Harri Juutilainen

Ositusmetodit reaaliaikaisessa renderöinnissä

Tietotekniikan kandidaatintutkielma

28. huhtikuuta 2017

Jyväskylän yliopisto

Tietotekniikka

Tekijä: Harri Juutilainen

Yhteystiedot: haeejuut@student.jyu.fi

Työn nimi: Ositusmetodit reaaliaikaisessa renderöinnissä

Title in English: Subdivision methods in real-time rendering

Työ: Kandidaatintutkielma

Sivumäärä: 23+0

Tiivistelmä: Tietokonegrafiikka ja siihen liittyvät menetelmät ovat nopeasti kehittyvä tutkimuksen ala. Tämän tutkimuksen tavoitteena on pohtia eri ositusmenetelmien toimintaa reaaliaikaisessa käytössä sekä vertailla näitä menetelmiä. Nykyaikaiselle käytölle tehokkaimmaksi osoittautui kolmesta tutkitusta menetelmästä tuorein (Brainerd ym. 2016). Tulevassa tutkimuksessa laitteistointegraatioon keskittyminen voisi tuoda ositusmenetelmät nopeammin kuluttajien saataville.

Avainsanat: osa-aluepinnat, tietokonegrafiikka, renderöinti

Abstract: Computer graphics and its methods are a rapidly developing field in research. The purpose of this study is to discuss and compare the functionality of several subdivision methods in real-time use. The latest of the three studied methods turned out to be the most optimal for modern use (Brainerd ym. 2016). Focusing future research on hardware integration could bring subdivision methods for consumers sooner.

Keywords: subdivision surfaces, computer graphics, rendering

Kuviot

Sisältö

1 Johdanto

"With an increasing demand for richer images with more and more visual detail, it is desirable to render such movie-quality assets in real time, enabling the use of subdivision surfaces in both content creation tools and interactive video games."

Brainerd ym. 2016

Tutkimuksen aiheena on tietokonegrafiikan piirtämiseen käytettyjen menetelmien tutkiminen. Tavoitteena on tutkia, miten grafiikan piirtäminen raskaampien mutta mallintamiseen paremmin soveltuvien ”elokuvatasoisten” metodien avulla voitaisiin siirtää reaaliaikaisiin sekä interaktiivisiin sovelluksiin, kuten peleihin ja 3D-mallintamisohjelmiin. Varsinkin animaatioalalla piirtäjän ja animoijan on tärkeää nähdä valmiiksi renderöity tulos reaaliaikaisesti eikä vain rautalankamalleina. Koska *osa-aluepinnat* (subdivision surface) ovat jo vuosia olleet elokuva- ja animaatioalan standardina 3D-sisällön kuvaamiseen (Schäfer ym. 2015), keskitytään tutkielmassa osa-aluepintojen arvioinnissa käytettävään rekursiiviseen algoritmiin (Catmull ja Clark 1978) ja siitä kehitettyihin menetelmiin.

Tutkimuksessa pohditaan miten CGI (computer-generated imagery) ja elokuvateollisuudessa käytettävät teknologiat saadaan toimimaan tehokkaasti ja hyvälaatuisena interaktiivisissa sovelluksissa, kuten peleissä. Tavoitteena on myös miettiä mitä rajoitteita nousee pinnalle, sekä sovelluksissa että laitteistoissa, kun näitä metodeja käytetään reaaliaikaisesti.

Tutkimusstrategiana on kuvaileva ja vertaileva kirjallisuuskatsaus, jonka tarkoituksena on aihepiiriin liittyvän tiedon esittely, arviointi ja tulkinta. Toteutus tapahtuu syventymällä artikkeleihin ja aihepiiriin liittyviin käsitteisiin, minkä jälkeen pohditaan niiden suhteita olemassaoleviin teorioihin ja malleihin. Tutkimuksessa käytetään hyödyksi peli- ja animaatioteollisuuden suurten yritysten, kuten Valven (Drone ym. 2010), Pixarin (Halstead, Kass ja DeRose 1993) ja Nvidian (Brainerd ym. 2016) avoimesti julkaisemia tutkimustuloksia. Tässä tutkimuksessa vertaillaan erilaisia graafisen ohjelmoinnin metodeja tehokkuuden ja käytännön hyödyn kannalta.

Luvussa kaksi avataan yleisiä termejä ja prosesseja, joita grafiikan piirtämisessä käytetään

ja joiden ymmärtämistä tarvitaan myöhemmissä luvuissa. Kolmannessa luvussa esitellään kolme erilaista ositusmenetelmää. Ensimmäisenä tutustutaan ”Feature-adaptive subdivision” (FAS) –renderöintimetodiin (Nießner ym. 2012) ja selitetään sen toiminta ja käyttötarkoitukset. Seuraavaksi esitellään FAS:n seuraava iteraatio ”Dynamic feature-adaptive subdivision” (DFAS) (Schäfer ym. 2015), sekä kerrotaan sen toimintaperiaatteet ja miten se paransi edeltäjästään. Viimeisenä käsitellään kolmannen, kaikista uusimman, metodin ”subdivision using adaptive quadtrees” (Brainerd ym. 2016) toiminta. Neljännessä luvussa vertaillaan edellämainittuja metodeja renderöintinopeuden ja tehokkuuden näkökulmasta, sekä pohditaan tekniikoiden käytännön sovellutuksia. Lopuksi tehdään yhteenveto ja pohditaan, mihin tutkimus alalla voisi seuraavaksi suunnata.

2 Tietokonegrafiikan käsitteet ja osa-aluepinnat

Tässä luvussa käsitellään tutkimuksessa käytettyjä tietokonegrafiikan käsitteitä sekä käydään tarkemmin läpi osa-aluepintojen kehitys ja niiden määritelmä.

2.1 Käsitteet

Ehkä yleisin grafiikan käsitteistä on *verteksi* (vertex, vertices). Verteksi on tietorakenne, joka kuvaa pistettä avaruudessa ja tyypillisesti sillä on X-,Y- ja Z-koordinaatit. Kun grafiikkaa renderöidään, eli kuvaa generoidaan koneellisesti, verteksien kuvaamat pisteet esitetään käyttäjälle graafisesti, esimerkiksi näytöllä. Verteksissä voi olla tallessa myös ylimääräistä renderöintiin tarvittavaa dataa, kuten väri tai normaalivektori. Verteksejä käytetään yleisesti muodostamaan *primitiivejä*, joilla tarkoitetaan yksinkertaisia muotoja kuten kolmioita. Näitä primitiivejä yhdistämällä saadaan aikaiseksi *verkko* (mesh, triangle mesh). Verkkoja käytetään, koska muistin säästämiseksi on optimaalisempaa käyttää yhdistettyjä kolmioita ennemmin kuin montaa yksittäistä.

3D-mallia mallintaessa luodaan yleensä aluksi matalaresoluutioinen ja mahdollisimman yksinkertainen *pohjaverkko* (base mesh), jota voi jälkeenpäin tarkentaa ja muokata. 3D-mallin pinta voi myös olla jaettu *tilkkuihin* (patch), jotka mahdollistavat monimutkaisempien pintojen kuvaamisen. Mallin pinnassa olevilla erillisillä tilkuilla voi olla erilaisia ominaisuuksia: esimerkiksi tekstuuri voi olla vierekkäisillä tilkuilla erilainen.

Grafiikan piirtämiseen käytetään näytönohjaimia, joiden ohjelmoimiseen tarvitaan sovellusohjelmarajapintoja, kuten OpenGL tai DirectX (Microsoft Windows Dev Center 2017). Edellämainituista rajapinnoista on koottu korkean tason yleiskuvat, joissa kuvataan renderöinnin vaiheet. Näitä abstrahointeja kutsutaan graafisen prosessoinnin putkistoiksi tai renderöintiputkistoiksi (graphics processing pipeline, rendering pipeline).

Näissä putkistoissa on yhtenä vaiheena tessellaattori, joka mahdollistaa verteksien ja kolmioiden dynaamisen generoimisen. Tesselloinnin tarkoituksena on siis jakaa saatu objekti pienemmiksi objekteiksi (yleensä kolmioiksi, pisteiksi tai suoriksi), mikä tekee siitä tarkem-

man tai sileämmän näköisen. Jatkossa yhtä graafisen prosessoinnin putkiston läpikäyntiä sanotaan *GPU-ohitukseksi* (GPU pass).

2.2 Osa-aluepinnat

Osa-aluepintoja käytetään tietokonegrafiikassa kuvaamaan sileitä pintoja. Tämä saadaan aikaiseksi osittamalla karkeampaa pohjaverkkoa. Tässä alaluvussa käydään läpi lyhyesti osa-aluepintojen kehitys ja nykytila sekä niiden hyödyt. Lopuksi avataan Catmull–Clark -osituksen algoritmi.

G. Chaikinin (1974) ideasta muodostaa käyrä monikulmiosta lisäämällä verteksejä ja kulmia (Halstead, Kass ja DeRose 1993) yleistettiin myöhemmin pintoihin toimivia metodeja. Näitä eri tavoin osa-aluepintoja kuvaavia metodeja kehittivät mm. Catmull ja Clark (1978) ja Doo ja Sabin (1978) sekä Loop (1987). Tässä tutkimuksessa keskitytään Catmull–Clark -osa-aluepintoihin ja niiden menetelmiin (Catmull ja Clark 1978). Catmull–Clark -pinnat ovat yleisesti elokuva-alalla käytetyimpiä osa-aluepintoja. Niiden ominaisuutena ovat muun muassa pinnan tahkoihin merkityt topologiat (Bolz ja Schröder 2002).

Brainerd ym. 2016 mukaan osa-aluepinnat ovat nousseet elokuva-alalla standardiksi — varsinkin hahmojen kuvaamisessa. Animaatiostudio Pixar, joka on avoimesti tukenut sekä kehittänyt osa-aluepintojen menetelmiä eteenpäin, on standardoinut ominaisuuksia[1], joihin osa-aluepinnoissa tulisi pyrkiä. Vuonna 2005 Edwin Catmull, Tony DeRose ja Jos Stam voittivat Oscar-palkinnon teknisestä saavutuksesta ja elokuvateollisuuden edistämisestä. Tämän palkinnon he ansaitsivat keksimästään osa-aluepintojen sovelluksesta ja käytöstä.

Osa-aluepinnat kehitettiin ratkaisemaan ongelmia, kuten pinnan tilkkujen väliin jääviä rakoja. Nießner ym. (2012) määrittelevät niiden hyödyiksi seuraavat:

- monimutkaisempien mallien tekeminen yhden pinnan avulla.
- laskoksien (crease) sekä terävien ja puoliterävien kulmien luominen.
- pinnan tessellointi dynaamisesti GPU:n avulla ja vain karkean verkon animointi.

Näiden ominaisuuksien avulla saadaan tehtyä monimutkaisempia pintoja vähemmällä data-

1. https://renderman.pixar.com/products/rispec/

määrällä.

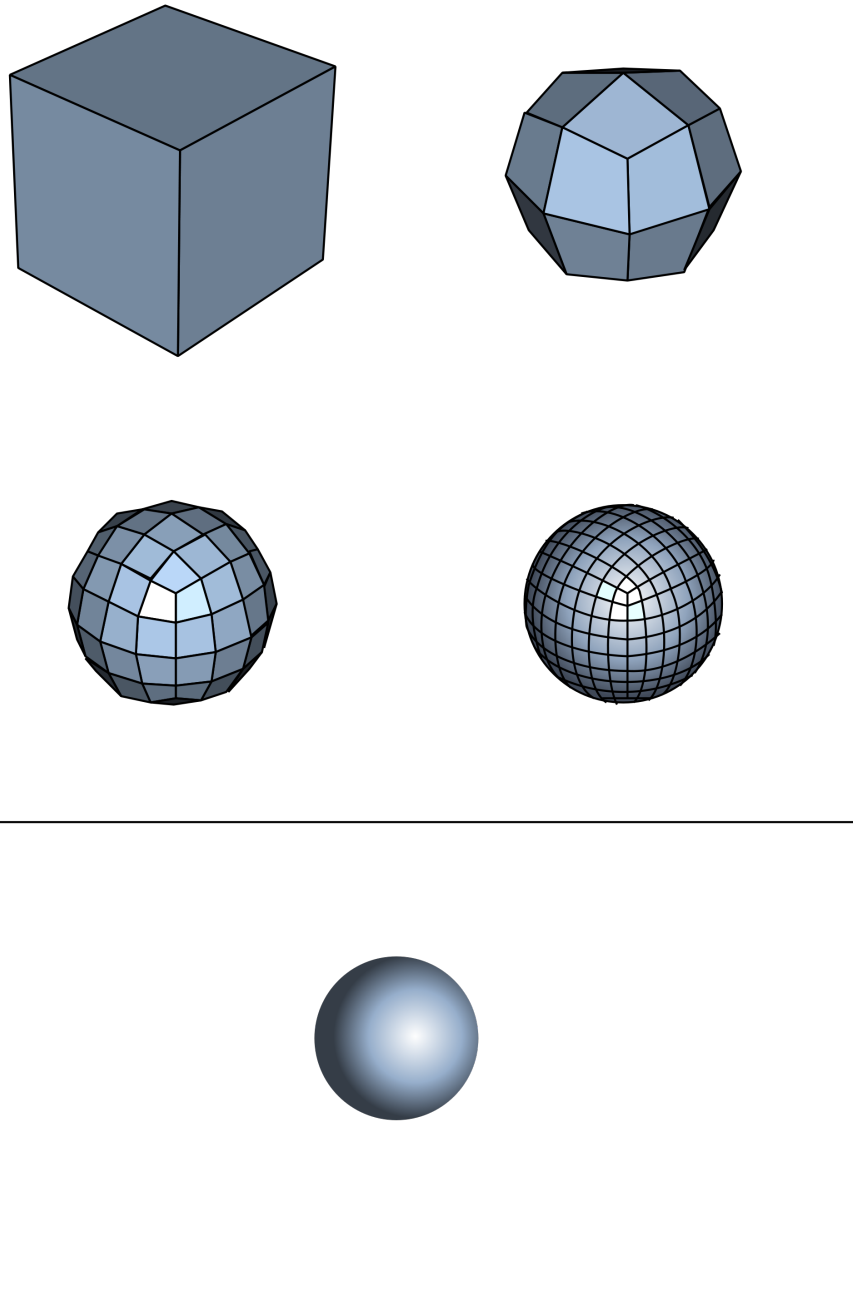
Kaikkia vastaavia ominaisuuksia ei esimerkiksi aiemmin alalla käytetyllä NURBS[2]-mallilla saanut tehtyä. Renderöinti- ja mallinnusalalla käytetty (ja vieläkin käytössä oleva) NURBS-malli perustuu B-käyristä[3] rakennettuun pintaan, kun taas osa-aluepinnoissa käytetään monikulmioita. NURBS-mallilla renderöidessä ilmeni tunnetusti paljon ongelmia — varsinkin reaaliaikaisesti käytettäessä saattoi kuvatun pinnan tilkkujen väliin tulla rakoja ja epämuodostumia. Osa-aluepintojen huono puoli NURBS-malliin verrattuna on kuitenkin se, että mallintaminen ja tekstuurin asettaminen kuvattuun malliin on monimutkaisempaa.

Elokuva-alalla yleisimmin hahmojen mallintamisessa käytetty Catmull ja Clark (1978) kehittämä ositus toimii jakamalla pintatilkku (surface patch) neljään alatilkkuun rekursiivisesti niin kauan, että tuloksena saatava tilkku on renderöitävän elementin pikselin (tai muun halutun arvon) kokoinen. Nießner ym. (2012) mukaan Catmull–Clark -pinnat ovat erityisesti nousseet reaaliaikaisten sovellusten käyttöön laitteistotessellaation kehityksen myötä. Catmull ja Clark (1978) kuvailevat metodiaan seuraavilla säännöillä:

- Jokaiselle tahkolle asetetaan uusi tahkopiste, joka on kaikkien vanhojen tahkopisteiden keskiarvo.
- Kulmien uusiksi pisteiksi asetetaan vanhojen kulmapisteiden ja kahden uuden tahkopisteen (jotka jakavat kulman) keskiarvo.
- Uusien verteksien pisteet saadaan kaavalla: $\frac{Q}{n} + \frac{2R}{n} + \frac{S(n-3)}{n}$, jossa
  - Q = kaikkien vanhan verteksipisteen vieressä olevien tahkopisteiden keskiarvo.
  - R = kaikkien vanhasta verteksipisteestä seuraavien vanhojen reunojen keskipisteiden keskiarvo.
  - S = vanha verteksipiste.
- Uudet verteksipisteet yhdistetään uusiin kulmapisteisiin ja näin saadaan uudet tahkot.

Osituksen tasoa sanotaan tämän ylläolevan algoritmin rekursion määräksi, joka siis määrää objektin ”sileyden”. Kuviossa 1 on näkyvillä kuution muotoiseen pohjaverkkoon tehty ositus kolmannelle osituksen tasolle asti.

2. Non-Uniform Rational Basis Spline
3. B-käyrä (B-spline) on sileä, kontrollipisteiden avulla muodostettu suora.



Kuvio 1. Catmull–Clark -osa-aluepinta (Wikipedia 2006)

3 Ositusmetodit

Tässä luvussa käydään läpi kolme metodia osa-aluepintojen renderöimiseen. Metodit on esitetty julkaisuvuoden mukaisessa järjestyksessä.

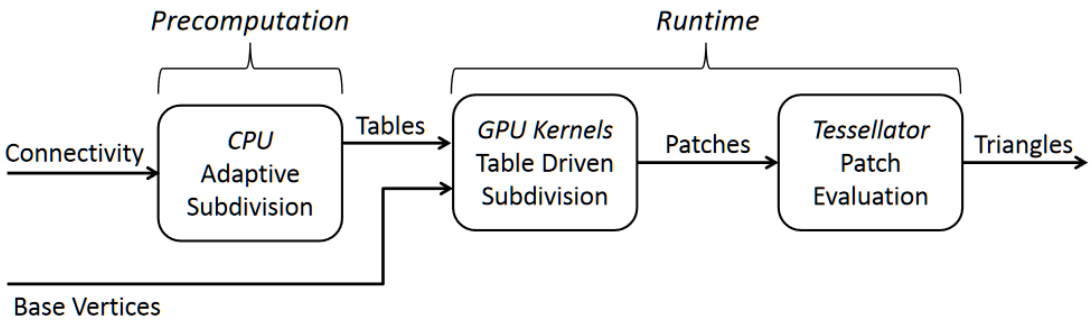
3.1 Feature-adaptive subdivision

Ominaisuuksiltaan mukautuva ositus (lyhennetty FAS) on algoritmi GPU-pohjaiseen Catmull–Clark -osa-aluepintojen renderöintiin. FAS on ensimmäinen algoritmi, joka renderöi Catmull–Clark -osa-aluepintoja reaaliaikaisesti toteuttaen *puoliterävän laskoksen* (semi-sharp crease) ja *hierarkkisen yksityiskohtaisuuden tasot* (HLOD, Hierarchical Level of Detail). Näistä ensimmäinen vähentää huomattavasti laskoksissa tarvittavien verteksien, kulmien ja pintojen määrää vapauttaen muistia muuhun käyttöön. HLOD taas antaa nimensä mukaisesti mahdollisuuden rajata yksityiskohtaisuuden tasoa eri objekteille, joten silläkin voidaan parantaa suorituskykyä. FAS:ää on käytetty pohjana esimerkiksi Pixarin avoimen lähdekoodin OpenSubDiv[1]- projektissa.

FAS:n toiminta voidaan jakaa kahteen vaiheeseen: esikäsittelyvaiheeseen, jonka suorittaa CPU, sekä GPU:n käsittelemään ajonaikaiseen komponenttiin. Algoritmille tulee syötteenä verkko, jossa on verteksejä ja pintoja, sekä mahdollisesti ylimääräisenä puoliteräviksi laskoksiksi merkittyjä kulmia sekä yksityiskohtien hierarkkista dataa. Kuviossa 2 näkyy yleisellä tasolla FAS:n toiminta. Tätä avataan seuraavaksi lisää.

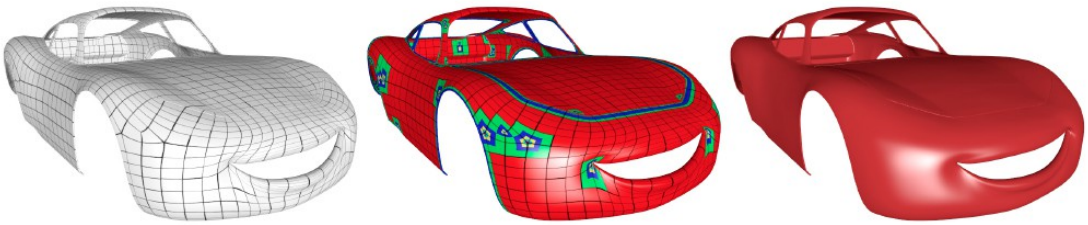
Esikäsittelyvaiheessa syötedatan perusteella lasketaan tauluja, jotka ohjaavat osituksen toimintaa. Jokaista ositustasoa vastaan lasketaan oma taulu. Esikäsittelyvaiheen jälkeen tuleva GPU:n tekemä mukautuva ositus toimii siten, että jokaisella ositustasolla tunnistetaan säännölliset ja epäsäännölliset[2] verteksit. Ositus tehdään vain epäsäännöllisten verteksien viereisille alueille. Kun ositus on tehty, käytetään laitteistotessellaattoria tuottamaan joka ositustasolle kahta erilaista tilkkua: täys- (full patch) ja siirtymätilkkua (transition patch). Täystilkut jakavat reunan saman ositustason tilkkujen kanssa, kun taas siirtymätilkut eri ositustason tilk-

1. http://graphics.pixar.com/opensubdiv/docs/intro.html
2. Kulman, laskoksen tai muun vieressä oleva



Kuvio 2. Vuokaavio FAS:n toiminnasta (Nießner ym. 2012, s. 23)

kujen kanssa (maksimisiirtymänä kuitenkin yksi). Tämä mahdollistaa sen, että ajon aikana voidaan säätää erilaisille tilkuille sopiva laitteistotessellaation määrä ja renderöityyn kuvaan saadaan haluttu määrä yksityiskohtia. Kuviossa 3 on näkyvissä syötteenä saatu pohjaverkko sekä FAS:n määrittämät eri ositustasojen tilkut.



Kuvio 3. Pohjaverkko (vas.), osituksen määrittävät tilkut (keskellä), renderöity malli (oik.) (Nießner ym. 2012, s. 2) ©Disney/Pixar

Nießner ym. (2012) esittivät algoritmia elokuvatasoisilla 3D-malleilla ja se pystyi generoimaan noin miljardi kolmiota sekunnissa. Metodi toteuttaa myös Pixarin asettamat RenderMan-ominaisuudet Catmull–Clark -osa-aluepinnoille. FAS pääsee suorituskyvyssä samalle tasolle Loop ym. (2009) tehokkaan Gregory-tilkuille perustuvan arviointimallin kanssa. FAS on tosin eksakti eikä arviointiin perustuva metodi.

Schäfer ym. (2015) mukaan rajoituksena FAS-metodissa on kuitenkin se, että se rajoittaa epäsäännöllisten verteksien ympärillä olevat alueet yhdelle ositustasolle. Tästä johtuen lähellä olevien kohteiden vieressä oleville kaukana oleville alueille voi tulla suuri määrä yli-

määräisiä tilkkuja, joita ylitesselloidaan niihin haluttuun yksityiskohtiin nähden. Tämä käyttää turhaan muistia.

3.2 Dynamic feature-adaptive subdivision

Dynaaminen ominaisuuksiin mukautuva ositus (lyhennetään DFAS) on iteraatio luvun 3.1 FAS:n toimintaan. Schäfer ym. (2015) kehittämä metodi vähentää tarvittavien tilkkujen määrää mahdollistamalla itsenäiset osituksen tasot jokaiselle epäsäännölliselle verteksille. Tässä alaluvussa käydään läpi DFAS:n toiminta ja miten se eroaa aikaisemmasta FAS:stä.

Samalla periaatteella kuin FAS-metodissa, DFAS-algoritmissa luodaan ensin tauluja. Toisin kuin FAS:ssä taulut ovat nyt staattisten sijaan dynaamisia. Tämä tarkoittaa sitä, että jokainen epäsäännöllinen verteksi voi valita sopivan ositustason ajon aikana. Ajon aikana GPU laskee jokaisen piirretyn framen kohdalla tessellaation tiheyden tilkuille, joissa on epäsäännöllisiä verteksejä. Seuraavaksi tehdään dynaamisesti määritelty ositus GPU:n laskemille lokaaleille alueille sen mukaan mitä käyttäjä on määrittänyt yksityiskohtaisuuden tasoksi. Lopuksi kaikki näin saadut tilkut viedään renderöitäväksi. DFAS käsittelee tilkut siis samanaikaisesti vaikka niillä olisi eri ositustasot.

DFAS eroaa edeltäjästään antamalla epäsäännöllisille vertekseille mahdollisuuden valita tarvitattavan osituksen tason. Schäfer ym. (2015) mukaan DFAS myös vähentää tilkkuja laskevien puskureiden ylijäämää. Tämä vähentää renderöintikutsujen määrää FAS:ään verrattuna.

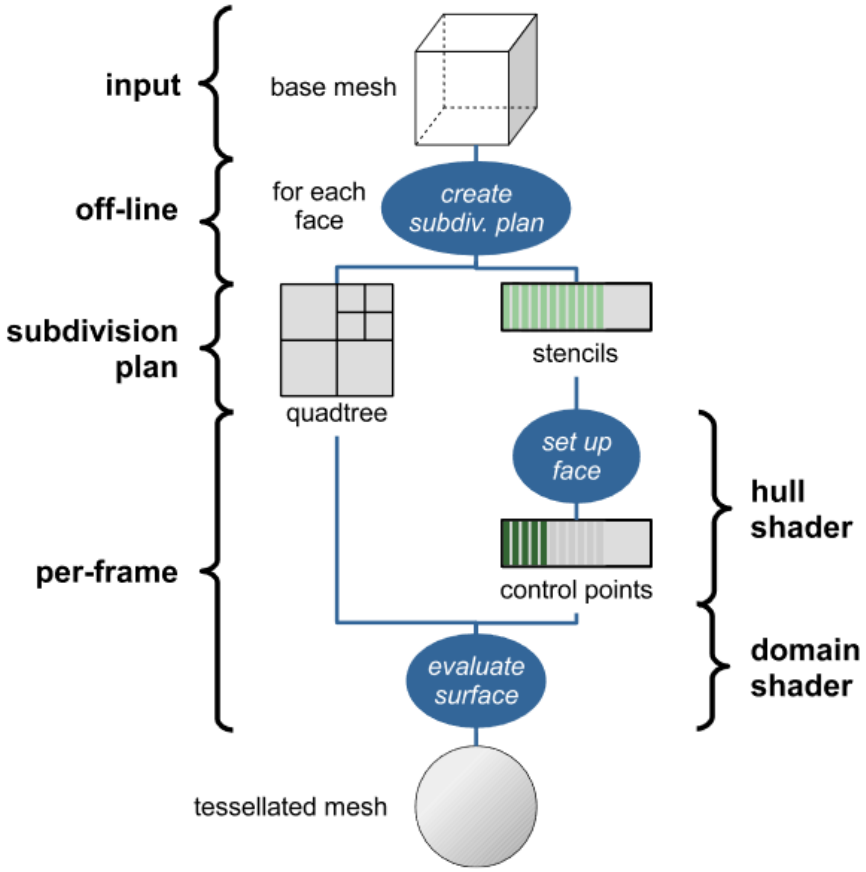
3.3 Subdivision using adaptive quadtrees

Tässä alaluvussa käydään läpi Brainerd ym. (2016) kehittämä ositusmetodi, joka on tutkittavista metodeista uusin. Metodista käytetään jatkossa lyhennettä SAQ (Subdivision using Adaptive Quadtrees). SAQ käyttää nimensä mukaisesti hyödykseen mukautuvia nelipuita. Luvun alussa käydään läpi, mitä nelipuut ovat ja mitä ne tuovat metodiin. Sen jälkeen seuraa yleiskuva SAQ:n toiminnasta ja algoritmin tarkempi läpikäynti.

Nelipuu (quadtree) on puurakenne, jonka lapsilla voi olla tasan neljä lapsisolmua. Nelipuun tyyppi määritellään sen kuvaaman datan tai sen muodon perusteella. Yleisimmät tyypit ovat

pistenelipuu ja aluenelipuu. Samet ja Webber (1985) mukaan näiden kahden yhdistelmä on kuitenkin kaikkein tehokkain monikulmioiden tallentamiseen interaktiivisissa ympäristöissä. Nelipuita käytetään yleensä jakamalla alue neljään osaan, joten ne sopivat hyvin rekursioihin.

SAQ:n algoritmin rakenne on seuraava: pohjaverkolle tehdään ositussuunnitelma (subdivision plan) ja ajon aikana jokaisen piirretyn framen kohdalla tehdään pinnan arviointi ja ositus tämän suunnitelman mukaan. Kuviossa 4 kuvataan yleisellä tasolla SAQ:n toimintaa. Tätä rakennetta avataan seuraavaksi lisää.



Kuvio 4. Vuokaavio SAQ:n toiminnasta (Brainerd ym. 2016, s. 3)

SAQ:lle tulee syötteenä Catmull–Clark -pohjaverkko, jossa on merkitty ominaisuutena epäsäännölliset verteksit ja reunat. Syötteenä saadulle verkolle tehdään ositussuunnitelma eli samantapainen mukautuva ositus tahkoille kuin luvussa 3.1. Osituksesta saadut osatahkot laitetaan nelipuurakenteeseen. Tässä rakenteessa olevat lehtisolmut voidaan arvioida ja asettaa kategorioihin. Jos ilmenee, että nelipuun lehtisolmussa on halutunlainen osatahko, tehdään

siitä sapluuna. Tätä sapluunaa voidaan käyttää tulevia tahkoja läpikäydessä huomattavasti nopeuttamaan prosessia. Tämä prosessi tehdään syötteenä saadun verkon jokaiselle tahkolle. Kategorioita ja sapluunoita tehdessä voidaan hyödyntää monia muistinkäyttöä parantavia keinoja, kuten esimerkiksi luvun 3.1 toteuttama puoliterävä laskos voidaan esittää laskoksena, jossa on merkittynä terävyys. Tämä vähentää selkeästi saadun nelipuun kokoa ja näin ollen nopeuttaa suoritusta.

Seuraavana vaiheena on tahkojen järjestely ja itse ositus. Pohjaverkosta viedään yksitellen tahkoja tessellaattorille. Ulostulona tessellaattorilta tulee verteksejä, joilla on sijainti verrattuna kyseiseen tahkoon. Sen jälkeen renderöintiputkistossa (2.1) toimiva varjostaja laskee tarvittavan tesselloinnin määrän ja tarvittavien kontrollipisteiden sijainnin. Ositus tapahtuu siis tässä vaiheessa varjostajalta saatujen kontrollipisteiden ja ositussuunnitelmasta saatujen sapluunoiden avulla.

Viimeisenä vaiheena on pinnan arviointi (surface evaluation), jossa käydään alussa tehty nelipuu läpi iteratiivisesti (huom. ei rekursiivisesti). Lehtisolmuista etsitään osa-tahkoja, jotka vastaavat edellisessä vaiheessa laskettujen verteksien sijainteja. Jokaiselle näin löydetylle lehtisolmulle on määritetty oma sääntönsä. Näiden sääntöjen avulla muokataan sekä arvioidaan uudestaan osituksen tarvetta. Tärkeimpänä näistä säännöistä on tavallisten (ei rajalla tai laskoksia) tahkojen sääntö, jossa Brainerd ym. (2016) mukaan oli tehokkaampaa renderöidä tahkot B-käyristä muodostetulla tilkulla.

Brainerd ym. (2016, s. 2) pitää SAQ:n tärkeimpinä ominaisuuksina seuraavia:

- Metodi pystyy renderöimään yhdellä GPU:n ohituksella osa-aluepintoja, jotka toteuttavat muun muassa puoliterävät laskokset
- Tesselloitavaksi viedään vain yksi, monesta tahkosta kerätty primitiivi, joka nopeuttaa renderöintiä
- Selkeästi nopeampi kuin nykyiset huippurenderöijät

Rajoitteena menetelmässä on Brainerd ym. (2016) havaintojen mukaan se, että ositussuunnitelmaa tehdessä pitää asettaa kiinteä maksimitaso ositukselle. Tämä johtaa siihen, että metodi joutuu rajatapauksissa turvautumaan arvioituihin arvoihin. Menetelmä soveltuu tällä hetkellä vain Catmull–Clark -osa-aluepinnoille, mutta on Brainerd ym. (2016) mukaan yleistettävissä

muillekin osa-aluepinnoille.

4 Metodien vertailu ja käytännön sovellukset

Tässä luvussa vertaillaan luvussa 3 esiteltyjä osa-aluepintojen menetelmiä. Vertailussa käytetään hyödyksi niiden kehittäjien tekemiä tilastoja niiden suorituskyvystä. Vertailun jälkeen pohditaan olemassaolevia käytännön sovelluksia, jotka hyödyntävät kyseisiä menetelmiä, sekä ehdotetaan mahdollisia uusia sovelluksia.

4.1 Vertailu

Kaikki vertailut on tehty Windows- käyttöjärjestelmillä ja kuluttajille saatavilla olevilla näytönohjaimilla. FAS testattiin NVIDIA GeForce GTX 480 -näytönohjaimella, kun taas DFAS ja SAQ testattiin NVIDIA GeForce GTX 980:llä. FAS ja DFAS käyttivät Direct3D 11 -rajapintaa, kun taas SAQ käytti OpenGL -rajapintaa. Yksi suurimpia vaikuttajia menetelmien tehokkuuteen ja nopeuteen on niiden tekemien GPU-ohitusten määrä sekä muistinhallinnan laatu. Menetelmien vertailussa tärkeimpänä arvona nähdään tässä tutkimuksessa renderöintiaika, joka vaikuttaa eniten reaaliaikaisiin sovelluksiin.

Metodeista julkaisuvuodeltaan aikaisinta, luvun 3.1 FAS:ää, verrataan menetelmään nimeltä *globaali verkon jalostus* (global mesh refinement). Globaali verkon jalostus on aikaisemmin ositukseen käytetty malli, joka oli muistivaatimuksiltaan hyvin raskas. Globaalissa verkon jalostuksessa kaikille renderöitävän kuvan malleille tehdään yhden tason ositus yhden GPU-ohituksen aikana. Näitä ohituksia on siis tarve tehdä useampi, jos on haluttua saada suurempi osituksen taso.

FAS osoittautui globaalin verkon jalostusta nopeammaksi, kun ositustasona on vähintään kaksi. Menetelmää verrattiin Nießner ym. (2012) omaan globaaliin jalostusmenetelmään sekä Shiue, Jones ja Peters (2005) kehittämään menetelmään. Nießner ym. (2012) mukaan laitteistotessellaatiota hyödyntämällä tilkkujen laskemiseen vältetään muistinhallinnan pullonkaulat ja saavutetaan nopeampi renderöintiaika.

Luvun 3.2 DFAS pystyi FAS:ään verrattuna nopeuttamaan selkeästi ositus- ja renderöintiaikaa. Tämä saatiin Schäfer ym. (2015) mukaan aikaiseksi vähentämällä FAS:n tekemää

turhaa tessellointia. Siitä huolimatta, että tesselloinnin optimoinnin aikaansaamat lokaalisti mukautuvat alueet veivät laskentatehoa, DFAS oli silti jopa puolet nopeampi kuin edeltäjänsä. Kuten luvussa 3.2 mainittiin, DFAS tekee myös vähemmän renderöintikutsuja kuin FAS. Vertailussa käytetyillä malleilla oli eri määrä pohjatilkkuja, mikä ei vaikuttanut DFAS:n tehokkuuteen. Renderöidyt alueet olivat myös tasaisemmin tesselloitu kuin FAS:llä (Schäfer ym. 2015)[s. 7]. Tasaisempi tessellointi tarkoittaa tässä tapauksessa laadukkaampaa lopputulosta.

Brainerd ym. (2016) SAQ:ta verrattiin Loop ym. (2009) Gregory-tilkkujen menetelmään sekä FAS-, DFAS- ja Pixarin OpenSubDiv- menetelmiin. Vertailussa on otettu huomioon, että eri menetelmät toteuttavat eri ominaisuuksia, kuten esimerkiksi laskoksia. Rajaamme tässä tutkimuksessa SAQ:n vertailun luvuissa 3.1 ja 3.2 esitettyihin metodeihin.

Luvun 3.3 SAQ eroaa muista metodeista kahdella merkittävällä tavalla. Ensimmäisenä erona se, että tessellaatiolaitteistolle viedään monen tahkon sijaan vain yksi primitiivi. Tämä yksinkertaistaa menetelmää huomattavasti. Toisekseen SAQ:ssa jokainen tahko prosessoidaan erikseen GPU:n renderöintiputkistoon integroituna.

Brainerd ym. (2016) tekemien testien mukaan ”SAQ on 1.2 - 3.4 kertaa nopeampi kuin muut eksaktit menetelmät (FAS ja DFAS), kun tessellointikerroin on vähintään 3”. Tessellointikertoimella tarkoitetaan tässä osituksen määrää (Microsoft Windows Dev Center 2017). Kaikista suurin hyöty SAQ:ssa on kuitenkin Brainerd ym. (2016) mukaan epäsäännöllisten tahkojen käsittelyssä, jossa se oli yli kolme kertaa nopeampi kuin FAS. Vertailu näiden kahden välille tehtiin mallilla, jossa oli pelkästään epätavallisia tahkoja. Vain säännöllisten tahkojen prosessoinnissa SAQ on FAS:n kanssa yhtä tehokas.

4.2 Käytännön sovellukset

Kuten luvussa 2.2 mainittiin, osa-aluepinnat ovat jo elokuva-alalla käytössä mallinnuksessa, animoinnissa ja renderöinnissä. Osa-aluepintojen menetelmiä nähtiin ensi kertaa valkokankaalla Pixarin lyhytelokuvassa ”Geri’s game” (1997). Elokuvassa päähahmon kasvot oli renderöity ja animoitu käyttäen osa-aluepintoja. Ositusmenetelmiä voi nähdä myös yleisimmissä mallinnus- tai animaatiosovelluksissa (esimerkiksi 3DMax, Blender ja Maya). Osa-

aluepintojen reaaliaikaisuus nousee tärkeäksi varsinkin animaatiossa, jossa animoijan on tärkeä nähdä malli renderöitynä ja pystyä silti tekemään siihen muutoksia.

Toinen tärkeä ala osa-aluepinnoille on alati kasvava peliteollisuus. Nießner ym. (2012), Schäfer ym. (2015) ja Brainerd ym. (2016) tavoitteena olikin tehdä ositusalgoritmeistaan sellaisia, että ne toimisivat kuluttajille avoimilla laitteistoilla ja olisivat osana seuraavan sukupolven pelejä. Valven Source[1]-pelimoottorissa käytössä oleva Hammer-kenttäeditori käyttää jo nyt ositusmenetelmiä. Hammerissa ositusta käytetään muun muassa kenttien muotojen muokkaamiseen. Brainerd ym. (2016) toteutti luvun 3.3 SAQ-metodilla osituksen reaaliaikaisesti renderöityyn kohtaukseen joka ajettiin modernilla pelimoottorilla.

Brainerd ym. (2016) mukaan osa-aluepintojen renderöinti saattaa olla laitteiston kehittyessä yhtä helppoa kuin kolmioista muodostettujen verkkojen renderöinti on nykyään. Schäfer ym. (2015) uskookin, että DFAS:n kyky asettaa tessellointitiheys tilkulle riippumatta siitä, onko se säännöllinen vai epäsäännöllinen, mahdollistaa tulevaisuuden laitteiston tukevan natiivia osa-aluepintojen renderöintiä. Natiivi laitteistotuki toisi varmasti uusia ulottuvuuksia osa-aluepintojen tehokkuudelle ja käyttötavoille.

1. https://developer.valvesoftware.com/

5 Yhteenveto

Osa-aluepintojen menetelmät ovat selkeästi kehittymässä oleva grafiikan osa-alue. Tässä tutkimuksessa esitetiin osa-aluepintoja ja niiden nykytilaa, sekä kolmea erilaista metodia ositukseen. Tutkimuksessa myös pohdittiin niiden eroja renderöintitehon ja eri ominaisuuksien kannalta. Elokuvatasoisten, monimutkaisempien mallien renderöinnissä tehokkaimmaksi osoittautui Brainerd ym. (2016) kehittämä nelipuita hyödyntävä ositusmetodi. Muilla esitetyillä metodeilla on kuitenkin omat vahvuutensa ja elokuva-alalla edelleen käytetyin metodi onkin varmasti ”Feature Adaptive Subdivision” (Nießner ym. 2012).

Osa-aluepintojen suurimpana etuna on se, että ne on järkevästi yleistetty monille kohteille. Monipuolisuus ja joustavuus on varsinkin grafiikan alalla tärkeää, koska renderöinnin kohteita ja erilaisia malleja on yhtä paljon kuin tekijöitä. Shiue, Jones ja Peters 2005 mukaan ositusmenetelmien tulevaisuudessa voisi näkyä varsinkin optimointi ja rinnakkaisajon hyödyntäminen. Myös eri menetelmien yhdistely siten, että eri tilanteissa käytetään niihin sopivaa menetelmää voisi olla mahdollista. Näin voisi yhdistää esimerkiksi eksaktien menetelmien ja arviointimenetelmien hyötyjä.

Suuri osa Catmull–Clark -pinnoille tehdyistä menetelmistä yrittää parantaa suorituskykyä käyttämällä tehokkaampia tietorakenteita ja algoritmeja. CPU:ta käytetään lähinnä tukemaan renderöintiä esimerkiksi taulukkolaskennalla ja suurin osa laskentatyöstä annetaankin näytönohjaimelle. Tätä puutetta koetetaan ratkaista muun muassa Bolz ja Schröder (2002) tutkimuksessa, jossa pyritään optimoimaan CPU:n muistinkäyttöä osituksessa. CPU:n prosessointitehon hyötykäyttö osituksessa voi olla yksi ratkaisu osa-aluepintojen yleistymiseksi reaaliaikaisiin sovelluksiin.

Tutkimusta voisi jatkaa syventymällä osa-aluepintojen laitteistointegraatioon. Schäfer ym. (2015) mukaan tämä voisi tuoda osa-aluepinnat vielä nopeammin kuluttajien saataville ja suunnata kehitystä kohti reaaliaikaista elokuvatasoista grafiikkaa. Kysymyksenä herääkin, tuleeko teknologian ja laitteiston kehittyminen yliajamaan tarpeen renderöinnin optimoinnille vai onko jatkokehitys esimerkiksi luvussa 3 esitellyille menetelmille tarpeen.

Lähteet

Bolz, Jeffrey, ja Peter Schröder. 2002. “Rapid Evaluation of Catmull-Clark Subdivision Surfaces”. Teoksessa *Proceedings of the Seventh International Conference on 3D Web Technology,* 11–17. Web3D ’02. Tempe, Arizona, USA: ACM. ISBN: 1-58113-468-1. doi:10.1145/504502.504505. http://doi.acm.org/10.1145/504502.504505.

Brainerd, Wade, Tim Foley, Manuel Kraemer, Henry Moreton ja Matthias Nießner. 2016. “Efficient GPU Rendering of Subdivision Surfaces Using Adaptive Quadtrees”. *ACM Trans. Graph.* (New York, NY, USA) 35, numero 4 (): 113:1–113:12. ISSN: 0730-0301. doi:10.1145/2897824.2925874. http://doi.acm.org/10.1145/2897824.2925874.

Catmull, E., ja J. Clark. 1978. “Recursively generated B-spline surfaces on arbitrary topological meshes”. *Computer-Aided Design* 10 (6): 350–355. ISSN: 0010-4485. doi:http://dx.doi.org/10.1016/0010-4485(78)90110-0. http://www.sciencedirect.com/science/article/pii/0010448578901100.

Doo, D., ja M. Sabin. 1978. “Behaviour of recursive division surfaces near extraordinary points”. *Computer-Aided Design* 10 (6): 356–360. ISSN: 0010-4485. doi:http://dx.doi.org/10.1016/0010-4485(78)90111-2. http://www.sciencedirect.com/science/article/pii/0010448578901112.

Drone, Shanon, Denis Kovacs, Jason Mitchell ja Denis Zorin. 2010. “Real-Time Creased Approximate Subdivision Surfaces with Displacements”. *IEEE Transactions on Visualization and Computer Graphics* (Los Alamitos, CA, USA) 16:742–751. ISSN: 1077-2626. doi:doi.ieeecomputersociety.org/10.1109/TVCG.2010.31.

Halstead, Mark, Michael Kass ja Tony DeRose. 1993. “Efficient, Fair Interpolation Using Catmull-Clark Surfaces”. Teoksessa *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques,* 35–44. SIGGRAPH ’93. Anaheim, CA: ACM. ISBN: 0-89791-601-8. doi:10.1145/166117.166121. http://doi.acm.org/10.1145/166117.166121.

Loop, Charles. 1987. “Smooth Subdivision Surfaces Based on Triangles”. https://www.microsoft.com/en-us/research/publication/smooth-subdivision-surfaces-based-on-triangles/.

Loop, Charles, Scott Schaefer, Tianyun Ni ja Ignacio Castaño. 2009. “Approximating Subdivision Surfaces with Gregory Patches for Hardware Tessellation”. *ACM Trans. Graph.* (New York, NY, USA) 28, numero 5 (): 151:1–151:9. ISSN: 0730-0301. doi:10.1145/1618452.1618497. http://doi.acm.org/10.1145/1618452.1618497.

Microsoft Windows Dev Center. 2017. “Programming Guide for Direct3D 11: Graphics Pipeline”. https://msdn.microsoft.com/en-us/library/windows/desktop/ff476882(v=vs.85).aspx.

Nießner, Matthias, Charles Loop, Mark Meyer ja Tony Derose. 2012. “Feature-adaptive GPU Rendering of Catmull-Clark Subdivision Surfaces”. *ACM Trans. Graph.* (New York, NY, USA) 31, numero 1 (): 6:1–6:11. ISSN: 0730-0301. doi:10.1145/2077341.2077347. http://doi.acm.org/10.1145/2077341.2077347.

Samet, Hanan, ja Robert E. Webber. 1985. “Storing a Collection of Polygons Using Quadtrees”. *ACM Trans. Graph.* (New York, NY, USA) 4, numero 3 (): 182–222. ISSN: 0730-0301. doi:10.1145/282957.282966. http://doi.acm.org/10.1145/282957.282966.

Schäfer, H., J. Raab, B. Keinert, M. Meyer, M. Stamminger ja M. Nießner. 2015. “Dynamic Feature-adaptive Subdivision”. Teoksessa *Proceedings of the 19th Symposium on Interactive 3D Graphics and Games,* 31–38. i3D ’15. San Francisco, California: ACM. ISBN: 978-1-4503-3392-4. doi:10.1145/2699276.2699282. http://doi.acm.org/10.1145/2699276.2699282.

Shiue, Le-Jeng, Ian Jones ja Jörg Peters. 2005. “A Realtime GPU Subdivision Kernel”. *ACM Trans. Graph.* (New York, NY, USA) 24, numero 3 (): 1010–1015. ISSN: 0730-0301. doi:10.1145/1073204.1073304. http://doi.acm.org/10.1145/1073204.1073304.

Wikipedia, the free encyclopedia. 2006. “A Catmull-Clark subdivision surface”. Creative Commons Attribution-Share Alike 3.0 Unported license. Made by Romainbehar in K-3D, vectorized from raster rendering by Mysid. https://commons.wikimedia.org/wiki/File:Tesselation_pipeline.svg.