Tampereen yliopisto

Akseli Epäilys

SÄTEENSEURANNAN LAITTEISTOKIIHDY-TYSRATKAISUJEN VERTAILU

Kandidaatintyö
Informaatioteknologian ja viestinnän tiedekunta
Tarkastaja: Sakari Lahti
Tammikuu 2024

TIIVISTELMÄ

Akseli Epäilys: Säteenseurannan laitteistokiihdytysratkaisujen vertailu
Kandidaatintyö
Tampereen yliopisto
Tieto- ja sähkötekniikan kandidaattiohjelma
Tammikuu 2024

Säteenseuranta on erittäin realistisen tietokonegrafiikan mahdollistava tekniikka. Säteenseurannan hyödyntäminen on kuitenkin erittäin laskennallisesti vaativaa, minkä vuoksi sen nopeuttamiseen on kehitetty erityistä laitteistoa. Tämän työn tavoitteena on selvittää, minkälaisia ratkaisuja säteenseurannan laitteistokiihdytykseen on olemassa ja miten ne eroavat toisistaan.

Työ on toteutettu kirjallisuuskatsauksena. Työn alkuosa käsittelee säteenseurannan perusteita ja sen käytännön toteutusta säteenseurantaputken muodossa. Seuraavaksi käsitellään kiihdytysrakenteita sekä säteenseurannan laitteistokiihdytystä yleisellä tasolla. Tämän jälkeen tarkastellaan kolmen PC-markkinoilla toimivan valmistajan ratkaisuja, ja lopuksi verrataan niitä keskenään.

Kiihdytysrakenteiden osalta BVH havaittiin suosituimmaksi ratkaisuksi osana laitteistokiihdytystä, ja säteenseurantaprosessista tunnistettiin joitakin osia, jotka soveltuvat toteutettaviksi laitteistotasolla. Näitä ovat ainakin leikkaustestaukset, kiihdytysrakenteen läpikäyntilogiikka ja kiihdytysrakenteen muodostus. Lisäksi koherenssilajittelun todettiin olevan mahdollinen tapa optimoida laitteisto säteenseurantaa varten.

Tarkastelluista ratkaisuista AMD:n ratkaisut osoittautuivat yksinkertaisimmiksi, toteuttaen ainoastaan leikkaustestaukset laitteistotasolla. Tässä ratkaisussa todettiin kuitenkin hyväksi puoleksi sen ohjelmoitavuus. Intelin Xe-HPG-arkkitehtuuri taas on kehittyneempi ja toteuttaa leikkaustestaukset, kiihdytysrakenteen läpikäynnin ja varjostimien koherenssilajittelun laitteistossa. Nvidian Ada Lovelace -arkkitehtuurin todettiin sisältävän eniten ominaisuuksia: leikkaustestaukset, kiihdytysrakenteen läpikäynti, varjostimien koherenssilajittelu, läpinäkyvyystestausta nopeuttava Opacity Micromap Engine ja yksinkertaisemman BVH:n mahdollistava Displaced Micro-Mesh Engine.

Leikkaustestauskapasiteetteja vertaillessa kävi ilmi, että AMD:n RDNA 3 ja Intelin Xe-HPG omaavat maksimikonfiguraatioissaan verrattavissa olevan laatikkotestauskapasiteetin, mutta RDNA 3:n kolmiotestauskapasiteetti on noin 1,6-kertainen Xe-HPG:hen verrattuna. Nvidian ratkaisujen leikkaustestauskapasiteeteista ei ole täysiä tietoja saatavilla, mutta olemassa olevien tietojen perusteella tehtiin päätelmä, jonka mukaan Nvidian Ada Lovelace -arkkitehtuurin kolmiotestauskapasiteetti on huomattavasti kilpailijoita korkeampi: noin kuusinkertainen RDNA 3:een verrattuna.

Avainsanat: säteenseuranta, laitteistokiihdytys, tietokonegrafiikka, grafiikkaprosessori

SISÄLLYSLUETTELO

1. JOHDANTO

Reaaliaikainen tietokonegrafiikka on vihdoin aloittanut kauan odotetun siirtymän säteenseurantaan, joka on esimerkiksi elokuvien erikoistehosteista tuttu tekniikka. Jo 1960-luvulla alkunsa saanutta säteenseurantaa pidetään tietokonegrafiikan ”graalin maljana”, sillä sen avulla voidaan simuloida valon käyttäytymistä hyvinkin tarkasti, mahdollistaen erittäen realistisen grafiikan.

Säteenseurannassa on kuitenkin kriittinen ongelma; se on erittäin laskennallisesti vaativa tekniikka. Elokuvateollisuuden tapauksessa yksittäisen kuvan valmistumista on mahdollista odottaa jopa tunteja, ja laskentaa suorittamassa saattaa olla jopa tuhansia prosessoreja sisältävä supertietokone. Reaaliaikaisen grafiikan tapauksessa sen sijaan kuvan valmistumista on aikaa odottaa korkeintaan joitakin kymmeniä millisekunteja, ja laskenta suoritetaan useimmiten loppukäyttäjän omalla tietokoneella.

Jotta visio reaaliaikaisesta säteenseurannasta olisi realistisesti mahdollista toteuttaa, täytyy säteenseurantaa varten luoda erityistä laitteistoa, joka suorittaa laskennan yleiskäyttöisen prosessorin sijasta. Tämä *laitteistokiihdytys* mahdollistaa säteenseurantasuorituskyvyn moninkertaistumisen. Merkittävä kehitys säteenseurannan laitteistokiihdytyksessä tapahtui vuonna 2018 Nvidian Turing-grafiikkaprosessoriarkkitehtuurin julkaisun myötä. Tämän jälkeen myös monet muut valmistajat ovat tuoneet markkinoille omia säteenseurannan laitteistokiihdytysratkaisujaan.

Tämän työn tavoitteena on tarkastella olemassa olevia säteenseurannan laitteistokiihdytysratkaisuja ja niiden eroja. Työ pohjustaa aihetta esittelemällä säteenseurannan teorian perusteita ja käytännön toteutusta sekä säteenseurannassa hyödynnettäviä kiihdytysrakenteita. Työ esittelee säteenseurannan laitteistokiihdytystä yleisellä tasolla erittelemällä säteenseurantaprosessin laitteistokiihdytettäväksi soveltuvia osia, minkä jälkeen työ tarkastelee kolmen PC-markkinoilla toimivan valmistajan laitteistokiihdytysratkaisuja ja vertailee niiden lähestymistapoja eri näkökulmista.

Aiheen käsittely alkaa luvusta 2, joka tarkastelee ensin säteenseurantaa yleisellä tasolla ja esittelee sitten käytännön säteenseurannan vaiheet säteenseurantaputken muodossa. Luku 3 esittelee säteenseurannan kiihdytyksen perusperiaatteita. Luvussa 4 tarkastellaan eri valmistajien tarjoamia kaupallisia ratkaisuja, joita vertaillaan toisiinsa luvussa 5. Viimeinen luku eli luku 6 sisältää yhteenvedon vertailun tuloksista.

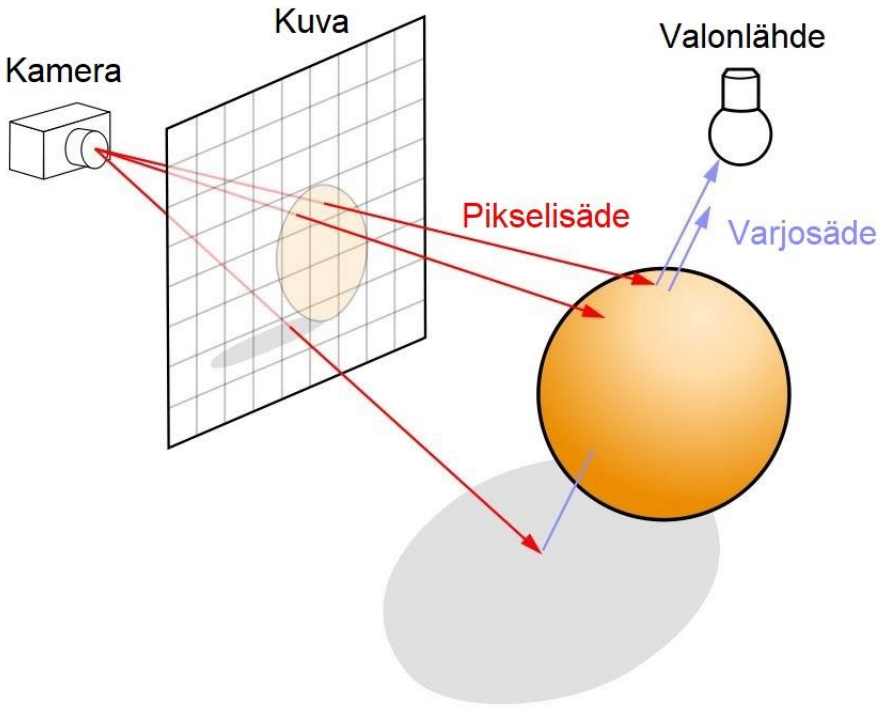
2. SÄTEENSEURANTA

2.1 Teoria

Renderöinti on prosessi, jossa luodaan kuva jonkinlaisen mallin pohjalta. Yksinkertainen malli voisi esimerkiksi kuvata kolmion pisteiden sijainnit kaksiulotteisessa koordinaatistossa. Tällöin renderöintialgoritmi voisi määrittää jokaisen pikselin värin esimerkiksi sen perusteella, ovatko pikselille määritetyt koordinaatit kuvatun kolmion alueella. Renderöinti on tietokoneiden käytettävyyden kannalta kriittinen prosessi, sillä kaikki näytöllä näkyvä on täytynyt ensin renderöidä. 3D-renderöinti viittaa renderöintiin, jossa malli on kolmiulotteinen.

Säteenseuranta (engl. ray tracing) on 3D-renderöinnissä hyödynnetty tekniikka, jossa seurataan valonsäteiden kulkua virtuaalisessa ympäristössä [1]. Säteenseurantaa voidaan hyödyntää myös muulla tavoin, esimerkiksi realistisen tilaäänen tuottamiseen [2]. Tämä työ kuitenkin keskittyy käsittelemään säteenseurantaa nimenomaan tietokonegrafiikan näkökulmasta.

Säteenseurantaa voidaan soveltaa renderöintiin usein eri tavoin. Nykypäivänä olennaisimpia säteenseurantaa hyödyntäviä renderöintitekniikoita ovat polunseuranta (engl. path tracing) [3] sekä erilaiset hybriditekniikat [4], jotka hyödyntävät säteenseurantaa tuottamaan vain osan kuvasta. Koska säteenseuranta mallintaa oikean valon käyttäytymistä, voi sen avulla tuottaa erittäin realistista grafiikkaa. Monet renderöintitekniikat, kuten reaaliaikaisessa grafiikassa yleinen rasterointi [5], vaativat monimutkaisia erityiskeinoja tuottamaan monia reaalimaailman optisia ilmiöitä, kuten esimerkiksi heijastuksia ja valon taittumista, kun taas säteenseurantaan perustuvien tekniikoiden avulla nämä muodostuvat luonnostaan. Kuvaa muodostaessa valonsäteitä seurataan tehokkuussyistä useimmiten käänteisessä suunnassa eli säteet lähetetään virtuaalisesta kamerasta kuvan pikselien muodostaman ruudukon määräämiin suuntiin. Tätä havainnollistaa kuva 1.



Kuva 1. Havainnollistus yksinkertaisesta säteenseuranta-algoritmista. Muokattu lähteestä [6].

Pikselin väri määräytyy säteen osumakohdasta kameraa kohti lähtevän valon perusteella. Toisin sanoen tarvitsee määrittää valo, joka kulkee sädettä pitkin, mutta vastakkaissuuntaisesti. Tämän määrittäminen tarkasti ei ole helppo tehtävä, vaan käytännöllisesti katsoen mahdotonta. Siksi säteenseurantamenetelmät pyrkivät approksimoimaan lähtevän valon määrää ja laatua. Jotta lähtevä valo voitaisiin määrittää, tarvitaan tieto pisteen emittoimasta valosta, pisteeseen saapuvasta valosta sekä materiaalin ominaisuuksista pinnalla, jolla piste sijaitsee. Olennaisin ja ongelmallisin osa on saapuvan valon määritys. Suuntia, joista valoa voi saapua pisteeseen, on rajaton määrä. Lisäksi valo voi saapua usean kimmokkeen kautta.

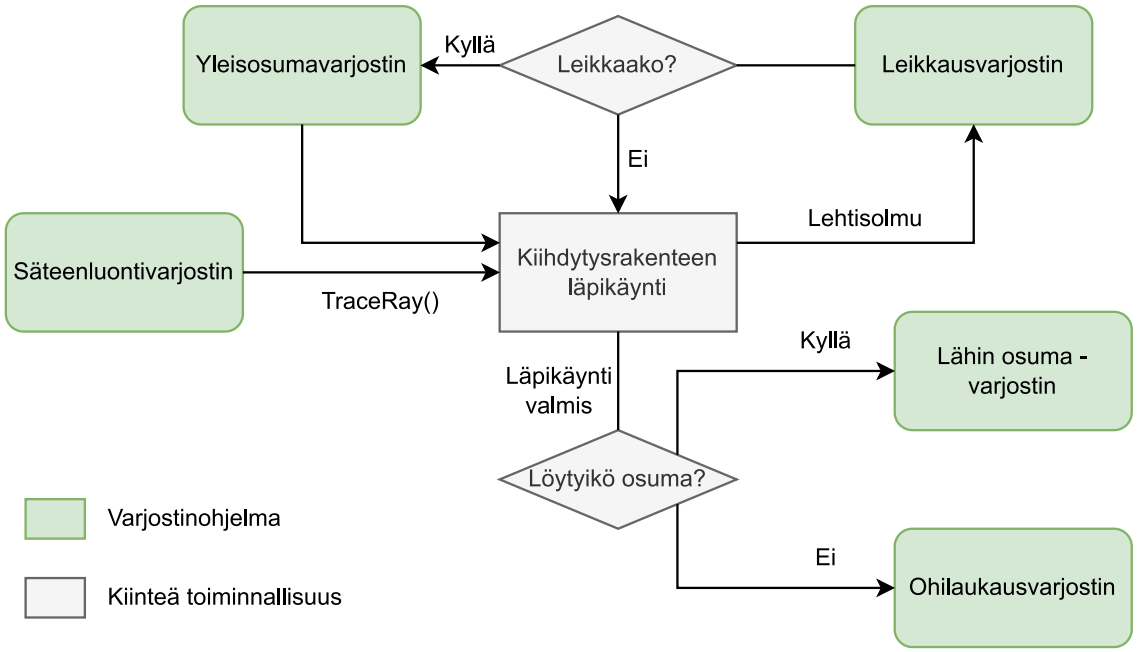
Yksinkertainen tapa approksimoida saapuvaa valoa on lähettää säde jokaista tunnettua valonlähdettä kohti, kuten esimerkiksi Whittedin [1] menetelmässä. Jos säde osuu matkalla johonkin muuhun, on piste varjossa valonlähteeseen nähden, eikä kyseistä valonlähdettä oteta huomioon. Muussa tapauksessa valonlähteen tuottama valo lisätään lopputulokseen. Tämä tapa määrittää saapuva valo on tehokas, mutta yksinään tämän menetelmän hyödyntäminen tuottaa kuvan, joka sisältää merkittävää vääristymää. Toisin sanoen se ei kykene ottamaan huomioon esimerkiksi epäsuoraa valaistusta. Yksi esimerkki vääristämättömästä menetelmästä on polunseuranta [3]. Polunseuranta tuottaa

lopputuloksen, joka on keskimäärin oikea, mutta sisältää satunnaista virhettä, joka ilmenee kuvassa kohinana. Polunseuranta approksimoi pisteeseen saapuvaa valoa lähettämällä useita säteitä satunnaisiin suuntiin, ja määrittämällä näitä säteitä pitkin vastakkaiseen suuntaan kulkevan valon. Seurauksena on rekursiivinen algoritmi, jossa säteet kimpoilevat ympäriinsä. Käytännön renderöinnin tapauksessa kimmokkeiden maksimimäärää on rajoitettava, jottei kimmokkeiden ääretön määrä estä algoritmin loppuun suoritusta.

2.2 Säteenseurantaputki

Grafiikkaputki (engl. graphics pipeline) on eräänlainen kuvaus vaiheista, jotka suoritetaan grafiikan renderöimiseksi. Grafiikkaputki koostuu kiinteistä ja ohjelmoitavista osista. Kiinteiden osien toiminta on ennalta määrättyä, ja ne suoritetaan usein erityisesti niitä varten suunnitelluissa grafiikkaprosessorin osissa. Ohjelmoitavien osien toiminnan yksityiskohdat taas ovat kehittäjien päätettävissä, ja ne suoritetaan grafiikkaprosessorin yleiskäyttöisissä laskentayksiköissä. Ohjelmoitavien osien toiminnan määrittäviä lyhyitä ohjelmia kutsutaan varjostimiksi (engl. shader).

Perinteinen grafiikkaputki perustuu rasterointiin, joka on laskennallisesti verrattain kevyt renderöintitekniikka suhteessa säteenseurantaan perustuviin tekniikoihin ja siten usein reaaliaikaiseen renderöintiin paremmin soveltuva [7]. Säteenseurantaputki on säteenseurannan tarpeisiin muovattu versio grafiikkaputkesta. Kuva 2 esittää yksinkertaistetun mallin DirectX 12 -rajapinnan [8] määrittämästä säteenseurantaputkesta. Myös muiden grafiikkarajapintojen, kuten Vulkanin [9], säteenseurantaputket noudattavat pitkälti samaa mallia.



Kuva 2. DirectX 12 -rajapinnan säteenseurantaputken vuokaavio, mukaillen lähdettä [10].

Kuvan 2 mukaisen säteenseurantaputken ohjelmoitavia osia toteuttavat säteenluontivarjostin (engl. ray generation shader), leikkausvarjostin (engl. intersection shader), yleisosumavarjostin (engl. any-hit shader), lähin osuma -varjostin (engl. closest-hit shader) sekä ohilaukausvarjostin (engl. miss shader). Kiinteät osat muun muassa suorittavat kiihdytysrakenteen läpikäynnin sekä ohjaavat prosessin kulkua.

Säteenluontivarjostimen tehtävänä on käynnistää säteenseurantaputki ja luoda säde kutsumalla TraceRay()-funktiota. Putken päätteeksi se myös kirjoittaa lopullisen tuloksen muiden varjostimien tuottamien tulosten perusteella. Keskeisessä osassa säteenseurantaputkea on kiihdytysrakenteen läpikäynti. Kiihdytysrakenteen ansiosta ei tarvitse tarkistaa jokaista geometrian peruselementtiä eli primitiiviä erikseen osuman varalta, vaan mahdollinen osuma voidaan rajata pieneen joukkoon primitiivejä. Kiihdytysrakenteiden toimintaa käsitellään tarkemmin luvussa 3.1.

Kun kiihdytysrakennetta läpikäytäessä päädytään lehtisolmuun, voidaan suorittaa leikkausvarjostin, joka tarkistaa, osuuko säde johonkin tiettyyn primitiiviin. Leikkausvarjostin on tarkoitettu käytettäväksi itse määritettyjen primitiivityyppien kanssa. Useimmissa tapauksissa primitiiveinä käytetään kolmioita, jolloin erillinen leikkausvarjostin ei ole tarpeellinen, vaan voidaan hyödyntää grafiikkaprosessorin oletuskäyttäytymistä. Kun säteen on todettu osuvan primitiiviin, on mahdollista suorittaa yleisosumavarjostin, jolla

osuma voidaan tarvittaessa hylätä. Tyypillinen syy osuman hylkäämiselle on osumakohdan läpinäkyvyys [11]. Joitakin yksityiskohtaisia muotoja mallinnetaan usein varsinaisen geometrian sijaan tekemällä geometriaan määritetystä tekstuurista läpinäkyvä kohdissa, joissa geometriaa ei kuuluisi olla.

Kun säteen alkupistettä lähimpänä oleva hyväksytty osumakohta on määritetty, suoritetaan lähin osuma -varjostin (engl. closest hit shader). Lähin osuma -varjostin voi suorittaa valaistuslaskentaa ja palauttaa dataa säteenluontivarjostimelle. Rekursiivisen säteenseurannan tapauksessa se voi myös lähettää uusia säteitä kutsumalla TraceRay()-funktiota. Tapauksessa, jossa hyväksyttyä osumaa ei löytynyt, suoritetaan ohilaukausvarjostin (engl. miss shader). Ohilaukausvarjostin voi palauttaa dataa säteenluontivarjostimelle esimerkiksi ympäristökartan (engl. environment map) perusteella.

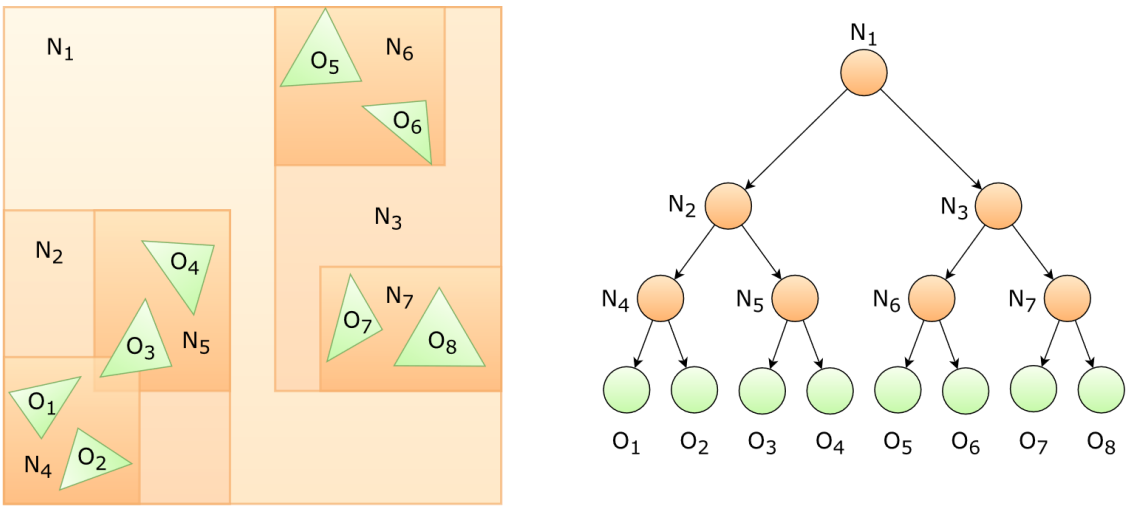
Säteenseurantaputken suorituksen päätteeksi palataan suorittamaan putken käynnistänyt säteenluontivarjostin loppuun. Säteenluontivarjostin määrittää tällöin lopullisen tuloksen eli pikselin värin muiden varjostimien tuottamien tulosten perusteella.

3. SÄTEENSEURANNAN KIIHDYTYS

3.1 Kiihdytysrakenteet

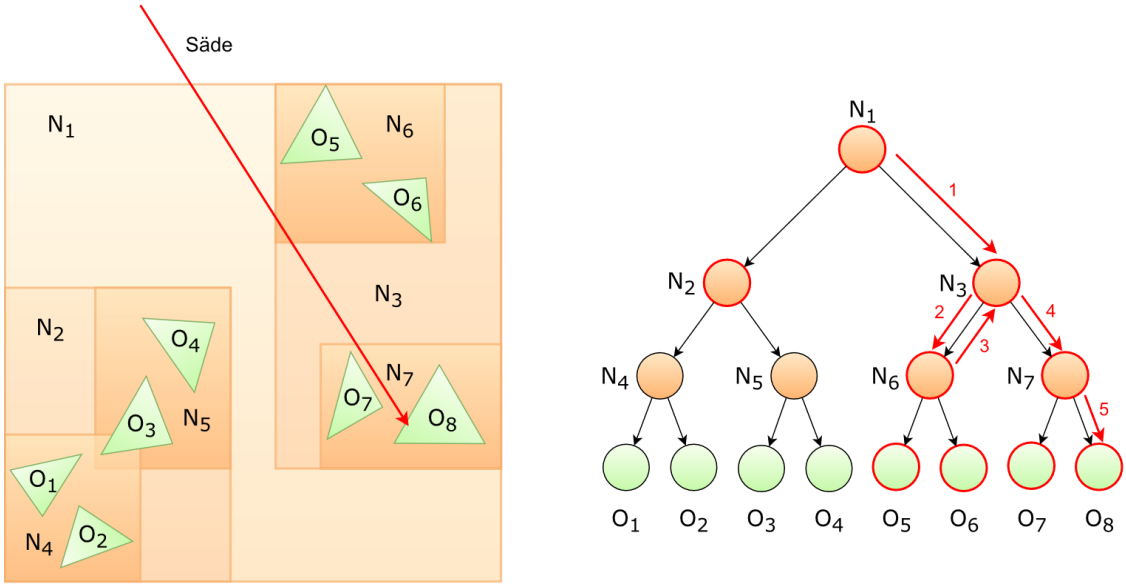
Säteenseurannan keskiössä on säteen ja ympäristössä sijaitsevan geometrian leikkauspisteen määrittäminen. Jokaisen primitiivin tarkistaminen leikkauspisteen varalta olisi kuitenkin kohtuuttoman raskasta laskennallisesti. Tästä syystä säteenseurannassa hyödynnetään kiihdytysrakenteita. Kiihdytysrakenteiden ansiosta suurin osa primitiiveistä voidaan tehokkaasti poissulkea mahdollisina osumapisteen sijainteina. Kiihdytysrakenteet ovat tyypillisesti puurakenteita, joiden lehdet sisältävät primitiivejä, ja muut solmut jakavat tilaa tai geometriaa osiin jollakin tavalla.

Joitakin esimerkkejä mahdollisista kiihdytysrakenteista ovat Octree, kd-puu ja BVH (Bounding Volume Hierarchy) [12]. Näistä BVH on saavuttanut suurimman suosion; se on käytössä kaikissa viimeaikaisissa laitteistokiihdytysratkaisuissa (katso luku 4). Syitä BVH:n suosiolle ovat ainakin sen ennakoitavissa oleva muistin käyttö, algoritmia muuttamalla säädettävät muodostusajat sekä yksinkertainen rakenteen päivitysprosessi geometrian muuttuessa [13]. BVH-rakenne perustuu rajauslaatikoihin (engl. bounding box), usein akselien suuntaisesti asemoituihin sellaisiin eli AABB:ihin (Axis-Aligned Bounding Box) [13]. Puurakenteen alimmalla tasolla olevat laatikot sisältävät yhden tai useamman primitiivin, ja ylemmillä tasoilla olevat laatikot sisältävät alemman tason laatikoita. Kuva 3 havainnollistaa yksinkertaista BVH2-rakennetta eli BVH-rakennetta, jossa jokaisella solmulla on korkeintaan kaksi lasta.



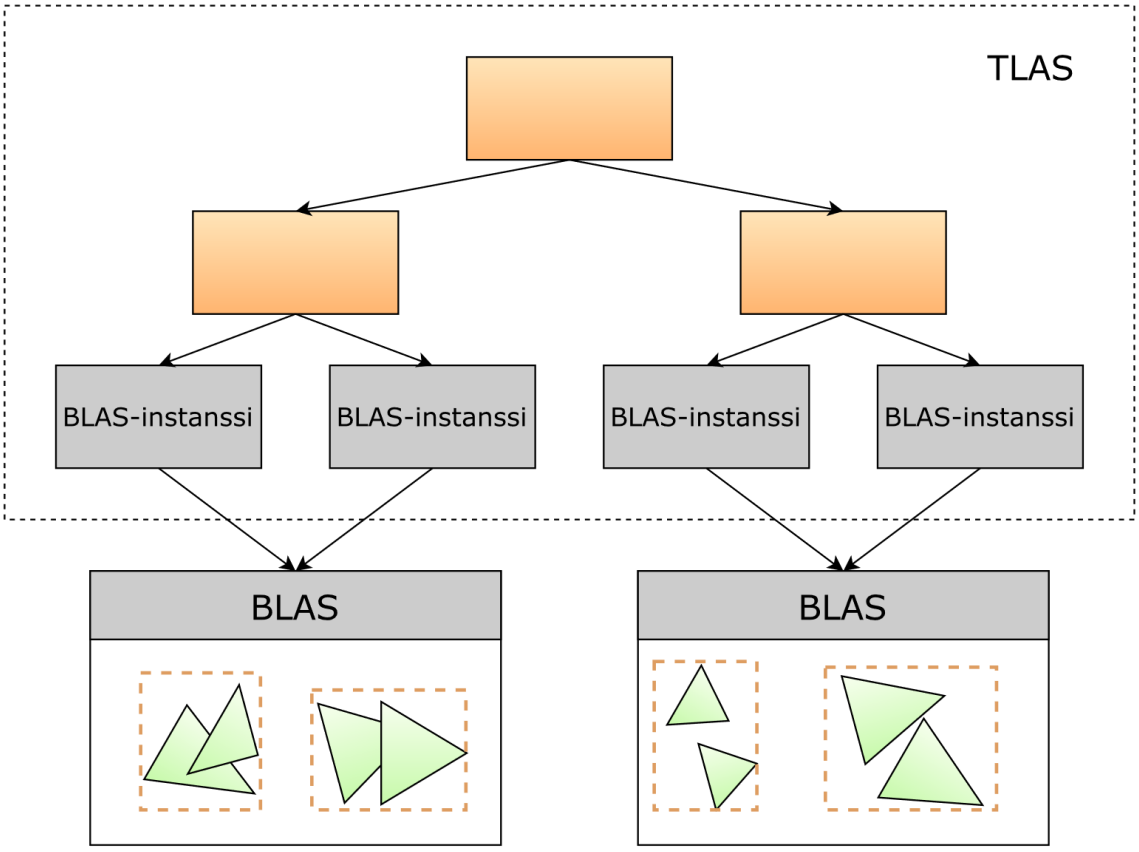
Kuva 3. Esimerkki yksinkertaisesta BVH-rakenteesta. Vasemmalla kaksiulotteinen esitys. Oikealla puumuodossa oleva esitys. Mukailee lähdettä [14].

BVH:n taustalla on yksinkertainen idea; jos säde ei osu johonkin tiettyyn BVH-rakenteen laatikkoon, ei se voi osua myöskään mihinkään kyseisen laatikon lapsista, sillä nämä sijaitsevat laatikon sisällä. Tästä johtuen säteen osumapistettä määritettäessä aloitetaan ylimmän tason laatikoista ja määritetään, mitä näistä laatikoista säde leikkaa. Kuva 4 esittää esimerkin erään mahdollisen BVH:n läpikäyntialgoritmin toiminnasta. Punaisella kehystetyt solmut ovat solmuja, joiden leikkausta säteen kanssa täytyi testata algoritmin suorituksen aikana. Tässä esimerkkitapauksessa täytyi siis testata säteen leikkausta yhteensä viiden laatikon ja neljän kolmion kanssa.



Kuva 4. Esimerkki BVH:n läpikäyntialgoritmista.

Tehokkuussyistä BVH jaetaan usein kaksitasoiseksi hierarkiaksi, joka koostuu alatason kiihdytysrakenteista (BLAS, Bottom Level Acceleration Structure) sekä ylätason kiihdytysrakenteesta (TLAS, Top Level Acceleration Structure) [15]. BLAS sisältää kiihdytysrakenteen jollekin geometriselle kokonaisuudelle, kuten vaikkapa jonkin esineen 3D-mallille. BLAS:llä voi olla useita instansseja, eli samaa BLAS:ää voidaan käyttää usealle eri kopiolle samasta 3D-mallista kopioimatta koko rakennetta. TLAS:n lehdissä sijaitsevien BLAS-instanssien yhteyteen liitetään muunnosmatriisi, joka määrittää kyseisen instanssin sijainnin ja orientaation. Kuva 5 havainnollistaa kaksitasoisen BVH:n rakennetta.



Kuva 5. Esimerkki kaksitasoisesta BVH-rakenteesta. Mukailee lähdettä [16].

Nykyisten säteenseurannan kiihdytysominaisuuksia sisältävien grafiikkaprosessorien tapauksessa BVH:n rakentaminen on tyypillisesti grafiikkaprosessorin ajurien vastuulla. Säteenseurantaa tukevat grafiikkarajapinnat tarjoavat muutamia lippumuuttujia, joilla voi vaikuttaa rakentamisprosessiin esimerkiksi asettamalla se suosimaan nopeampia rakennusaikoja nopeampien läpikäyntiaikojen sijaan tai päinvastoin [17], [18].

3.2 Säteenseurannan laitteistokiihdytys

Moderni säteenseurannan laitteistokiihdytys toteutetaan tyypillisesti grafiikkaprosessorissa sijaitsevien erillisten yksiköiden muodossa. Nämä yksiköt on suunniteltu suorittamaan tiettyjä säteenseurannassa usein toistuvia operaatioita huomattavasti nopeammin kuin grafiikkaprosessorin yleiskäyttöiset laskentayksiköt. Myös pelkkien grafiikkaprosessorin yleiskäyttöisten yksiköiden hyödyntämistä säteenseurannassa voidaan pitää eräänlaisena laitteistokiihdytyksen muotona, mutta tämä työ keskittyy vain erityisesti säteenseurantaa varten suunniteltuihin laitteistoyksiköihin. Tällaiset yksiköt eivät ole ohjelmoitavia, joten laitteistossa toteutettavien säteenseurantaprosessin osien täytyy olla toiminnallisuudeltaan kiinteitä.

Whitted [1] totesi yli 95% säteenseurantaohjelmansa laskenta-ajasta kuluvan säteiden ja primitiivien leikkauspisteiden laskentaan. Nykyisten kiihdytysrakenteiden ansiosta tämä lukema tuskin enää pitää paikkansa, mutta säde-primitiivi-leikkaustestit ovat edelleen erittäin olennainen osa säteenseurantaa. Lisäksi kiihdytysrakennetta läpikäytäessä tarvitsee määrittää, leikkaako säde kiihdytysrakenteen laatikoita, joten myös säde-laatikko-leikkaustestit muodostuvat olennaisiksi. Tästä syystä useat säteenseurannan laitteistokiihdytysratkaisut sisältävät erillisiä laitteistoyksiköitä näiden operaatioiden suorittamiseen [19]–[21].

Kiihdytysrakenteen läpikäynti voidaan määrittää kiinteäksi ominaisuudeksi – näin myös ainakin osa grafiikkarajapinnoista tekee (katso luku 2.2) – ja sopii siten toteutettavaksi laitteistossa (esimerkiksi [19], [20]). Tällöin kiihdytysrakenteen läpikäynnin suorittava yksikkö siis vastaanottaa säteen ja kiihdytysrakenteen tiedot ja suorittaa koko läpikäyntiprosessin, hyödyntäen myös edellä esiteltyjä leikkaustestausyksiköitä, ja palauttaa säteen lopullisen osumapisteen. Lopullisen osumapisteen määrittämiseksi tarvitsee joskus suorittaa varjostinohjelma, kuten esimerkiksi yleisosumavarjostin. Tällöin läpikäyntiyksikön täytyy lähettää varjostinohjelma yleiskäyttöisten laskentayksiköiden suoritettavaksi ja jatkaa läpikäyntiä tuloksen valmistuttua.

Reaaliaikaisen säteenseurannan suorituskyvyn kannalta olennaista on myös kiihdytysrakenteen rakennus. Kiihdytysrakenne täytyy päivittää tai uudelleenrakentaa aina kun geometria muuttuu. Monissa tapauksissa tämä tarkoittaa uudelleenrakennusta jokaisella kuvalla [22]. Kiihdytysrakenteen laatu vaikuttaa myös itse säteenseurannan nopeuteen, mutta laadukkaamman rakenteen muodostaminen vaatii enemmän laskentaa, mikä saattaa muodostua ongelmaksi reaaliaikaisessa säteenseurannassa. Myös kiihdytysrakenteen rakentaminen sopii toteutettavaksi laitteistossa [23].

Säteenseurannan suorittaminen grafiikkaprosessorilla muodostaa koherenssiongelman [24]. Grafiikkaprosessorien yleiskäyttöiset laskentayksiköt pohjautuvat SIMD-periaatteeseen (Single Instruction, Multiple Data), eli ne suorittavat yhden operaation samanaikaisesti usealla eri syötedatalla. Säteenseurannassa eri suuntiin kimpoilevat säteet käynnistävät erilaisia varjostinohjelmia, mikä heikentää SIMD-laskentayksiköiden tehokkuutta. Epäkoherenttien säteiden takia täytyy grafiikkaprosessorin myös jatkuvasti hakea muistista eri varjostinohjelmia sekä kiihdytysrakenteen osia, mikä heikentää välimuistijärjestelmän tehokkuutta. Tehokkuuden parantamiseksi jotkin laitteistoratkaisut optimoivat joko varjostinohjelmien [25], [26] tai itse säteiden [19] suoritusjärjestystä.

4. KAUPALLISET RATKAISUT

Tämä luku esittelee PC-markkinoilla toimivien valmistajien eli AMD:n, Intelin ja Nvidian säteenseurantalaitteistoa. PC-alustaan liittyy useimpia muita alustoja vähemmän rajoitteita esimerkiksi virrankulutukseen liittyen, joten sillä on muita alustoja paremmat edellytykset reaaliaikaiseen säteenseurantaan. Muita säteenseurantalaitteiston valmistajia ovat ainakin Imagination Technologies [19], Apple [27], ARM [28] ja Qualcomm [29].

4.1 AMD

AMD:lla on kaksi säteenseurannan kiihdytyselementtejä sisältävää grafiikkaprosessoriarkkitehtuuria: RDNA 2 [21] (julkaistu 2020) ja RDNA 3 [30] (julkaistu 2022). Näissä arkkitehtuureissa jokainen CU (Compute Unit) sisältää yhden Ray Accelerator -yksikön, joka kykenee suorittamaan säteen leikkaustestausoperaation joko neljälle BVH-rajauslaatikolle tai yhdelle kolmiolle kellojaksossa. Laitteistokiihdytystä BVH:n läpikäynnille ei ole, eli BVH:n läpikäyntiprosessia ohjataan grafiikkaprosessorin yleiskäyttöisillä laskentayksiköillä.

Ray Accelerator -yksikölle annetaan syötteenä osoitin BVH-solmuun, jonka lapsisolmuille operaatio halutaan suorittaa, sekä säteen maksimipituus, alkupiste, suunta ja käänteinen suunta. Jos syötteenä oleva BVH-solmu sisältää laatikoita, tekee Ray Accelerator -yksikkö leikkaustestin kaikille laatikoille ja palauttaa osoittimet laatikoihin osumaetäisyyden mukaisessa järjestyksessä. BVH-solmun sisältäessä kolmioita tekee Ray Accelerator -yksikkö leikkaustestin yhdelle kolmiolle, ja palauttaa osumastatuksen, kolmion id:n sekä leikkausajankohdan, joka määrittää missä kohdassa sädettä leikkauspiste sijaitsee. Vaihtoehtona on myös palautustila, jossa osumastatus ja kolmion id korvataan osumapisteen barysentrisillä koordinaateilla. [21], [30]

AMD:n AMDVLK-ajurien lähdekoodin [31] perusteella Ray Accelerator -yksikkö tukee kolmioiden pakkausta. PAIR_TRIANGLE_COMPRESSION-pakkaustilassa BVH:n lehdet sisältävät kaksi kolmiota, jotka jakavat yhteisen sivun [32], [33]. Yhteisen sivun ansiosta kolmiopari voidaan esittää vain neljän pisteen avulla. AMDVLK:n lähdekoodissa on myös joitakin kohtia, jotka vaikuttaisivat viittaavan neljän kolmion pakkaukseen, jossa nämä neljä kolmiota esitettäisiin viiden pisteen avulla. Tähän viittaa se, että NODE_TYPE_TRIANGLE-määrittelyjä on neljä kappaletta [33]. Lisäksi CalcTriangleCompressionVertexOffsets-funktio sisältää kartoitukset pisteille myös neljän kolmion ta-

pauksessa [34]. AMDVLK ei kuitenkaan vaikuta tällä hetkellä hyödyntävän neljän kolmion pakkausta, mutta sisältää listauksen RESERVED-pakkaustilasta [33], joka voisi mahdollisesti olla varattu tätä varten.

RDNA 3 lisää laitteistotuen säteenseurantarajapintojen tarjoamille lippumuuttujille, joilla voidaan kytkeä päälle erilaisia karsintatapoja (engl. culling) [30]. Karsinta määrittää joukon primitiivejä, jotka jätetään huomiotta kuvaa renderöitäessä. Laitteistotukeen sisältyy ainakin takapintakarsinta (engl. backface culling), läpinäkyvyys-/läpinäkymättömyyskarsinta ja primitiivityyppiin pohjautuva karsinta. Takapintakarsinta karsii pois primitiivit, joiden etupinta osoittaa pois kamerasta. Läpinäkyvyyskarsinta karsii primitiivit, joiden on merkattu sisältävän läpinäkyviä kohtia, ja läpinäkymättömyyskarsinta primitiivit, jotka on merkattu läpinäkymättömiksi. Primitiivityyppiin pohjautuva karsinta karsii joko kaikki kolmiot, mutta jättää muun tyyppiset primitiivit karsimatta, tai päinvastoin. Lippumuuttujien avulla RDNA 3 kykenee suorittamaan aikaista alipuun karsintaa [35, katso 36]. RDNA 3 -laitteisto vaikuttaisi siis tarkastavan karsintamahdollisuuksia jonkin verran etukäteen BVH-rakenteen läpikäynnin aikana. Tämä mahdollistaa muutaman viimeisen leikkaustestauksen ohittamisen, jos kaikkien alipuun primitiivien todetaan olevan karsittavissa.

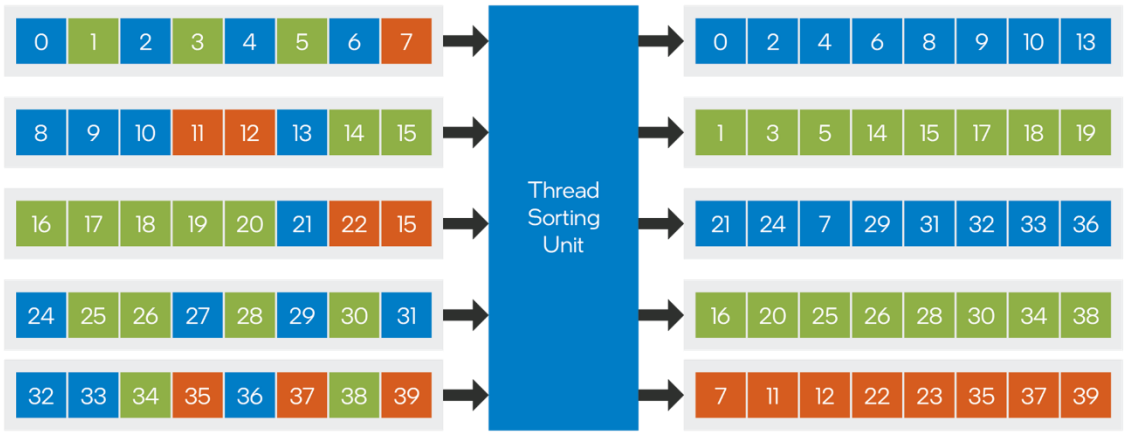
4.2 Intel

Intelin säteenseurantaa tukeva grafiikkaprosessoriarkkitehtuuri on vuonna 2022 julkaistu Xe-HPG-arkkitehtuuri. Xe-HPG koostuu Render Slice -lohkoista, joita on maksimissaan kahdeksan. Jokainen Render Slice sisältää neljä X^e-ydintä yleislaskentaa varten. Lisäksi jokaista X^e-ydintä kohden Render Slice sisältää yhden Ray Tracing Unit -yksikön (RTU), sekä yhden Thread Sorting Unit -yksikön (TSU). RTU:ita ja TSU:ita on siis maksimissaan 32. [37]

RTU käy BVH-rakenteen läpi itsenäisesti ja suorittaa tarvittavat leikkaustarkistukset palauttaen säteen osumapisteen. RTU koostuu BVH-välimuistista, kahdesta Traversal Pipelinesta sekä Triangle/Quad Intersection -yksiköstä. Traversal Pipeline sisältää BVH:n läpikäyntilogiikan sekä kuusi Box Intersection -yksikköä, jotka toteuttavat säteen ja BVH-laatikoiden leikkaustestauksen. [38] Traversal Pipeline kykenee myös muuntamaan säteen BLAS-instansointia varten [39]. Tämä tarkoittaa säteen matemaattisen esityksen muuntamista sopivaksi BLAS:n sisäiseen koordinaatistoon, joka usein poikkeaa TLAS:n koordinaatistosta.

TSU pyrkii ratkaisemaan koherenssiongelman varjostimien suorituksen osalta [25]. TSU ryhmittelee suoritusta odottavat varjostinohjelmakutsut niiden varjostintietueosoitteiden

perusteella. Tällöin kutsut samaan varjostinohjelmaan voidaan suorittaa yhdessä ryppäässä hyödyntäen grafiikkaprosessorin SIMD-yksiköitä. Kuva 6 havainnollistaa TSU:n toimintaa. Numeroidut laatikot kuvaavat varjostinohjelmakutsuja ja värit kuvaavat varjostintietueosoitteita.



Kuva 6. Havainnollistus TSU:n toiminnasta [25].

Intelin käyttämän BVH-rakenteen lehdet sisältävät kaksi kolmiota, jotka jakavat yhden yhteisen sivun [39]. Tämä tarjoaa joitakin merkittäviä etuja yhden kolmion ratkaisuun verrattuna. Ensinnäkin, kahden vierekkäisen kolmion yhteinen AABB on harvoin merkittävästi suurempi kuin yksittäisen kolmion [11]. Tällä tavoin voidaan siis vähentää tarkistettavien BVH-laatikoiden määrää kasvattamatta kolmiotarkastusten määrää merkittävästi. Toiseksi, joitakin yhteiseen sivuun liittyviä laskentatuloksia voidaan käyttää uudelleen, mikä vähentää laskentaresurssien tarvetta [11]. Kolmanneksi, kolmiopari voidaan esittää pienemmällä määrällä dataa kuin kaksi erillistä kolmiota (neljä pistettä kuuden sijaan) [11]. Triangle/Quad Intersection -yksikkö suorittaa molempien kolmioiden testauksen samanaikaisesti.

Xe-HPG-arkkitehtuuri tarjoaa tuen myös läpikäyntivarjostimille (engl. traversal shader) [39]. Läpikäyntivarjostimet eivät vielä ole osana tärkeimpien grafiikkarajapintojen säteenseurantaputkia, mutta tilanne saattaa muuttua tulevaisuudessa. Läpikäyntivarjostimet lisäisivät ohjelmoitavan vaiheen osaksi BVH-rakenteen läpikäyntiä. Tätä varten BVH-rakenteeseen lisättäisiin uusi solmutyyppi, ohjelmoitava instanssi, johon tultaessa läpikäyntivarjostin suoritettaisiin [40].

Läpikäyntivarjostimien lisäämiselle motiivina toimii se, että niillä voitaisiin mahdollistaa parempia tekniikoita dynaamisen yksityiskohtaisuustason (engl. dynamic level of detail) eli dynaamisen LOD:n hyödyntämiseksi säteenseurannassa [40]. Dynaaminen LOD on tekniikka, jossa 3D-objekti voidaan korvata epätarkemmalla, pienemmän kolmiomäärän

omaavalla vastinkappaleella, jos se täyttää tietyt kriteerit. Kriteeriksi voidaan esimerkiksi asettaa tietty etäisyys kamerasta, jolloin eroa laadussa voi olla vaikea tai mahdoton havaita, ja saavutetaan säästöjä laitteistoresursseissa heikentämättä kuvanlaatua merkittävästi. Läpikäyntivarjostin sisältäisi logiikkaa, joka valitsee halutun LOD-tason, ja lähettää säteen kyseisen tason sisältävään BVH-instanssiin [40]. Tämä vaatisi BVH-rakennetta, jossa voi olla useita sisäkkäisiä tasoja perinteisen kahden tason (TLAS ja BLAS) sijaan. Tästä syystä Xe-HPG-arkkitehtuuri sisältää tuen kahdeksalle sisäkkäiselle BVH-tasolle [39].

4.3 Nvidia

Nvidia on julkaissut kolme grafiikkaprosessoriarkkitehtuuria, jotka sisältävät säteenseurantaa kiihdyttäviä elementtejä. Kyseiset arkkitehtuurit ovat Turing (julkaistu 2018) [20], Ampere (julkaistu 2020) [41] ja Ada Lovelace (julkaistu 2022) [26]. Nvidian arkkitehtuureissa jokainen SM (Streaming Multiprocessor) sisältää yhden RT-ytimen, jonka tehtävänä on kiihdyttää säteenseurantaa. RT-ytimet sisältävät erilliset yksiköt säteiden ja BVH-laatikoiden leikkauksien testaamiseen, sekä säteiden ja kolmioiden leikkauksien testaamiseen. RT-ytimet käyvät itsenäisesti koko BVH-rakenteen läpi ja palauttavan tiedon osumasta tai sen puutteesta SM:n käsiteltäväksi.

Ampere-arkkitehtuurin 2. sukupolven RT-ytimet kaksinkertaistavat kapasiteetin säteen ja kolmioiden leikkauksien testaamiselle Turing-arkkitehtuuriin verrattuna. Ampere-arkkitehtuuri mahdollistaa myös RT-ytimien hyödyntämisen samanaikaisesti joko grafiikka- tai laskentatehtävien prosessoinnin kanssa. [41]

Lisäksi 2. sukupolven RT-ytimissä on uusi yksikkö liike-epäterävyyden (engl. motion blur) luomiselle säteenseurannan avulla [41]. Liike-epäterävyyden tapauksessa kolmioita ei käsitellä staattisena kuvan renderöinnin aikana, vaan niille määritellään funktio, joka määrää niiden sijainnin tiettynä ajanhetkenä. Säteille annetaan aikaleimat, jonka mukaisella kolmion sijainnilla kyseisen säteen laskut suoritetaan. Edellä mainitun uuden yksikön tehtävä on kolmion sijainnin määritys, jonka tulos välitetään säde/kolmio-testausyksikölle leikkauksen testausta varten [41].

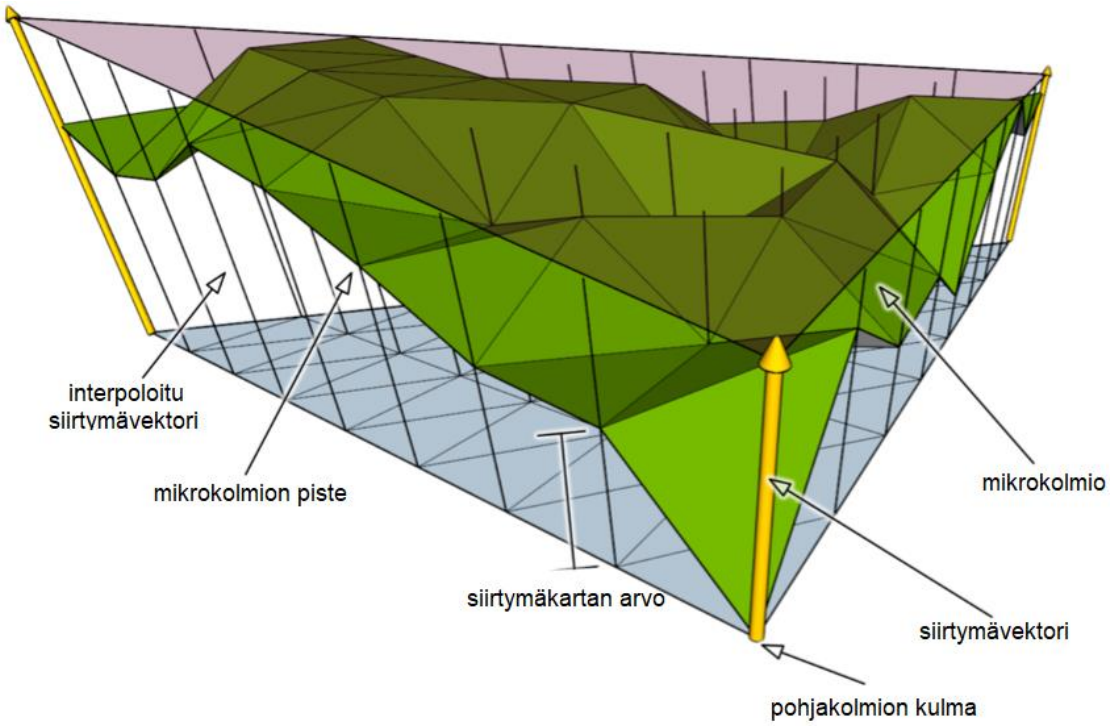
Ada Lovelace -arkkitehtuurin 3. sukupolven RT-ytimet jälleen kaksinkertaistavat säteen ja kolmioiden leikkauksien testauskapasiteetin edeltävään Ampere-arkkitehtuuriin verrattuna [26]. Lisäksi Ada Lovelace -arkkitehtuuri lisää ainakin kolme uutta ominaisuutta, jotka nopeuttavat säteenseurantaa.

Yksi näistä on RT-ytimissä sijaitseva Opacity Micromap Engine, jonka tehtävänä on vähentää tarvetta yleisosumavarjostimien suoritukselle läpinäkyviä kohtia sisältävän geometrian tapauksessa [26]. Yleisosumavarjostimen suoritus hidastaa säteen osumapisteen selvitystä merkittävästi, sillä RT-ydin joutuu lähettämään varjostimen SM:n suoritettavaksi ja odottamaan tuloksia. Opacity Micromap Engine mahdollistaa osumakohdan läpinäkyvyyden määrittämisen RT-ytimessä tietyissä tapauksissa hyödyntämällä läpinäkymättömyysmaskeja (engl. opacity mask) [26].

Läpinäkymättömyysmaski on kolmiolle etukäteen muodostettu maski, joka jakaa kolmion useisiin mikrokolmioihin [42]. Mikrokolmioilla on neljä mahdollista tilaa; läpinäkyvä, läpinäkymätön, tuntematon läpinäkyvä ja tuntematon läpinäkymätön [42]. Tuntemattomassa tilassa olevan mikrokolmion alueella on sekä läpinäkyviä, että läpinäkymättömiä kohtia. Kahden tuntemattoman tilan väliltä valitaan oletettavasti vertaamalla mikrokolmion läpinäkyvän ja läpinäkymättömän alueen pinta-aloja.

Opacity Micromap Engine selvittää, mihin läpinäkymättömyysmaskin mikrokolmioon säde osuu [26]. Läpinäkyvän mikrokolmion tapauksessa voidaan osuma jättää huomiotta, ja läpinäkymättömän mikrokolmion tapauksessa osuma voidaan raportoida [26]. Ainoastaan tuntemattoman mikrokolmion tapauksessa tarvitaan yleisosumavarjostinta. Vähemmän tarkkuutta vaativissa tilanteissa on mahdollista ohittaa yleisosumavarjostimen suoritus myös tuntemattomaksi merkittyjen mikrokolmioiden tapauksessa [42]. Tällöin läpinäkyvyys määräytyy sen perusteella, kumpi kahdesta tuntemattomasta tilasta mikrokolmiolla on. Nvidian mukaan Opacity Micromap Engine kaksinkertaistaa suorituskyvyn paljon läpinäkyvää geometriaa sisältävissä tilanteissa [26].

Toinen uusi ominaisuus 3. sukupolven RT-ytimissä on Displaced Micro-Mesh Engine. Displaced Micro-Mesh (DMM) on tekniikka, jonka avulla suuren määrän yksityiskohtia sisältävä 3D-malli voidaan esittää murto-osalla perinteisen tavan kolmiomäärästä korvaamalla puuttuvat yksityiskohdat siirtymäkartoilla (engl. displacement map) [26]. Jokaiseen yksinkertaistetun mallin kolmioon siis liittyy siirtymäkartta. Pohjakolmio ja siirtymäkartta yhdessä muodostavat eräänlaisen uuden primitiivityypin. Pohjakolmio jaetaan useiden mikrokolmioiden verkoksi, ja jokaista mikrokolmion kulmapistettä vastaa arvo siirtymäkartassa, jonka perusteella kulmapisteen lopullinen sijainti lasketaan. Pohjakolmion ja siirtymäkartan yhdistelmä muunnetaan lopulliseksi geometriaksi renderöintivaiheessa. Kuva 7 havainnollistaa DMM:n toimintaa.



Kuva 7. Micro-Mesh-primitiivi, muokattu lähteestä [43].

DMM:n hyödyntämiseen liittyy useita etuja; se pienentää 3D-mallin kokoa muistissa, yksinkertaistaa BVH:ta ja poistaa tarpeen säilyttää 3D-mallista useita eri LOD-versioita. Maggiodormo et al. [44] mukaan yksinkertaistetun BVH:n mahdollistama parannus perinteiseen tapaan verrattuna oli mediaaniltaan kuusinkertainen BVH:n koon osalta ja nelinkertainen BVH:n rakennusajan osalta. Lisäksi 3D-mallin koko muistissa pieneni keskimäärin viidestoistaosaan alkuperäisestä. Hintana tälle kuitenkin oli 1,3-kertainen hidastuminen varsinaisessa säteenseurantaosuudessa.

Kolmas Ada Lovelace -arkkitehtuurin lisäämä ominaisuus on Shader Execution Reordering eli SER, joka ryhmittelee suoritettavia varjostinohjelmia koherenteiksi joukoiksi [26]. Näin voidaan tehokkaammin hyödyntää grafiikkaprosessorin SIMD-yksiköitä niiden suorittamiseen. SER:än käyttäminen ja ohjaaminen tapahtuu erillisen rajapinnan kautta [45]. Nvidian mukaan SER mahdollistaa jopa kaksinkertaisen suorituskyvyn tietyissä tapauksissa [45].

5. KAUPALLISTEN RATKAISUJEN VERTAILU

Tämä luku vertailee luvussa 4 esiteltyjä arkkitehtuureita toisiinsa. Vertailu on jaettu kolmeen osaan: ominaisuudet, kiihdytysrakenne ja leikkaustestauskapasiteetti. Luvussa esitetty tieto perustuu lukuun 4, pois lukien joitakin täydennyksiä, joiden lähteet on merkitty erikseen.

5.1 Ominaisuudet

Taulukko 1 esittää eri grafiikkaprosessoriarkkitehtuurien toteuttamat luvussa 3.2 linjatut säteenseurantaominaisuudet. Uusimpia arkkitehtuureita verratessa AMD:n RDNA 3 toteuttaa selkeästi vähiten säteenseurannan laitteistokiihdytysominaisuuksia. Intelin Xe-HPG ja Nvidian Ada Lovelace toteuttavat samat perusominaisuudet, mutta näiden ulkopuolella Nvidialla on enemmän ominaisuuksia (liike-epäterävyysyksikkö, OMM ja DMM) kuin Intelillä (läpikäyntivarjostimet). Yksikään vertailun arkkitehtuureista ei toteuta BVH:n rakennukselle erillistä yksikköä. Nvidian DMM kuitenkin tarjoaa vaihtoehtoisen tavan nopeuttaa BVH:n rakentamista.

Taulukko 1. Grafiikkaprosessoriarkkitehtuurien säteenseurannan laitteistokiihdytyksen perusominaisuudet.

	AMD		Intel	Nvidia		
	RDNA 2	RDNA 3	Xe-HPG	Turing	Ampere	Ada Lovelace
Säde-laatikko-testaus	x	x	x	x	x	x
Säde-kolmio-testaus	x	x	x	x	x	x
BVH:n läpikäynti			x	x	x	x
Koherenssilajittelu			x			x
BVH:n rakennus						

Intel ja Nvidia molemmat toteuttavat ratkaisun koherenssiongelmaan varjostimien suorituksen osalta, mutta ratkaisuissa on kuitenkin joitakin eroja. Intelin arkkitehtuuri sisältää erillisen laitteistoyksikön, TSU:n, joka lajittelee varjostimia ja ohjaa niiden suoritusta. Nvidian SER taas vaikuttaisi olevan ainakin osittain ohjelmistoon pohjautuva ratkaisu; Nvidia kertoo Ada Lovelace -arkkitehtuurin olevan suunniteltu ”SER huomioon ottaen”, ja mainitsee erityisesti optimoinnit SM:ään sekä muistijärjestelmään [26]. SER:ää ohjataan erillisen rajapinnan kautta [26], kun taas Intelin TSU:ta hyödynnetään automaattisesti soveltuvia tehtäviä suoritettaessa.

Myös LOD-tekniikoiden osalta on arkkitehtuureilla merkittäviä eroja. AMD:n laitteisto ei erityisesti huomioi LOD:ta, sillä BVH:n läpikäynnin puuttumisen johdosta ohjelmoija kykenee hallitsemaan säteenseurantaprosessia lähes täysin, ja teoriassa voi siten vapaasti toteuttaa haluamansa LOD-tekniikan. Intelin ja Nvidian tapauksessa BVH-läpikäyntilaitteisto asettaa rajoitteita ohjelmoitaville LOD-tekniikoille, ja siksi molemmat ovat uusimmissa arkkitehtuureissaan erityisesti huomioineet LOD:n. Intel lisää LOD-tekniikoiden toteuttamiseen joustavuutta tukemalla läpikäyntivarjostimia, jotka mahdollistavat LOD-tason valinnan ohjelmallisesti BVH:n läpikäynnin aikana. Nvidia taas huomioi LOD:n osana DMM-tekniikkaa, joka poistaa tarpeen säilyttää 3D-mallista useampaa eri LOD-versiota. DMM ei kuitenkaan mahdollista LOD-tason määrittämistä kesken BVH:n läpikäynnin, kuten Intelin ratkaisu.

5.2 Kiihdytysrakenne

AMD käyttää BVH4-rakennetta eli BVH:ta, jossa laatikoita sisältävillä solmuilla on neljä lasta [21], [30]. Vastaavasti AMD:n Ray Accelerator -yksikkö suorittaa neljä säde-laatikko-testausoperaatiota kerralla eli käsittelee kaikki BVH-solmun lapset kerralla. AMD:n BVH:n lehtisolmut voivat ainakin sisältää yksi tai kaksi kolmiota, mutta viitteitä on laitteiston tuesta myös neljälle kolmiolle. Ray Accelerator -yksikkö suorittaa kuitenkin vain yhden säde-kolmio-testin kerrallaan, joten se voi joutua käsittelemään samaa lehtisolmua useamman kerran.

Intelin Traversal Pipelinen sisältämä kuusi säde-laatikko-testausyksikköä viittaa laitteiston olevan suunniteltu BVH6-rakennetta varten, eli laatikoita sisältävillä solmuilla olisi kuusi lasta. Intelin Vulkan-ajurien lähdekoodi vahvistaa tämän; BVH-solmua kuvaava tietorakenne sisältää dataa kuudelle lapsisolmulle [46]. Intel siis käyttää leveämpää BVH-rakennetta kuin AMD. Teoriassa tämä mahdollistaa säteen osumapisteen määrittämisen pienemmällä määrällä läpikäyntivaiheita, sillä jokaisella vaiheella on mahdollisuus poissulkea suurempi määrä mahdollisia osumakohtia. Toisaalta myös vaadittu säde-laatikko-testausyksiköiden määrä kasvaa. Intelin BVH:n lehtisolmut sisältävät kaksi viereistä kolmiota, tai vain yhden kolmion, jos tämä ei ole mahdollista. AMD:sta poiketen Intelin säde-kolmio-testausyksikkö vaikuttaisi käsittelevän molemmat kolmiot kerralla. Tästä huolimatta Intelin ratkaisu sisältää pienemmän määrän kolmiotestauskapasiteettia suhteessa laatikkotestauskapasiteettiin kuin AMD:n ratkaisu, sillä Intelin RTU:ssa kaksi Traversal Pipelinea jakavat yhden kolmiotestausyksikön.

Nvidian tapauksessa BVH:sta ja laitteiston yksityiskohdista on tarjolla melko vähän tietoa. Tiedetään kuitenkin, että kapasiteetti säde-kolmio-leikkaustestaukseen on nelinker-

taistunut RT-ydintä kohti, kun verrataan uusinta Ada Lovelace -arkkitehtuuria ensimmäiseen Turing-arkkitehtuuriin. Nvidia siis vaikuttaisi keskittyvän vahvasti kolmiotestaukseen, mikä saattaisi viitata BVH:hon, jonka lehdet sisältävät useampia kolmioita. Tähän viittaa myös Nvidian DMM-tekniikka. Ada Lovelace -arkkitehtuuria esittelevässä asiakirjassa mainitaan yksinkertaistettu BVH DMM-tekniikkaa käytettäessä [26]. Tässä yksinkertaistetussa BVH:ssa yksi lehtisolmu sisältää ainakin graafisen esityksen perusteella yhden pohjakolmion, joka voidaan jakaa useaksi mikrokolmioksi. Pohjakolmio on mahdollista jakaa jopa 1024 mikrokolmioksi [42], joten BVH:n lehtisolmu voi sisältää huomattavan määrän kolmioita, vaatien huomattavan määrän kolmiotestauskapasiteettia.

5.3 Leikkaustestauskapasiteetti

Tarkastelemalla grafiikkaprosessorin leikkaustestauskapasiteetin teoreettista maksimia voidaan muodostaa jonkinlainen käsitys grafiikkaprosessorin kyvystä löytää säteiden osumakohtia. On kuitenkin syytä huomioida, että maksimileikkaustestauskapasiteetti on vain yksi suorituskykyä määrittävä tekijä, ja ei välttämättä vastaa reaalimaailman suorituskykyä.

Taulukko 2 esittää eri grafiikkaprosessoriarkkitehtuurien leikkaustestauskapasiteetit. Kutakin luvussa 4 esiteltyä arkkitehtuuria kohden on valittu yksi grafiikkakortti, jonka grafiikkaprosessori sisältää kyseiselle arkkitehtuurille saatavilla olevan maksimikonfiguraation. Grafiikkakorttien perustiedot perustuvat TechPowerUpin grafiikkaprosessoritietokantaan [47]. Testauskapasiteetit on laskettu hyödyntäen tietoja testausyksiköiden määristä sekä grafiikkaprosessorien maksimikellotaajuuksista. Lisäksi Intelin tapauksessa on otettu huomioon kolmiotestausyksikön kyky testata kaksi komiota kerralla. Säteenseurantayksikkö viittaa AMD:n tapauksessa Ray Accelerator -yksikköön, Intelin tapauksessa RTU:hun ja Nvidian tapauksessa RT-ytimeen. Nvidian tapauksessa kolmiotestauskapasiteetti on suhteutettu Turing-arkkitehtuurin RT-ytimen kolmiotestauskapasiteettiin, jota on merkitty symbolilla x.

Taulukon 2 perusteella uusimpia arkkitehtuureita vertaillessa Intel yltää lähes AMD:n tasolle laatikkotestauskapasiteetissa huomattavasti pienemmästä transistorien määrästä huolimatta. Nvidian laatikkotestauskapasiteettia ei ole tiedossa. Kolmiotestauskapasiteetissa Intel taas jää melko huomattavasti AMD:sta jälkeen siitäkin huolimatta, että Intelin kolmiotestausyksikön on laskettu tekevän kaksi testiä kellojaksossa. Nvidian osalta absoluuttisia lukuja ei kolmiotestauksenkaan osalta ole, mutta Ampere- ja Ada Lovelace -arkkitehtuurien testauskapasiteetti voidaan suhteuttaa Turing-arkkitehtuurin kapasiteettiin hyödyntämällä tietoja grafiikkakorttien kellotaajuuksista ja RT-ytimien määristä, sekä luvussa 4.3 esitettyjä tietoja kolmiotestauskapasiteetin kasvusta arkkitehtuurien välillä.

Tällöin nähdään, että Turing-arkkitehtuuriin verrattuna Ampere-arkkitehtuuri on kolmiotestauskapasiteetiltaan lähes 2,5-kertainen, ja Ada Lovelace -arkkitehtuuri yli 11-kertainen.

Taulukko 2. Grafiikkaprosessoriarkkitehtuurien leikkaustestauskapasiteetin vertailu.

	AMD		Intel	Nvidia		
Arkkitehtuuri	RDNA 2	RDNA 3	Xe-HPG	Turing	Ampere	Ada Lovelace
Arkkitehtuurin julkaisuvuosi	2020	2022	2022	2018	2020	2022
Grafiikkakortin nimi	RX 6950 XT	RX 7900 XTX	Arc A770	Quadro RTX 8000	RTX 3090 Ti	RTX 6000 Ada
Grafiikkaprosessorin nimi	Navi 21	Navi 31	DG2-512	TU102	GA102	AD102
Pinta-ala	520 mm²	529 mm²	406 mm²	754 mm²	628 mm²	609 mm²
Transistorien määrä	26,8 mrd.	57,7 mrd.	21,7 mrd.	18,6 mrd.	28,3 mrd.	76,3 mrd.
Maksimikellotaajuus	2310 Mhz	2499 Mhz	2400 Mhz	1770 Mhz	1860 Mhz	2505 Mhz
Säteenseurantayksiköiden määrä	80	96	32	72	84	142
Laatikkotestauskapasiteetti (testiä/kellojakso)	320	384	384	?	?	?
Kolmiotestauskapasiteetti (testiä/kellojakso)	80	96	64	72x	168x	568x
Laatikkotestauskapasiteetti (testiä/s)	739 mrd.	960 mrd.	922 mrd.	?	?	?
Kolmiotestauskapasiteetti (testiä/s)	185 mrd.	240 mrd.	154 mrd.	(127 mrd.)x	(312 mrd.)x	(1420 mrd.)x

Vertailukelpoisten absoluuttisten kolmiotestauslukujen saamiseksi voitaisiin tehdä oletus, että Turing-arkkitehtuurin RT-ydin kykenee ainakin yhteen kolmiotestiin kellojaksoa kohden, jolloin taulukkoon 2 voitaisiin sijoittaa $x = 1$. Huomioiden muut tunnetut arkkitehtuurit, on melko epätodennäköistä, että Turing-arkkitehtuuri olisi tätä hitaampi. Oletuksen perusteella Turing-arkkitehtuurin kolmiotestauskapasiteetti olisi 127 mrd. testiä/s, Ampere-arkkitehtuurin 312 mrd. testiä/s ja Ada Lovelace -arkkitehtuurin 1420 mrd. testiä/s. Oletuksen perusteella Ada Lovelace -arkkitehtuurin kolmiotestauskapasiteetti olisi siis huomattavasti kilpailevia arkkitehtuureita suurempi: lähes kuusinkertainen RDNA3:een verrattuna, ja yli yhdeksänkertainen Xe-HPG-arkkitehtuuriin verrattuna. Tämä tulos tukee edellisessä aliluvussa esitettyä päätelmää, jonka mukaan Nvidia panostaa erittäin vahvasti kolmiotestaukseen.

6. YHTEENVETO

Työn tavoitteena oli selvittää, minkälaisia ratkaisuja säteenseurannan laitteistokiihdytykseen on olemassa ja miten ne eroavat toisistaan. Työ toteutettiin kirjallisuuskatsauksena. Ensin pohjustettiin aihetta esittelemällä säteenseurantaa ja sen kiihdytystä yleisesti. Seuraavaksi esiteltiin tarkasteluun valittujen valmistajien ratkaisuja, ja lopuksi verrattiin niitä keskenään ominaisuuksien, valitun kiihdytysrakenteen sekä leikkaustestauskapasiteetin osalta.

Työn tuloksena voidaan todeta, että eri valmistajat lähestyvät säteenseurannan laitteistokiihdytystä hieman eri tavoilla. AMD:n laitteisto kiihdyttää säteenseurantaa hyödyntämällä erillisiä leikkaustestausyksiköitä. Esimerkiksi kiihdytysrakenteen läpikäynnille ei ole erillisiä laitteistoyksiköitä. Tämän ratkaisun etuna on ohjelmoitavuus; kiihdytysrakenteen läpikäyntilogiikka voidaan toteuttaa ohjelmistotasolla, mikä mahdollistaa esimerkiksi parempien LOD-tekniikoiden hyödyntämisen.

Intel toteuttaa huomattavasti suuremman määrän säteenseurantaominaisuuksia laitteistotasolla. Intelin ratkaisu sisältää leikkaustestausyksiköt, BVH:n läpikäynnin sekä varjostinohjelmien suoritusjärjestystä optimoivan yksikön. Leikkaustestauskapasiteetin osalta AMD:n RDNA 3:een verrattuna Intelin Xe-HPG kykenee melko samanlaiseen määrään laatikkotestauksia, mutta jää kolmiotestauskapasiteetissa jälkeen. Xe-HPG kuitenkin kykenee tähän huomattavasti pienemmällä transistorimäärällä, mikä osaltaan viittaa sen painottavan säteenseurantaa enemmän kuin AMD:n arkkitehtuurit.

Myös Nvidian uusin laitteisto sisältää leikkaustestausyksiköt, BVH:n läpikäynnin sekä varjostinohjelmien suoritusjärjestyksen optimoinnin. Näiden lisäksi se sisältää ominaisuuksia läpinäkyvyystestauksen ja BVH:n rakentamisen nopeuttamiseen. Joidenkin Nvidian laitteiston ominaisuuksien hyödyntäminen vaatii erillistä huomiota kehittäjiltä sekä erillisten rajapintojen käyttöä. Varmoja tietoja Nvidian arkkitehtuurien leikkaustestauskapasiteeteista ei ole saatavilla, mutta Nvidian voidaan todeta keskittyvän vahvasti kolmiotestaukseen. Nvidian Ada Lovelace -arkkitehtuurin kolmiotestauskapasiteetti on luvussa 5.2 esitetyn päätelmän mukaan moninkertaisesti kilpailijoitaan suurempi.

Reaaliaikainen säteenseuranta on vielä melko varhaisessa kehitysvaiheessa ja siihen liittyvä laitteisto kehittyy nopeasti. Kaikkien työssä käsiteltyjen valmistajien arkkitehtuureissa on valmiuksia ominaisuuksille, joita reaaliaikaisessa säteenseurannassa yleisesti hyödynnettävät rajapinnat eivät vielä tue, ja eri arkkitehtuurien välillä on melko suuria

eroja ominaisuuksissa. On todennäköistä, että säteenseurantalaitteiston kehittyessä pidemmälle etenkin erityistä huomioita kehittäjiltä vaativat ominaisuudet yhdenmukaistuvat kilpailevien arkkitehtuurien välillä, jotta saavutetaan suurempi yhteensopivuusaste ohjelmakoodin ja eri laitteistoratkaisujen välillä.

LÄHTEET

[1] T. Whitted, ”An improved illumination model for shaded display”, vsk. 23, nro 6, 1980, [Verkossa]. Saatavissa: https://dl.acm.org/doi/pdf/10.1145/358876.358882

[2] W. Mueller ja F. Ullmann, ”A scalable system for 3D audio ray tracing”, teoksessa *Proceedings IEEE International Conference on Multimedia Computing and Systems*, Florence, Italy: IEEE Comput. Soc, 1999, ss. 819–823. doi: 10.1109/MMCS.1999.778592.

[3] J. T. Kajiya, ”THE RENDERING EQUATION”, vsk. 20, nro 4, 1986.

[4] C. Barré-Brisebois *ym.*, ”Hybrid Rendering for Real-Time Ray Tracing”, teoksessa *Ray Tracing Gems*, E. Haines ja T. Akenine-Möller, Toim., Berkeley, CA: Apress, 2019, ss. 437–473. doi: 10.1007/978-1-4842-4427-2_25.

[5] J. Pineda, ”A Parallel Algorithm for Polygon Rasterization”, 1988.

[6] ”File:Ray trace diagram.svg - Wikipedia”. Viitattu: 29. marraskuuta 2023. [Verkossa]. Saatavissa: https://commons.wikimedia.org/wiki/File:Ray_trace_diagram.svg

[7] P. Walewski, T. Gałaj, ja D. Szajerman, ”Heuristic based real-time hybrid rendering with the use of rasterization and ray tracing method”, *Open Phys.*, vsk. 17, nro 1, ss. 527–544, loka 2019, doi: 10.1515/phys-2019-0055.

[8] Microsoft Corporation, ”DirectX Raytracing”. Microsoft Corporation. Viitattu: 6. marraskuuta 2023. [Verkossa]. Saatavissa: https://microsoft.github.io/DirectX-Specs/d3d/Raytracing.html

[9] Khronos Group, ”Vulkan”. Khronos Group. Viitattu: 31. joulukuuta 2023. [Verkossa]. Saatavissa: https://registry.khronos.org/vulkan/

[10] W. Usher, ”The Shader Binding Table Demystified”, teoksessa *Ray Tracing Gems II: Next Generation Real-Time Rendering with DXR, Vulkan, and OptiX*, A. Marrs, P. Shirley, ja I. Wald, Toim., Berkeley, CA: Apress, 2021, ss. 193–211. doi: 10.1007/978-1-4842-7185-8_15.

[11] S. Fenney ja A. Ozkan, ”Compressed Opacity Maps for Ray Tracing”, s. 1, 2023, doi: 10.2312/HPG.20231133.

[12] A. Breglia, A. Capozzoli, C. Curcio, ja A. Liseno, ”Comparison of Acceleration Data Structures for Electromagnetic Ray-Tracing Purposes on GPUs [EM Programmer’s Notebook]”, *IEEE Antennas Propag. Mag.*, vsk. 57, nro 5, ss. 159–176, loka 2015, doi: 10.1109/MAP.2015.2470685.

[13] D. Meister, S. Ogaki, C. Benthin, M. J. Doyle, M. Guthe, ja J. Bittner, ”A Survey on Bounding Volume Hierarchies for Ray Tracing”, *Comput. Graph. Forum*, vsk. 40, nro 2, ss. 683–712, touko 2021, doi: 10.1111/cgf.142662.

[14] ”Thinking Parallel, Part II: Tree Traversal on the GPU”, NVIDIA Technical Blog. Viitattu: 19. lokakuuta 2023. [Verkossa]. Saatavissa: https://developer.nvidia.com/blog/thinking-parallel-part-ii-tree-traversal-gpu/

[15] C. Benthin, S. Woop, I. Wald, ja A. T. Áfra, ”Improved two-level BVHs using partial re-braiding”, teoksessa *Proceedings of High Performance Graphics*, Los Angeles California: ACM, heinä 2017, ss. 1–8. doi: 10.1145/3105762.3105776.

[16] M. Saed, Y. H. Chou, L. Liu, T. Nowicki, ja T. M. Aamodt, ”Vulkan-Sim: A GPU Architecture Simulator for Ray Tracing”, teoksessa *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Chicago, IL, USA: IEEE, loka 2022, ss. 263–281. doi: 10.1109/MICRO56248.2022.00027.

[17] Khronos Group, ”Vulkan Documentation - VkBuildAccelerationStructureFlagBitsKHR(3)”. Viitattu: 6. marraskuuta 2023. [Verkossa]. Saatavissa: https://registry.khronos.org/vulkan/specs/1.3-extensions/man/html/VkBuildAccelerationStructureFlagBitsKHR.html

[18] Microsoft Corporation, ”DirectX Raytracing (DXR) Functional Spec”, DirectX-Specs. Viitattu: 6. marraskuuta 2023. [Verkossa]. Saatavissa: https://microsoft.github.io/DirectX-Specs/d3d/Raytracing.html
[19] K. Beets, ”RAY TRACING FOR THE MASSES”, [Verkossa]. Saatavissa: https://www.imaginationtech.com/products/gpu/img-dxt-gpu/
[20] NVIDIA Corporation, ”NVIDIA Turing GPU Architecture”. 2018. [Verkossa]. Saatavissa: https://images.nvidia.com/aem-dam/en-zz/Solutions/design-visualization/technologies/turing-architecture/NVIDIA-Turing-Architecture-Whitepaper.pdf
[21] Advanced Micro Devices Inc., ”’RDNA 2’ Instruction Set Architecture: Reference Guide”, [Verkossa]. Saatavissa: https://www.amd.com/content/dam/amd/en/documents/radeon-tech-docs/instruction-set-architectures/rdna2-shader-instruction-set-architecture.pdf
[22] Epic Games Inc., ”Lumen Performance Guide”. Viitattu: 7. tammikuuta 2024. [Verkossa]. Saatavissa: https://docs.unrealengine.com/5.2/en-US/lumen-performance-guide-for-unreal-engine/
[23] T. Viitanen, M. Koskela, P. Jääskeläinen, A. Tervo, ja J. Takala, ”PLOCTree: A Fast, High-Quality Hardware BVH Builder”, *Proc. ACM Comput. Graph. Interact. Tech.*, vsk. 1, nro 2, ss. 1–19, elo 2018, doi: 10.1145/3233309.
[24] Y. Deng, Y. Ni, Z. Li, S. Mu, ja W. Zhang, ”Toward Real-Time Ray Tracing: A Survey on Hardware Acceleration and Microarchitecture Techniques”, *ACM Comput. Surv.*, vsk. 50, nro 4, ss. 1–41, heinä 2018, doi: 10.1145/3104067.
[25] Intel Corporation, ”Intel® Arc™ Graphics Developer Guide for Real-Time Ray Tracing in...”, Intel. Viitattu: 10. lokakuuta 2023. [Verkossa]. Saatavissa: https://www.intel.com/content/www/us/en/developer/articles/guide/real-time-ray-tracing-in-games.html
[26] NVIDIA Corporation, ”NVIDIA Ada GPU Arcitecture”. 2023. [Verkossa]. Saatavissa: https://images.nvidia.com/aem-dam/Solutions/Data-Center/l4/nvidia-ada-gpu-architecture-whitepaper-v2.1.pdf
[27] Apple Inc., ”Explore GPU advancements in M3 and A17 Pro - Tech Talks - Videos”, Apple Developer. Viitattu: 20. joulukuuta 2023. [Verkossa]. Saatavissa: https://developer.apple.com/videos/play/tech-talks/111375/
[28] Arm Holdings PLC, ”Arm Immortalis-G715 and Arm Mali-G715 Performance Counters Reference Guide”. Viitattu: 20. joulukuuta 2023. [Verkossa]. Saatavissa: https://developer.arm.com/documentation/107776/0105/Shader-core-ray-tracing-unit?lang=en
[29] Qualcomm Inc., ”Snapdragon 8 Gen 2 Mobile Platform”. Viitattu: 20. joulukuuta 2023. [Verkossa]. Saatavissa: https://www.qualcomm.com/products/mobile/snapdragon/smartphones/snapdragon-8-series-mobile-platforms/snapdragon-8-gen-2-mobile-platform
[30] Advanced Micro Devices Inc., ”’RDNA3’ Instruction Set Architecture: Reference Guide”, [Verkossa]. Saatavissa: https://www.amd.com/content/dam/amd/en/documents/radeon-tech-docs/instruction-set-architectures/rdna3-shader-instruction-set-architecture-feb-2023_0.pdf
[31] ”GPUOpen-Drivers/AMDVLK”. GPUOpen Drivers, 24. joulukuuta 2023. Viitattu: 30. joulukuuta 2023. [Verkossa]. Saatavissa: https://github.com/GPUOpen-Drivers/AMDVLK
[32] ”gpurt/src/shaders/PairCompression.hlsl at 86360020040cbcf69a93addf1a53260ac6e2632b · GPUOpen-Drivers/gpurt”. Advanced Micro Devices Inc. Viitattu: 30. joulukuuta 2023. [Verkossa]. Saatavissa: https://github.com/GPUOpen-Drivers/gpurt/blob/86360020040cbcf69a93addf1a53260ac6e2632b/src/shaders/PairCompression.hlsl
[33] ”gpurt/src/shared/rayTracingDefs.h at 86360020040cbcf69a93addf1a53260ac6e2632b · GPUOpen-Drivers/gpurt”. Advanced Micro Devices Inc. Viitattu: 30. joulukuuta 2023. [Verkossa]. Saatavissa:

https://github.com/GPUOpen-Drivers/gpurt/blob/86360020040cbcf69a93addf1a53260ac6e2632b/src/shared/rayTracingDefs.h
[34]”gpurt/src/shaders/Common.hlsl at 86360020040cbcf69a93addf1a53260ac6e2632b · GPUOpen-Drivers/gpurt”. Advanced Micro Devices Inc. Viitattu: 30. joulukuuta 2023. [Verkossa]. Saatavissa: https://github.com/GPUOpen-Drivers/gpurt/blob/86360020040cbcf69a93addf1a53260ac6e2632b/src/shaders/Common.hlsl
[35]Advanced Micro Devices Inc., ”AMD RDNA 3 -kalvosetti”, 2022.
[36]J. Silva, ”AMD shares additional details on the RDNA 3 architecture | KitGuru”. Viitattu: 21. joulukuuta 2023. [Verkossa]. Saatavissa: https://www.kitguru.net/components/graphic-cards/joao-silva/amd-shares-additional-details-on-the-rdna-3-architecture/
[37]Intel Corporation, ”Introduction to the Xe-HPG Architecture”, Intel. Viitattu: 7. tammikuuta 2024. [Verkossa]. Saatavissa: https://www.intel.com/content/www/us/en/developer/articles/technical/introduction-to-the-xe-hpg-architecture.html
[38]*Intel Arc A-Series Graphics Ray Tracing Technology Deep Dive*, (2. syyskuuta 2022). Viitattu: 10. joulukuuta 2023. [Verkossa Video]. Saatavissa: https://www.youtube.com/watch?v=J5eIOv-CrB8
[39]*A Quick Guide to Intel’s Ray-Tracing Hardware | GDC 2022 | Intel Software*, (27. huhtikuuta 2022). Viitattu: 10. lokakuuta 2023. [Verkossa Video]. Saatavissa: https://www.youtube.com/watch?v=SA1yvWs3lHU
[40]W.-J. Lee, G. Liktor, ja K. Vaidyanathan, ”Flexible Ray Traversal with an Extended Programming Model”, teoksessa *SIGGRAPH Asia 2019 Technical Briefs*, Brisbane QLD Australia: ACM, marras 2019, ss. 17–20. doi: 10.1145/3355088.3365149.
[41]NVIDIA Corporation, ”NVIDIA Ampere GA102 GPU Architecture”. 2020. [Verkossa]. Saatavissa: https://www.nvidia.com/content/PDF/nvidia-ampere-ga-102-gpu-architecture-whitepaper-v2.pdf
[42]NVIDIA Corporation, ”Micro-Mesh - Basics”, [Verkossa]. Saatavissa: https://developer.download.nvidia.com/ProGraphics/nvpro-samples/slides/Micro-Mesh_Basics.pdf
[43]NVIDIA Corporation, ”NVIDIA Displacement Micromap Python Toolkit”. Viitattu: 3. tammikuuta 2024. [Verkossa]. Saatavissa: https://github.com/NVIDIAGameWorks/Displacement-MicroMap-Toolkit/blob/main/micromesh_python/notebook/micromesh.ipynb
[44]A. Maggiordomo, H. Moreton, ja M. Tarini, ”Micro-Mesh Construction”, *ACM Trans. Graph.*, vsk. 42, nro 4, ss. 1–18, elo 2023, doi: 10.1145/3592440.
[45]NVIDIA Corporation, ”Shader Execution Reordering”. 2022. Viitattu: 1. lokakuuta 2023. [Verkossa]. Saatavissa: https://developer.nvidia.com/sites/default/files/akamai/gameworks/ser-whitepaper.pdf
[46]”mesa/src/intel/vulkan/grl/gpu/qbvh6.h at 957009978ef6d7121fc0d710d03bc20097d4d46b · Mesa3D/mesa”. Intel Corporation. Viitattu: 4. tammikuuta 2024. [Verkossa]. Saatavissa: https://github.com/Mesa3D/mesa/blob/957009978ef6d7121fc0d710d03bc20097d4d46b/src/intel/vulkan/grl/gpu/qbvh6.h
[47]”TechPowerUp GPU database”, TechPowerUp. Viitattu: 12. joulukuuta 2023. [Verkossa]. Saatavissa: https://www.techpowerup.com/gpu-specs/