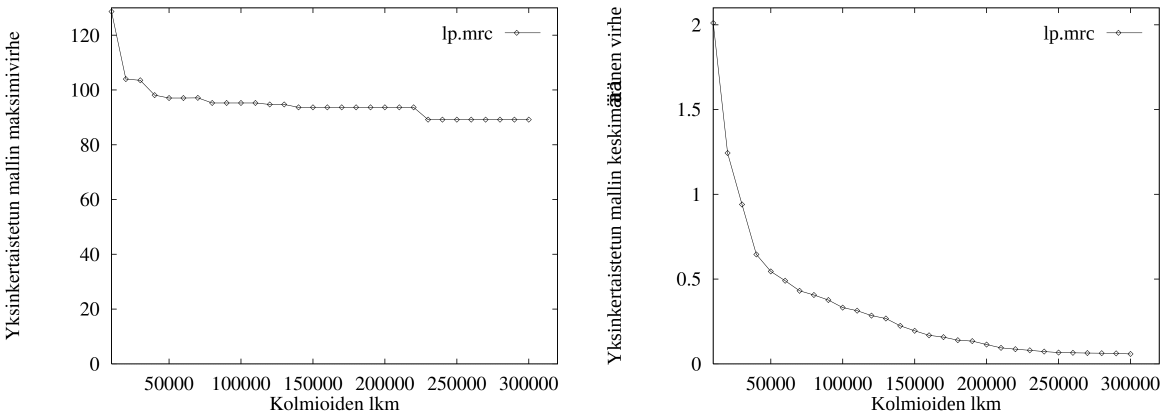
1. Lasketaan matriisi $\mathbf{Q}$ kaikille solmuille.
2. Valitaan parit, jotka voidaan yhdistää.
3. Lasketaan optimaalinen yhdistämiskohde $\overline{\mathbf{v}}$, joka minimoi $\Delta(\mathbf{v})$:ta jokaiselle kaarelle $(\mathbf{v_1}, \mathbf{v_2})$. Yhdistämiskohteen virheestä $\overline{\mathbf{v}}^{\mathbf{T}}(\mathbf{Q_1} + \mathbf{Q_2})\overline{\mathbf{v}}$ muodostuu kyseisen parin yhdistämisen hinta.
4. Sijoitetaan kaikki parit kekoon (heap) hinnan mukaan nousevaan järjestykseen.
5. Iteratiivisesti poistetaan keon päällimmäinen pari $(\mathbf{v_1}, \mathbf{v_2})$ keosta, yhdistetään tämä pari ja ylläpidetään kaikkien $\mathbf{v_1}$:n kanssa yhteydessä olevien solmujen hinnat.

5 Tulokset

QSlim-ohjelmiston tuotamien yksinkertaistettujen mallien virhettä mitattiin SlimKit paketin mukana tulleella smferror-ohjelmalla. Ohjelma lukee alkuperäisen ja yksinkertaistetun mallin SMF-tiedoston ja mittaa jokaisen solmun etäisyyttä toisen mallin pintaan. Ohjelma laskee mallien välisen maksimivirheen ja virheen keskiarvon.

Tulokset on muodostettu käyttäen kahta aitoa virusta. Käytetyt virukset ovat PRD1 (lp.mrc) ja Ross river virus (rrv_2f.mrc).

Yksinkertaistamisessa syntyvää virhettä on mitattu PRD1 viruksella. Ensimmäinen mittari on maksimivirhe (Kuva 4a), joka mittaa suurimman mallien välillä olevan virheen. Toinen käytetty mittari on virheen keskiarvo (Kuva 4b).

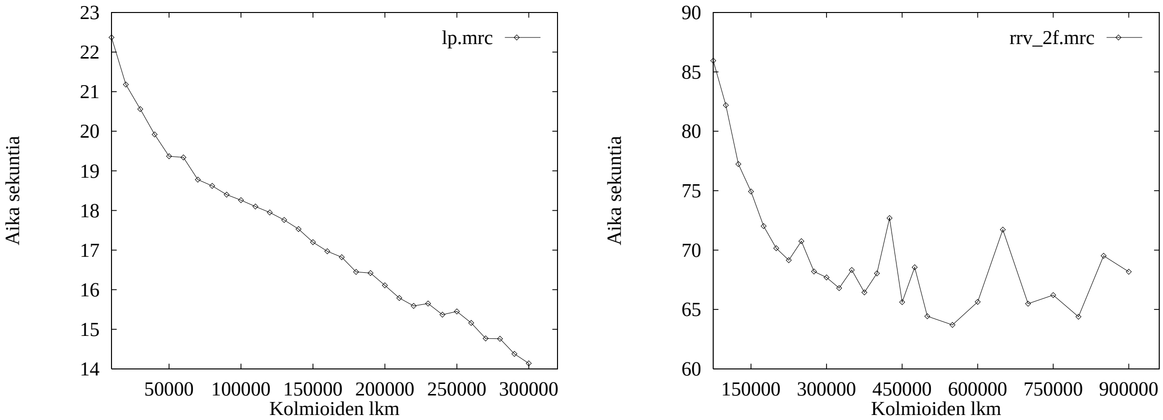


(a) Maksimivirhe

(b) Keskimääräinen virhe

Kuva 4: Alkuperäisen ja yksinkertaistetun mallin välinen virhe.

Toinen tärkeä testattava kohde on kuinka kauan mallin yksinkertaistamiseen aikaa kuluu. Testauksessa käytettiin virusmalleja PRD1 (Kuva 5a) ja Ross river virus (Kuva 5b). Suurin osa ajasta kului yksinkertaistetun mallin kirjoittamiseen levylle. Ross river mallissa on huomioitava, että tulokseen sisältyy levykirjoitus NFS:n yli, koska pöytäkoneen muisti ei riitä mallin yksinkertaistamiseen.



(a) lp.mrc

(b) rrv_2f.mrc

Kuva 5: Yksinkertaistamiseen kuluva aika.

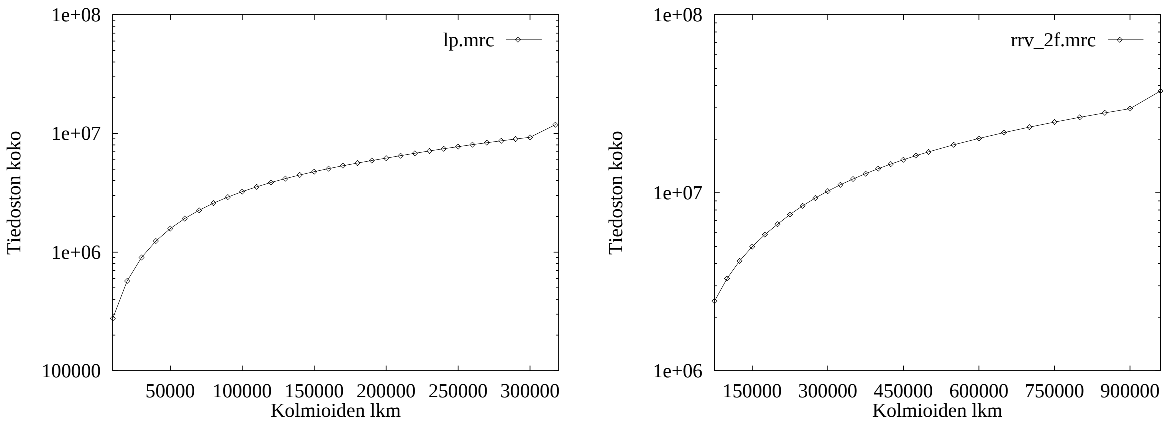
Käytetyn yksinkertaistamismenetelmän hyödyllisyyttä voidaan mitata myös käytetyn keskusmuistin kannalta. Käytettyyn keskusmuistiin vaikuttaa alkuperäisen mallin tiheyskartan koko ja valittu tiheysarvo (Taulukko 1).

Taulukko 1: Yksinkertaistamiseen kuluva keskusmuisti.

Virus	Tiheysarvo	Kolmioiden lukumäärä	Tarvittava muistimäärä
PRD1	163	317634	63 MB
Ross River	1950	960452	189 MB
Ross River	1100	1572084	ei mahdu muistiin (256 MB)
phi6_p1	160	85000	17 MB
phi6_nc	160	205200	41 MB

Kolmioiden lukumäärän raja-arvo täytyy määritellä jokaiselle virusmallille erikseen, jotta viruksen biologisesti merkittävät piirteet säilyvät. Mitään yleispätevää kolmioiden lukumäärän raja-arvoa ei pystytty määrittämään. Silmämääräisen tutkinnan mukaan PRD1 virukselle sopiva kolmioiden raja-arvo on 70000 ja phi6_p1 virukselle 10000 kolmiota.

Tehtävän tavoitteena oli luoda menetelmä, jonka avulla virusmallia voisi visualisoida esimerkiksi verkon ylitse. Tietoliikenteen kannalta on tärkeää tietää yksinkertaistetun mallin koko. Edelleen on käytetty viruksia PRD1 (Kuva 6a) ja Ross river (Kuva 6b).



(a) lp.mrc

(b) rrv_2f.mrc

Kuva 6: Yksinkertaistetun mallin koko.

6 Yhteenveto

Tehtävässä on tutustuttu kirjallisuudessa esitettyihin pintojen yksinkertaistamiseen kehitettyihin menetelmiin ja valittu perustellusti QSlim- menetelmä. Tavoitteena oli valita sovellusalueelle sopiva menetelmä, joka säilyttäisi virusmallin biologisesti merkittävät piirteet. Valittu menetelmä on toteutettu VVV-visualisointiohjelmistoon käyttäen apuna valmista toteutusta. Valitun menetelmän tuottamat tulokset on arvioitu silmämääräisen tutkinnan lisäksi käyttäen yksinkertaista mittaa. Ratkaisu on testattu oikeilla virusmalleilla, jotta nähtäisiin kuinka paljon virusmallia voidaan yksinkertaistaa hävittämättä olennaisia piirteitä.

Pintaesitystä pystytään merkittävästi yksinkertaistamaan virusmalleissa, joiden pinta on suhteellisen sileä. Runsaasti yksityiskohtia sisältäviä virusmalleja voidaan yksinkertaistaa vähemmän, ilman biologisesti merkittävien piirteiden häviämistä. Kolmioiden lukumäärän raja-arvo täytyy määritellä jokaiselle virusmallille erikseen, jotta viruksen biologisesti merkittävät piirteet säilyvät. Mitään yleispätevää kolmioiden lukumäärän raja-arvoa ei pystytty määrittämään.

Viitteet

[CCMS96] A. Ciampalini, P. Cignoni, C. Montani, , R Scopigno. Metro: Measuring Error on Simplified Surfaces. Technical report, Technical Report B4-01-01-96, Intituto I. E. I.-C. N. R., Pisa, Italy, Tammikuu 1996.

[CCMS97] A. Ciampalini, P. Cignoni, C. Montani, , R. Scopigno. Multiresolution Decimation based on Global Error. Maaliskuu 1997.

[CVM$^+$96] J. Cohen, A. Varshney, D. Manocha, G Turk, H. Weber, P. Agarwal, F. Brooks, , W. Wright. Simplification envelopes. *SIGHRAPH 96 Proc.*, sivut 119–128, 1996.

[GH97] M. Garland, P. S. Heckbert. Surface Simplification using Quadric Error Metrics. *SIGHRAPH 97 Proc.*, sivut 209–216, Elokuu 1997.

[HDD$^+$93] H. Hoppe, T. DeRose, T. Dunchamp, J. McDonald, , W. Stuetzle. Mesh Optimization. *SIGHRAPH 93 Proc.*, sivut 19–26, 1993.

[HG97] P. S. Heckbert, M. Garland. Survey of Polygonal Surface Simplification Algorithms. Technical report, School of Computer Science, Carnegie Mellon University, Toukokuu 1997.

[LE87] W. E. Lorensen, Cline H. E. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. *Computer Graphics*, 21(4):163–169, Heinäkuu 1987.

[LT98] P. Lindström, G. Turk. Fast and Memory Efficient Polygonal Simplification. *Visualization 98 Proc.*, 1998.

[SML96] W. Schroeder, K. Martin, , B. Lorensen. *The Visualization Toolkit, An Object-Oriented Approach To 3D Graphics*. Prentice Hall, 1996.

[SZL92] W. J. Schroeder, J. A. Zarge, , W. E. Lorensen. Decimation of Triangle Meshes. *Computer Graphics*, 26(2):65–70, Heinäkuu 1992.

[WW92] A. Watt, M. Watt. *Advanced Animation and Rendering Techniques, Theory and Practice*. Addison-Westley, 1992.