Varjojen laskenta – tietokonegrafiikan klassinen ongelma

Jukka Arvo
Turun Yliopisto
Informaatioteknologian laitos
Turku Centre for Computer Science
jarvo@cs.utu.fi

Tiivistelmä

Varjojen laskenta on yksi tietokonegrafiikan klassisista ongelmista ja erilaisia algoritmeja varjojen laskentaan on kehitetty jo lähes kolmenkymmenen vuoden ajan. Varjojen globaali luonne tekee niiden nopean laskemisen erityisen vaikeaksi, koska mallinnettavan maailman jokaiselle pisteelle täytyy ratkaista näkyvissä oleva valonlähteen pinta-ala. Teoreettisesti tämä tarkoittaa minimissään neliöllistä pinta-alanäkyvyysrelaatioiden määrää mallinnettavan maailman pisteiden suhteen.

Tämän artikkelin tarkoitus on esitellä varjoalgoritmiikkaa reaaliaikaisten sovellusten näkökulmasta, esitellä alueen keskeinen terminologia, tuoda esille varjoalgoritmiikan oleellisimmat ongelmat sekä antaa lukijalle käsitys reaaliaikaisen varjolaskennan uusimmista suuntauksista.

1 Johdanto

Kuvan laskemista mallintamalla valon fysikaalisia ominaisuuksia kutsutaan yleisesti *kuvasynteesiksi* (*image synthesis*). Laskentaan liittyviä algoritmeja on tutkittu laajalti tietokonegrafiikan puitteissa aina 1970-luvulta lähtien. Kuvasynteesin keskeisenä tavoitteena pidetään usein kuvatodellisuutta (*photorealism*), mutta vaikka klassinen valokuvaus tuottaakin realistisia kuvia, on sen perustana kameraoptiikka ja filmin kemialliset ominaisuudet.

Digikameroissa filmin kemiallinen prosessi on korvattu valoherkällä kuvakennolla, joka sisältää miljoonia valoherkkiä antureita. Jokainen anturi muodostaa valaistuksen mukaan signaalin, jonka digikameran ohjelmisto tallentaa muistiyksikköön. Verrattuna digikuvaukseen tietokonegrafiikassa kuvakennona toimii usein näytön pikselirasteri ja jo-

kaiseen pikseliin lasketaan siinä näkyvä valaistun maailman osa. Näin ollen valon ja pintojen fysikaalisten ominaisuuksien mallintamisesta tulee yhä keskeisempää, kun kuvienlaskenta-algoritmit kehittyvät monimutkaisemmiksi ja realistisemmiksi.

Realistinen kuvasynteesi koostuu useista osista, mutta yksi kuvasynteesin keskeisimmistä vaiheista on valaistusmallin muodostaminen. Valaistusmallit voidaan yleisesti jakaa kahteen pääryhmään: *lokaaleihin* valaistusmalleihin ja *globaaleihin* valaistusmalleihin.

Kaikille lokaaleille valaistusmalleille on yhteistä, että ne ottavat laskuissaan huomioon ainoastaan mallinnettavan pinnan sekä valonlähteiden ominaisuudet. Lokaalit valaistusmallit ovat näin ollen laskennallisesti nopeita, mutta ne eivät pysty mallintamaan esimerkiksi varjoja, kappaleiden välisiä heijastuksia tai valon etenemistä väliaineessa. Globaalit valaistusmallit pystyvät puolestaan ottamaan huomioon nämä ilmiöt, mutta aiheuttavat samalla suuria vaatimuksia laskentakapasiteetille. Esimerkiksi realististen heijastusten laskemisessa joudutaan jokaista mallinnettavan maailman pintaa käsittelemään potentiaalisena valonlähteenä.

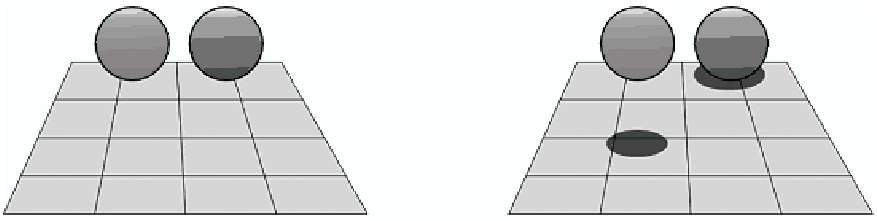
Yksi keino realistisempien kuvien nopeaan laskemiseen on erottaa valaistusmalleista varjojen laskenta omaksi kokonaisuudekseen. Käyttämällä lokaaleja valaistusmalleja yhdessä varjoalgoritmien kanssa, päästään usein askeleen verran lähemmäs tehokkaasti laskettavia realistisia kuvia. Mallinnettavien maailmojen monimutkaisuus ja käytettävissä oleva rajallinen laskentakapasiteetti luovat kuitenkin edelleen suuria haasteita yhä älykkäämpien varjoalgoritmien kehittämiselle, esimerkiksi algoritmeille, jotka osaavat jakaa laskentakapasiteettia epätasaisesti kuvan eri alueille. Kaikki kuvan alueethan eivät tarvitse yhtä tarkkoja laskelmia, koska jo pelkästään näyttölaitteiden rajallinen tarkkuus asettaa selkeitä rajoja käytettävälle laskennalle.

Myös varjojen laskenta on luonteeltaan globaali ongelma, koska niiden laskennassa täytyy jokaiselle mallinnettavan maailman pisteelle ratkaista, näkeekö se valonlähteen vai ei, ja kuinka suuri pinta-ala valonlähteestä on näkyvissä. Varjojen laskenta onkin suurelta osin pisteen ja pinnan näkyvyysrelaation nopeaa laskemista.

Ihminen pystyy usein tunnistamaan tietokoneella lasketun kuvan epärealistiseksi. Tätä aihetta on tutkittu tietokonegrafiikan puitteissa melko paljon ja lukuisia syitä kuvien epärealistisuuteen onkin löydetty. Varjojen vaikutuksen ihmisen havainnointikykyyn on todettu tutkimuksissa [7] olevan erilaisista tekijöistä kaikkein tärkein. Varjojahan on reaalimaailmassa joka puolella ja niiden puuttuminen tietokoneella lasketuista kuvista heikentää erityisesti objektien välisten etäisyyssuhteiden havainnointia. Kuva 1 on esimerkki varjojen tärkeydestä yksinkertaisessa ympäristössä.

1.1 Grafiikkakortit

Vaikka tietokoneiden laskentakapasiteetti onkin kehittynyt viime vuosina voimakkaasti, kuluu keskusyksiköltä (Central Processing Unit) usein liian paljon



Kuva 1: Varjojen merkitys objektien välisten etäisyyssuhteiden havainnoinnissa. Vasemmalla etäisyyssuhteet ovat epäselvät, kun taas oikealla kappaleiden väliset suhteet pystytään havaitsemaan nopeasti varjojen sijainnin perusteella.

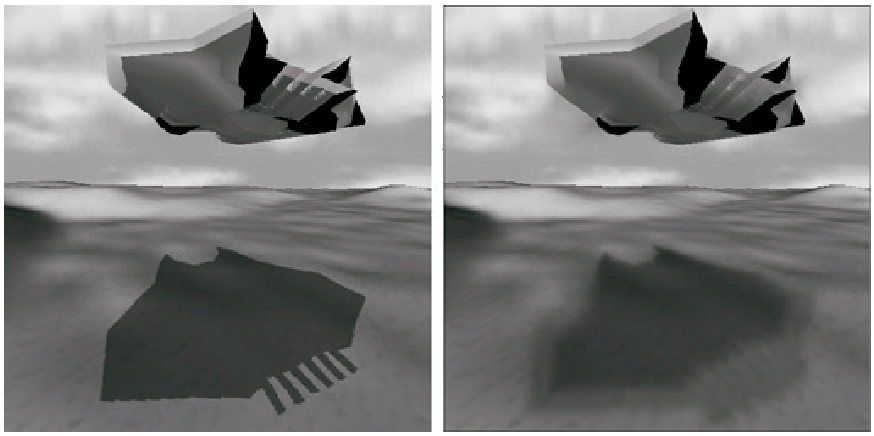
aikaa yhden realistisen kuvan laskentaan reaaliaikasovelluksissa, joissa kuvia lasketaan noin 50 kuvan sekuntivauhdilla. Dynaamista reaaliaikalaskentaa varten onkin nykyään tarjolla suorituskykyisiä tietorinnakkaiseen rakenteeseen (Single Instruction stream Multiple Data stream eli SIMD) perustuvia kuluttajatason grafiikkakortteja, jotka pystyvät projektiopiirtämään näytölle miljoonia kolmioita sekunnissa. Näiden ohjelmoitavien grafiikkakorttien suuri suorituskyky onkin ollut yksi keskeisimpiä tekijöitä tuomaan varjoalgoritmiikkaa mukaan yhä useampiin reaaliaikaisiin sovelluksiin, esimerkiksi dynaamisiin visualisointeihin ja tietokonepeleihin.

Tämän päivän uusimmilla kuluttajatason grafiikkakorteilla pystytään dynaamisia varjoja laskemaan tehokkaasti kahdella algoritmityypillä: *varjotilavuuksilla* [4] (*shadow volume*) ja *varjokartoilla* [8] (*shadow map*). Näiden kahden pääalgoritmiluokan lisäksi (Luvut 3, 4) on olemassa erilaisia hybridimenetelmiä varjojen laskemisen nopeuttamiseksi, mutta kyseisten algoritmien laitteistoläheinen toteuttaminen on vielä varsin ongelmallista. Teokset [2, 1] ovat hyviä aloituslähteitä aiheesta enemmän kiinnostuneelle lukijalle.

2 Peruskäsitteet

Reaalimaailmassa valonlähteillä on aina olemassa fysikaalinen pinta-ala. Tietokonegrafiikassa laskennan yksinkertaistamiseksi käytetään kuitenkin usein pistemäistä valonlähdettä. Pistemäisen valonlähteen laskennallinen nopeus perustuu varjojen binääriluonteeseen, koska tutkittava piste voi olla vain kokonaan varjossa (*umbra*) tai valossa. Pistemäiset valonlähteet tuottavat näin ollen vain teräväreunaisia eli *kovia* varjoja (kuva 2).

Kovia varjoja ei esiinny pinta-alan omaavilla valonlähteillä. Tällöin mallinnettavan maailman osat voidaan jakaa valonlähteen suhteen kolmeen ryhmään. Kokonaan valossa, kokonaan varjossa sekä *puolivarjossa* (*penumbra*) oleviin osiin. Puolivarjoalueilla mallinnettavan maail-



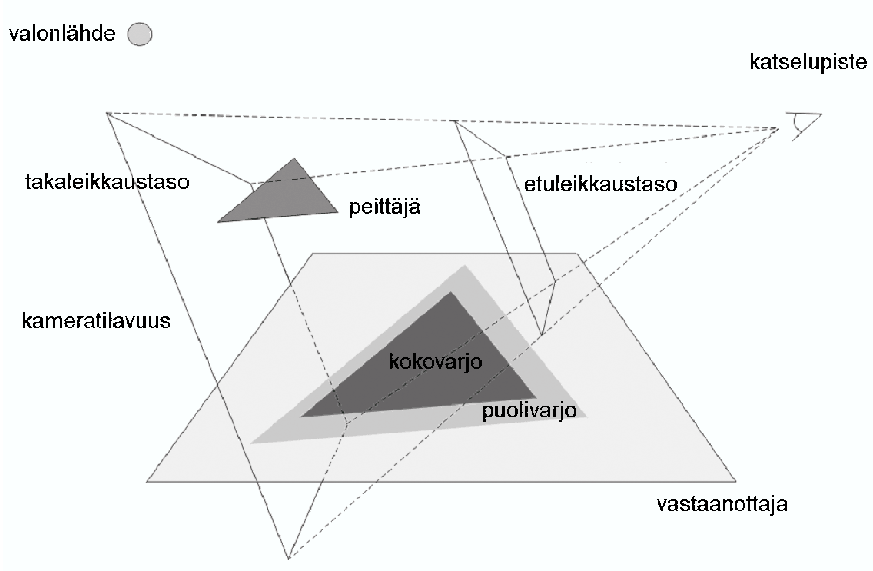
Kuva 2: *Vasen:* Kuvan varjot ovat kovia. Kovat varjot korostavat merkittävästi objektien välisiä spatiaalisia suhteita. *Oikea:* Kuvan varjot ovat pehmeitä, jolloin valonlähteen, varjoa heittävän kappaleen ja varjoa vastaanottavan kappaleen etäisyyssuhteet havaitaan helposti varjoreunan pehmeydestä.

man piste näkee valonlähteen vain osittain, mikä synnyttää pehmeän siirtymän kokovarjosta täysin valaistuun osaan.

Varjo-algoritmiikassa varjoa heittävää kappaletta kutsutaan *peittäjäksi* (*occluder*), joka sulkee näkyvyyden valonlähteen ja tutkittavan pisteen välillä. Varjon vastaanottavaa kappaletta kutsutaan *vastaanottajaksi* (*receiver*). Avaruuden osaa, josta laskettava kuva muodostetaan, kutsutaan *kameratilavuudeksi* (*view frustum*). Kameratilavuus muodostuu leikatusta pyramidista, jonka määrittävät etuleikkaustaso ja takaleikkaustaso yhdessä katselupisteen kanssa. Valokuvauksessa etuleikkaustaso vastaa kameran linssin pintaa takaleikkaustason ollessa äärettömän kaukana. Tietokonegrafiikassa etuleikkaustaso joudutaan kuitenkin siirtämään kauemmas ja takaleikkaustaso lähemmäs katselupisteestä, koska muuten kuvassa näkyviä pintoja ei pystytä määrittämään oikein rajallisen laskentatarkkuuden vuoksi. Kuva 3 havainnollistaa edellä esiteltyjä peruskäsitteitä.

3 Varjotilavuus-algoritmi

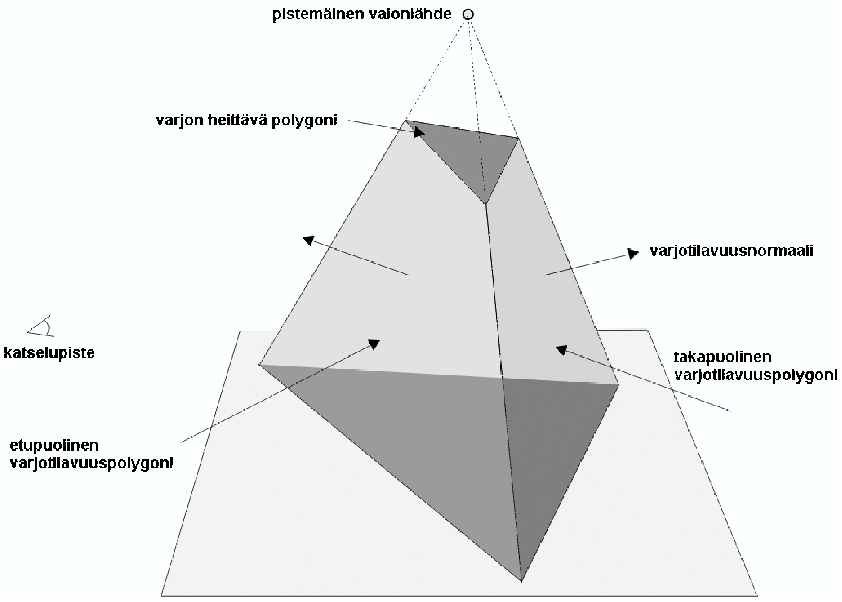
Reaaliaikaisessa tietokonegrafiikassa mallinnettava maailma esitetään usein kolmioverkkoina, jossa maailman objektien pinnat koostuvat kulmapisteistä, niitä yhdistävistä särmistä ja särmien rajaamista pinnoista. Yleinen tapa kolmion esittämi-



Kuva 3: Kuva havainnollistaa varjoalgoritmiikkaan tiiviisti liittyviä käsitteitä. Katselupiste yhdessä etu- ja takaleikkaustasojen kanssa muodostaa kameratilavuuden, jonka sisällä olevasta geometriasta lopullinen kuva lasketaan. Pinta-alan omaava valonlähde ei näy kokonaan kaikkialla vastaanottavalla pinnalla, joten peittäjäkappale heittää koko- ja puolivarjosta muodostuvan pehmeän varjon.

seen on käyttää kolmea indeksiä erikseen esitettyihin kulmapisteisiin, sillä muuten vierekkäiset kolmiot tarvitsevat kopioita samoista kulmapisteistä. Kolmiota käytetään paljon kappalemallinnuksen perusprimitiivinä, koska kolme pistettä määrittää tason mielivaltaisen muunnoksen jälkeen, mikäli pisteet vain kuvautuvat eri pisteiksi. Näin ollen projektiopiirtämisessä käytettävät piirtorutiinit ovat yksinkertaisia ja niiden laitteistoläheinen toteutus on helppoa.

Tunnetun varjotilavuusalgoritmin [4] toiminta perustuu ylimääräiseen varjogeometrian luontiin valonlähteestä näkyvien peittäjien ääriviivojen suhteen. Algoritmi olettaa mallinnettavan maailman kolmioverkkojen olevan suljettuja ja siis jokaisen särmän kuuluvan kahdelle kolmiolle. Aluksi peittäjien ääriviivat haetaan tutkimalla mitkä särmät kuuluvat ääriviivalle. Se tapahtuu laskemalla kaksi pistetuloa särmän päätepisteestä valonlähteeseen osoittavan vektorin ja särmän omistavien kolmioiden normaalien välillä. Pistetulojen merkkien erotesssa toisistaan on



Kuva 4: Varjotilavuuspolygonit luodaan pistemäisen valonlähteen ja varjoa heittävän objektin ääriviivan mukaan. Syntyneet varjopolygonit luokitellaan katselupisteen mukaan *eteenpäin* ja *taaksepäin suuntautuviksi*.

kyseinen särmä ääriviivalla, koska toisen kolmion normaali muodostaa alle 90° ja toisen kolmion normaali yli 90° kulman tutkittavasta pisteestä valonlähteeseen osoittavan vektorin suhteen.

Tämän jälkeen jokaisesta kolmioverkon ääriviivan särmästä luodaan nelikulmainen varjopolygoni, joka alkaa ääriviivan särmästä ja jatkuu äärettömyyteen valonlähteestä poispäin. Luoduista varjopolygoneista muodostuu tällä tavalla *varjotilavuus*, jonka sisällä olevat maailman pisteet ovat varjossa (katso kuva 4).

Varjotilavuusalgoritmin keskeisin vaihe on määrittää kaikki kameratilavuuden ja varjotilavuuksien sisällä olevat maailman pisteet. Nämä pisteet voidaan määrittää laskemalla mallinnettavan maailman ja varjopolygonien välillä boolean-operaatioita. Näiden tuloksena saadaan joukko tilavuuksia, joiden sisällä olevat pisteet ovat varjossa. Tämä on kuitenkin laskennallisesti raskas ja ongelmallinen menetelmä varjotilavuuksien rajaamien alueiden laskentaan. Tästä johtuen varjotilavuusalgoritmi toteutetaankin te-

hokkaasti grafiikkakorteilla käyttämällä *sapluunapuskuria* [6] (*stencil buffer*).

Sapluunapuskurin käyttö perustuu kameratilavuudesta rasteroidun *syvyyskartan* (*z-buffer*) hyödyntämiseen. Syvyyskartta kertoo kuinka kaukana kuvatason pikseleissä näkyvät pinnat ovat katselupisteestä, eli jokaisen pikselin etäisyyden lähimpään kohteeseen näkösuoraa pitkin. Varjoalueet määritetään rasteroimalla kaikki varjopolygonit ja suorittamalla vähennys- ja lisäysoperaatioita sapluunapuskuriin. Katsojaan päin suuntautuvien polygonien kohdalla puskurin arvoa kasvatetaan yhdellä ja poispäin suuntautuvien kohdalla vähennetään yhdellä, mikäli varjopolygonin maalattava pikseli on katselupisteen ja lähimmän syvyyskartassa näkyvän pinnan välissä. Lopputuloksena puskurissa on positiivinen luku varjopikseleiden kohdalla ja nolla valaistujen pikseleiden kohdalla.

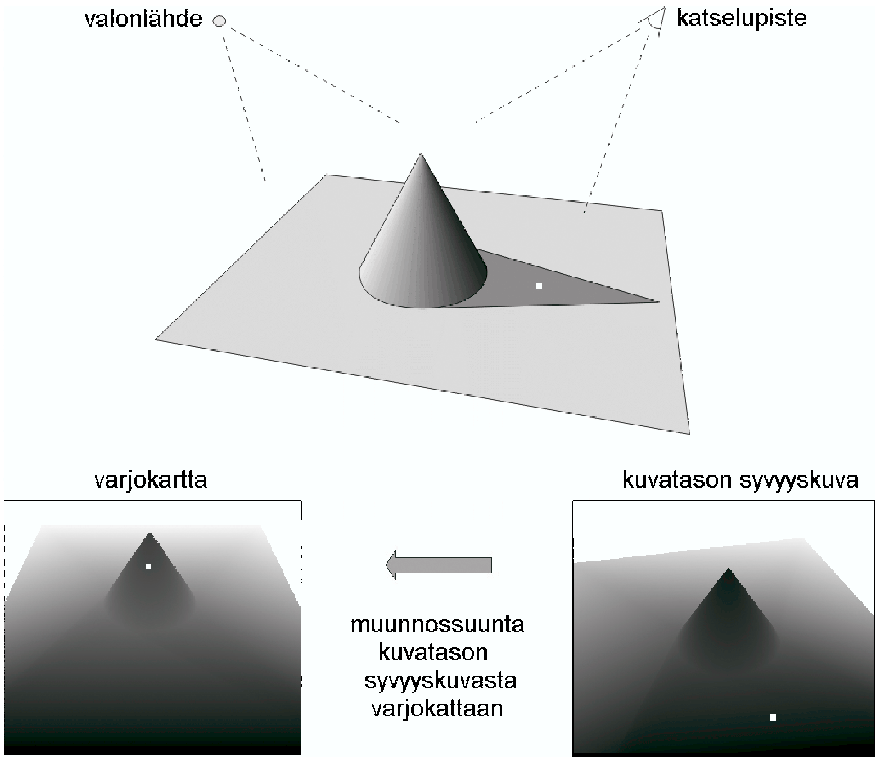
Varjotilavuusalgoritmien tuottamat varjorajat ovat aina korkealaatuisia, koska ne pysyvät analyyttisinä aina kameratilavuuden piirtämiseen saakka. Korkealla laadulla on kuitenkin hintansa, sillä vaikka nykyiset grafiikkakortit pystyvätkin käsittelemään suuria määriä polygoneja sekunnissa, tulee niiden rajallinen *maalauskapasiteetti* (*fill rate*) nopeasti vastaan suurien varjopolygonien kanssa. Varjotilavuusalgoritmin suurin haittapuoli onkin sen kuluttama suuri maalauskapasiteetti, koska pienenkin varjoalueen määrittäminen saattaa vaatia suurien sapluunapuskurialueiden maalauksia. Näin ollen yksi lähitulevaisuuden keskeisin varjotilavuusalgoritmiikan tutkimuskohde onkin turhien varjopolygonien tehokas karsiminen ja mahdollinen yksinkertaistaminen.

4 Varjokartta-algoritmi

Toisen, dynaamisissa reaaliaikasovelluksissa yleisesti käytetyn varjoalgoritmin, varjokartta-algoritmin [8] toiminta perustuu pisteen näkyvyysongelman diskretisointiin. Algoritmi luo ensin diskreetin syvyyskuvan valonlähteestä, johon tallennetaan jokaisen kuvatason pikselin esittämän pinnan etäisyys valonlähteestä. Tätä valonlähteestä katsottavaa syvyyskuvaa kutsutaan *varjokartaksi*.

Tämän jälkeen kamerasta käsin luodaan vastaavanlainen syvyyskuva ja jokainen syvyyskuvaan tallentuva piste muunnetaan vuorollaan valonlähteen koordinaatistoon. Muunnetun pisteen paikan mukaan luetaan varjokarttaan talletettu syvyysarvo ja sitä verrataan muunnetun pisteen etäisyyteen valonlähteestä. Etäisyyksien ollessa yhtä suuria, näkevät valonlähde ja kamera saman pisteen, minkä perusteella tutkittava piste on valossa. Muussa tapauksessa tutkittava piste on varjossa, koska valonlähteen näkemä pinta on lähempänä kuin tutkittava piste. Kuva 5 havainnollistaa varjokartta-algoritmin syvyyskuvia ja pistemuunnoksia eri koordinaatistojen välillä.

Varjokartta-algoritmin suurin etu varjotilavuusalgoritmiin verrattuna on varjojen laskennan epäsuora riippuvuus mallinnettavan maailman monimutkaisuudes-



Kuva 5: Varjokartta-algoritmi rasteroi ensin syvyyskuvan valonlähteestä ja kamerasta. Tämän jälkeen jokainen kameran pikseli transformoidaan valonlähteen koordinaatistoon ja vastaavan valonlähteen pisteen etäisyyttä verrataan kameran pisteen etäisyyteen. Mikäli etäisyys on yhtä suuri, on piste valossa, muulloin piste on varjossa. Muunnoksien havainnollistamiseksi valonlähteen ja kameran syvyyskuviin on merkitty valkoisella alue, joka sijaitsee yleiskuvan varjoalueella.

ta, koska varjojen laskenta perustuu diskreetteihin näkyvyysesityksiin valonlähteestä ja kamerasta. Menetelmän käyttökelpoisuutta rajoittavat kuitenkin diskretisoinnista aiheutuvat ns. aliasointivirheet, jotka näkyvät käytännössä epätarkkuuksina varjorajoilla. Erityisen voimakasta aliasointia esiintyy tilanteissa, joissa katsoja on selkeästi lähempänä valaistavaa kappaletta kuin valonlähde. Tällöin varjokartan yhteen pikseliin osuu monta kameran transformoitua pikseliä, mikä aiheut-

taa varjorajojen sahalaitaisuutta.

Lisäksi etäisyysvertailussa joudutaan käyttämään erilaisia kynnysarvoja muunnoksista ja syvyyspuskurin diskretisoinnista aiheutuvien epätarkkuuksien vuoksi. Nämä virheet korostuvat entisestään, mikäli valonlähteestä muodostetussa varjokartassa on käytetty perspektiiviprojektiota, joka pienentää kauempana olevia objekteja syvyysvaikutuksen aikaansaamiseksi. Tämän seurauksena varjokartan syvyysarvot epälinearisoituvat, koska muuten tasot eivät säilyisi tasoina projektiivisessa avaruudessa.

Vaikka varjokartta-algoritmi onkin varsin tehokas ja yleinen varjojen laskenta-algoritmi, on sen parissa edelleen muutamia keskeisiä ratkaisemattomia ongelmia. Yksi tällaisista ongelmista on oikean kynnysarvon valitseminen syvyystestauksen epätarkkuuksien vähentämiseksi. Diskretisointihan aiheuttaa aina virheitä syvyystestaukseen, vaikka sama maailman pinta näkyisikin valonlähteestä ja kamerasta. Vakioarvoisen kynnyksen käyttäminen syvyystestauksen virheiden kumoamiseksi on ongelmallista, koska verrattavat näytepisteet eivät yleensä kuvaa täysin samaa maailman pistettä. Näytepisteiden syvyyserot vaihtelevat pinnan sijainnin ja asennon mukaan valonlähteestä ja kamerasta katsottuna.

Varjokartta-algoritmi on kuitenkin helppo toteuttaa uusimmilla kuluttajatason grafiikkakorteilla, koska sen toiminnassa on paljon yhtäläisyyksiä katselupisteen kuvan tuottamiselle. Käytettävissä olevien tarkkuuksien puitteissa varjokartta-algoritmi soveltuu tänä päivänä etupäässä paikallisten valonlähteiden varjojen mallintamiseen.

5 Pehmeät varjot

Muutaman lähivuoden aikana tulemme todennäköisesti näkemään yhä enemmän dynaamisia varjoja reaaliaikaisissa sovelluksissa, kuten tietokonepeleissä ja visualisoinneissa. Tästä johtuen reaaliaikaisten varjoalgoritmien ongelmien ratkaisemiseksi tullaan varmasti tekemään yhä voimakkaammin töitä ympäri maailman niin akateemisissa kuin yritysmaailmankin tutkimusyksiköissä.

Pitkällä aikavälillä varjoalgoritmiikan tutkimuksen painopiste tulee siirtymään enemmän reaaliaikakäyttöön soveltuvien pehmeiden varjoalgoritmien pariin. Tämä tulee kuitenkin olemaan erittäin haastavaa, sillä realististen pehmeiden varjojen laskeminen vaatii pinta-alanäkyvyysongelmien ratkaisuja, ja nämä ongelmat ovat tunnetusti erittäin vaikeita ratkaista reaaliajassa [5]. Näin ollen yhden valonlähteestä otetun näytteenottopisteen mukaan generoitujen pehmeiden varjojen reunat eivät tule olemaan täysin realistisia, mutta pienen pinta-alan omaavilla valonlähteillä tämä likimääräistys voi kuitenkin tuottaa uskottavia pehmeitä varjorajoja. Yleisesti ottaen objektin ääriviiva muuttuu valonlähteen jokaisen näytteenottopaikan mukaan, joten nopeiden ääriviiva-algoritmien kehittäminen tulee entistäkin tärkeämmäksi.

Yksinkertaisimmillaan pehmeitä varjoja pystytään laskemaan käyttämällä kovia varjoalgoritmeja useaan kertaan lukui-

sista eri pinta-alavalonlähteen näytteenottopaikoista. Tällaiset usean näytteenoton algoritmit ovat käytännössä kuitenkin tehottomia, sillä realistiset pehmeät varjot saattavat tarvita helposti satoja näytteenottoja. Jo pelkästään pehmeän varjoalueen haluttu sävyjen määrä asettaa minimitarpeen näytteenottopaikoille, koska yksi näytteenottopaikka voi tuottaa ainoastaan yhden lisäsävyn pehmeään varjoalueeseen. Kuva 6 havainnollistaa suurten näytteenottomäärien tarpeellisuutta mallinnettaessa pehmeitä varjoja usealla näytteellä.

Yksi lupaava pehmeitä varjoja laskeva algoritmi esiteltiin hiljattain alan johtavassa konferenssissa [3]. Algoritmin toiminta perustuu varjotilavuusalgoritmiin, mutta kovien varjotilavuuksien lisäksi algoritmi luo pehmeille varjoille omat varjotilavuudet, joita käytetään pehmentämään kovia varjoreunoja. Algoritmin vakaus pohjautuu kameratilavuuteen luotuun pehmeään varjogeometriaan, joka säilyttää tarkkuuden aina rasterointiin saakka eikä diskretoinnista aiheituvia virheitä juurikaan ilmene.

Algoritmi on myös mahdollista toteuttaa kuluttajatason grafiikkakorteilla, mutta korttien tämänhetkinen laskentateho ei valitettavasti riitä kovinkaan monimutkaisille maailmoille. Algoritmin laskenta on raskasta, koska jokaisen ääriviivan pehmeä varjotilavuus joudutaan käsittelemään erikseen. Lisäksi algoritmi laskee pehmeiden varjojen reunat vain yhden ääriviivapisteen suhteen, joten tämä rajoittaa käytettävien valonlähteiden pinta-aloja. Pienillä valonlähteen pinta-aloilla likimääräistys on uskottava, mutta suuremmilla valonlähteillä likimääräistys ei enää toimi sujuvasti.

6 Yhteenveto

Esitellyn varjotilavuusalgoritmin kiistaton etu on varjorajojen analyyttisyys, kun taas varjokartta-algoritmi saattaa helposti tuottaa epätarkkoja varjorajoja. Toisaalta varjotilavuusalgoritmin suoritusaika kasvaa voimakkaasti valonlähteen vaikutuspiirin sisällä olevien kolmioiden määrän ja niistä generoitavien varjopolygonien koon suhteen. Varjokartta-algoritmin varjojen laskemisaika kasvaa lineaarisesti käytettävän tarkkuuden suhteen, mutta tarvittavien karttakuvien luontikustannus saattaa vaihdella, joskin yleensä varsin maltillisesti.

Kokonaisuudessaan varjokartta-algoritmi on yleisempi ja usein tehokkaampi tapa luoda varjoja, kun taas varjotilavuusalgoritmi tuottaa parempilaatuisia varjorajoja rajoitetummalle geometrialle. Lähitulevaisuudessa tulemme todennäköisesti näkemään erilaisia uusia reaaliaikasovelluksiin sopivia hybridimenetelmiä tarkkojen ja laskennallisesti nopeiden kovien varjojen tuottamiseen. Pehmeiden reaaliaikaisten varjojen yleistyminen dynaamisissa sovelluksissa tulee viemään vielä aikaa, koska pinta-alaongelmien nopea ratkaiseminen ei vielä onnistu.

Kuva 6: *Vasen:* Pienipinta-alainen valonlähde, jolla mallinnetaan pehmeitä varjoja usealla näytteenotolla. Kuvan aikaansaamiseksi vaadittiin 128 näytteenottoa. Mikäli varjoilla olisi suurempi pehmeä reuna, kasvaisi tarvittava näytteenottomäärä huomattavasti. *Oikea:* Suuremman pinta-alan omaava valonlähde, jolla mallinnetaan pehmeitä varjoja usealla näytteenotolla. Kuvan aikaansaamiseksi vaadittiin 1024 näytteenottoa. Kuva oikeassa alareunassa on pala pehmeää varjoa suurennettuna.

Kiitokset

Kiitokset Professori Antti Valmarille ja Professori Jukka Teuholalle tätä tekstiä koskevista asiantuntevista huomautuksista.

Viitteet

[1] Tomas Akenine-Möller and Eric Haines. *Real-Time Rendering, 2nd edition.* A.K. Peters Ltd., 2002.

[2] Jukka Arvo. *Real-Time Shadow Algorithms for Dynamic Environments*. Master's Thesis, University of Turku, December 2001.

[3] Ulf Assarsson and Thomas Akenine-Möller. A Geometry-based Soft Shadow Volume Algorithm using Graphics Hardware. *ACM Transactions of Graphics (Proceedings of ACM SIGGRAPH)* , vol. 22, no. 3, pp. 511–520, July 2003.

[4] Franklin Crow. Shadow Algorithms for Computer Graphics. In *Proceedings of SIGGRAPH '77*, ACM Press, pages 242–248, July, 1977.

[5] Fredo Durand. *Visibility: Analytical Study and Applications*. PhD thesis, Université Joseph Fourier, Grenoble, France, July 1999.

[6] Tim Heidmann. Real Shadows Real Time. *Iris Universe*, vol. 18, pages 18–31, November, 1991.

[7] Leonard C. Wanger, James A. Ferwerda, Donald P. Greenberg. Perceiving Spatial Relationships in Computer-Generated Images. *IEEE Computer Graphics and Applications*, volume 12, number 3, pages 44–58, May 1992.

[8] Lance Williams. Casting Curved Shadows on Curved Surfaces. In *Proceedings of SIGGRAPH '78*, ACM Press, pages 270–274, August, 1978.