UNIUNEA EUROPEANĂ GUVERNUL ROMÂNIEI Fondul Social European POSDRU 2007-2013 Instrumente Structurale 2007-2013 MINISTERUL EDUCAȚIEI NAȚIONALE OIPOSDRU Universitatea Politehnica din Bucuresti

FONDUL SOCIAL EUROPEAN
Investeşte în oameni!
Programul Operaţional Sectorial pentru Dezvoltarea Resurselor Umane 2007 – 2013. Proiect POSDRU/159/1.5/S/134398
Dezvoltarea resurselor umane din cercetarea doctorală şi postdoctorală: motor al societății bazată pe cunoaștere -KNOWLEDGE

UNIVERSITATEA **POLITEHNICA** DIN BUCUREŞTI

Facultatea Automatică şi Calculatoare

REZUMATUL TEZEI DE DOCTORAT

Redarea de scene masive în timp real

Rendering massive scenes în real-time

Autor:
Msc. Ing. Alexandru Lucian Petrescu

Conducător Ştiinţific
Prof. Dr. Ing. Florica Moldoveanu

Bucureşti 2015

ABSTRACT:

În această teză este propusă o bandă grafică nouă, care poate lucra cu un număr mare de traiectorii de lumină în timp real, folosind tehnici inovative care decuplează procesul de redare în mai multe componente modulare şi cu consum de memorie eficient. Sunt prezentate mai multe contribuţii şi comparate cu algoritmii de ultimă oră, care sunt sau pot fi folosiţi în contextul redării în timp real a imaginilor de sinteză. Aceasta bandă grafică originală poate reda atât imagini de înaltă calitate în timp real cât şi imagini foto realiste în mod interactiv. Contribuţiile acestei teze sunt împărțite în metode de redare geometrice şi metode de iluminare. Algoritmii de redare geometrică modifică metode de rasterizare de ultimă oră în scopul redării mai eficiente a scenelor mari, prin decuplarea bandwidth-ului (rata de access la date) non geometric, provenit din date de texturare de calculele geometrice. Aceasta abordare reduce drastic bandwidth-ul consumat şi este implementata atât pentru obiecte opace cât şi pentru obiecte transparente, în algoritmi de redare noi, numiți „Virtual Deferred" şi „Virtual Order Independent Transparency". Aceste metode de redare sunt integrate într-un sistem de texturare virtuală. Imaginea obiectelor opace rasterizate este îmbunătăţită cu un nou algoritm de anti-aliasing, bazat pe reconstrucție sub geometrică, specializat pentru algoritmii de tip „Deferred", care îmbunătățește metodele de anti-aliasing de ultimă oră. Obiectele transparente sunt iluminate şi texturate adaptiv, îmbunătățind astfel complexitatea de redare a obiectelor transparente a algoritmilor de rasterizare ultimă oră.

Această teză introduce atât metode de iluminare exacte cât şi aproximative. Metodele aproximative decuplează frecvenţele înalte de frecvenţele joase în transportul de lumina, calculându-le pe fiecare în mod diferit. „Conservative Inexact Voxelization" este o metodă nouă de voxelizare, cu o complexitate de construcţie mult mai bună decât complexităţile folosite de metodele de ultimă oră şi care beneficiază de informația stocată de algoritmii de tip „Deferred". Această metodă de voxelizare e folosită pentru a reduce complexitatea operatorului de vizibilitate din ecuaţia de redare, funcționând ca o structură de determinare de vizibilitate pentru iluminare cu lumini virtuale. Operatorul de vizibilitate introdus cu această structură de date este adaptiv din punct de vedere al percepției, fiind aproape exact în volumul de vizualizare şi inexact în afara lui. Metodele corecte de iluminare generează imagini cu algoritmul „Path Tracing Bidirecțional", care foloseşte un nou tip de eșantionare de importanţa. Algoritmul „Light Flux" de eșantionare de importanţa introduce o nouă structura de date, prin care traiectoriile neproductive trasate de la cameră spre scenă pot fi legate rapid de vârfurile din traiectoriile de lumină, scăzând astfel timpul de redare al algoritmului „Path Tracing Bidirecțional".

Cuvinte cheie: redare în timp real, decuplare, reducere bandwidth, scene complexe, iluminare globală

LISTA DE PUBLICATII ŞI PROIECTE DE CERCETARE:

Lucian Petrescu, Anca Morar, Florica Moldoveanu, Victor Asavei, "Real time reconstruction of volumes from very large datasets using CUDA", în the 15th International Conference on System Theory, Control, and Computing (ICSTCC), 14-16 Octombrie, pp 1-5, ISBN: 978-1-4577-1173-2, 2011

Anca Morar, Florica Moldoveanu, Victor Asavei, Lucian Petrescu, Alin Moldoveanu, Alexandru Egner, "GPGPU Based Non-photorealistic Rendering of Volume Data", în Control Engineering and Applied Informatics (CEAI), vol.15, no. 1, pp. 45-52, 2013

Lucian Petrescu, Moldoveanu Florica, Moldoveanu Alin, Morar Anca, Asavei Victor, "Efficient Picking Through Atomic Operations", 19th International Conference on Control Systems and Computer Science (CSCS), pp 66-70, ISBN: 978-1-4673-6140-8, 29-31 Mai 2013 2013

Lucian Petrescu, Moldoveanu Florica, Victor Asavei, Moldoveanu Alin, Oana Ferche, "A GPU Task Generator for Rendering" în ICSTCC 2014 - 18th International Conference On System Theory, Control and Computing, pp. 562-567, ISBN: 978-1-4799-4602-0, Octombrie, 2014.

Lucian Petrescu, Florica Moldoveanu, Victor Asavei, Alin Moldoveanu, "Analyzing Deferred Rendering Techniques", în Control Engineering and Applied Informatics (CEAI), acceptat, în proces de publicare.

Lucian Petrescu, Florica Moldoveanu, Victor Asavei, Alin Moldoveanu, "Virtual deferred rendering", în 20th International Conference on Control Systems and Computer Science (CSCS), pp 373-378, ISBN 978-1-4799-1780-8, DOI 10.1109, 27-29 Mai, 2015.

Lucian Petrescu, Florica Moldoveanu, Victor Asavei, Alin Moldoveanu, "Guarded Order Independent Transparency", The Scientific Bulletin of University POLITEHNICA of Bucharest, Series C, Electrical Engineering and Computer Science, Vol. 77, Iss. 1, ISSN 2286-3540, Aprilie, 2015 .

Proiectul „Personalized implants for hip arthroplasty" (SABIMAS, PNCDII-Joint Applied Research Projects, 2008-2011). http://se.cs.pub.ro/SABIMAS/.

CONŢINUT

1. INTRODUCERE

1.1. Motivație

Grafica pe calculator şi vizualizarea sunt domenii stabilite în ştiința calculatoarelor. Scopul lor principal este analiza, sinteza şi manipularea datelor vizuale, concentrate pe aspecte matematice şi computaţionale în loc pe cele estetice. Redarea în timp real este un subiect specializat în domeniul graficii pe calculator, care încearcă să recreeze rezultatele vizuale din grafica pe calculator, doar că prioritizează interactivitatea şi are scopul final de a menține iluzia unei realități virtuale continue. Oamenii percep realitatea din jurul lor ca o secvență de imagini care sunt reconstruite inconștient de către creier într-o singură mișcare. Procesul acesta poate fi simulat în redarea în timp real şi poate să creeze o iluzie convingătoare a realității, cu un număr de cadre suficient de mare, de obicei mai mare de 60. Scenele masive conțin un număr foarte mare de obiecte ce sunt distribuite inegal din punct de vedere spațial şi care variază în dimensiuni şi proprietăți. Aceste scene sunt uzuale în redarea în timp real pentru că ele descriu vederi comune din viața reală.

Scopul principal al acestei teze este acela de a introduce o noua bandă de redare pentru scene masive, care folosește rasterizarea şi care minimizează memoria şi rata accesului la memorie, oferind performantă de rulare stabilă şi o decuplare puternica a algoritmilor utilizați.

Cu toata că ecuația de redare este separabilă, algoritmii de redare sunt de obicei monolitici şi suferă de un nivel ridicat de cuplare, de aceea ei pot fi greu de menținut şi şi mai greu de analizat. Găsirea de noi metode de decuplare a proceselor şi metodelor utilizate în banda de redare poate oferi noi oportunități în proiectarea algoritmilor. Rezolvarea cuplării ridicate şi a costurilor mari de mentenanţă implicate de aceasta reprezintă unul dintre motivele tezei, căutarea şi găsirea de metode prin care se creste modularitatea între multele componente utilizate în generarea imaginii finale din redarea în timp real. Stabilitatea numărului de cadre pe secunda este la fel de importanţă ca viteza procesului de redare. Variațiile de performanţă sunt de obicei cauzate de operații infrecvente ca streaming-ul de date şi conversia lor, şi reprezintă de asemenea un motiv de cercetare.

Mărimea spațiului de stocare, eficienta accesului la acesta şi consumul de bandwidth sunt în general aspectele de performanţă cele mai importante în redarea în timp real. Deoarece puterea de procesare a GPU-urilor creste cu o rată semnificativ mai mare decât cea a eficientei memoriei, importanţa utilizării judicioase a memoriei va continua să crească. Un obiectiv al acestei teze este de a oferi algoritmi îmbunătățiți care minimizează spațiul de stocare şi bandwidth-ul pentru redarea de obiecte transparente sau opace. Cu toate că deja exista un număr foarte mare de lucrări ştiințifice pe tema rasterizarii de obiecte opace, încă există oportunitatea de a îmbunătăți utilizarea resurselor şi de a decupla algoritmii folosiți în benzile de redare de ultimă oră.

Iluminarea globală (IG) pentru redarea în timp real rămâne o problemă nerezolvată. Există multe clase de algoritmi ce încearcă să rezolve problema IG: bazați pe fotoni, pe multe lumini, pe raze, pe path-uri (spații definite pe seturi de raze) sau pe cache-ul de iradianta. În pofida acestei multitudini de soluții, aplicațiile acestor algoritmi în timp real sunt reduse, de aceea necesitatea pentru IG rapid, inexact nu a fost satisfăcută. Unul din scopurile acestei teze este de a oferi noi algoritmi pentru IG.

1.2. Contribuții

Această teză introduce o nouă bandă de redare bazată pe rasterizare, care are un grad mare de decuplare şi o utilizare eficienta a memoriei. Această bandă de redare decuplează submisiile de redare de submisiile de stare, determinarea de vizibilitate de citirea texturilor, citirea texturilor de colorare şi post procesarea de citirea texturilor. Banda de redare prezentata conține multe contribuții, ce variază de la îmbunătățirea algoritmilor existenți la metode de redare complet noi. Comparata cu benzile de redare de ultimă oră bazate pe rasterizare, banda de redare prezentată este semnificativ mai eficientă în a rezolva probleme de redare cu bandwidth ridicat, ca probleme ce apar în redarea de scene masive.

Teza introduce un nou algoritm pentru redarea de obiecte opace. „Deferred Virtual" este un nou tip de algoritm „Deferred", ce combina proprietățile algoritmilor de tip „Deferred" cu o singură redare de geometrie cu proprietățile texturării virtuale, pentru a obține cele mai bune metrici pentru probleme de redare de bandwidth şi complexitate ridicată. „Deferred Virtual" folosește texturarea virtuala pentru a face streaming de date, care este necesară pentru toate desenările de scene masive. Principalele contribuții ale algoritmului „Deferred Virtual" sunt spațiul de stocare mic si rata scăzut de acces la memorie pentru redarea de scene masive cu obiecte complexe. O noua metoda de anti aliasing este introdusă, care este compatibila cu „Deferred Virtual". Aceasta metoda de anti-aliasing decuplează procesul de determinare de vizibilitate de procesul de colorare, şi, comparată cu metodele de ultimă oră, necesită o rata redusă de acces la memorie. Metoda de antialiasing prezentată nu este morfologica şi nu suferă astfel de artefacte temporale, neavând nevoie de reproiecţie temporală în calcularea rezultatului final. Comparat cu metoda de ultimă oră SRAA, algoritmul prezentat folosește o metoda de legare la vecini mai exactă decât filtrul bilateral folosit în SRAA.

Etapa de colorare nu folosește rasterizare şi este un proces complet executat GPGPU. Toate citirile de texturi sunt executate folosind cache-ul definit de grupul de lucru GPGPU. Această etapă este urmată de o etapă opțională de post procesare, în care kernelul de post procesare beneficiază de cache-ul grupului de lucru GPGPU.

Redarea corectă de obiecte transparente este în mod special dificilă pentru metodele de redare în timp real bazate pe rasterizare, deoarece modul în care rasterizarea oferă interacțiunile dintre obiecte şi cameră nu este ordonat după axa de adâncime. De aceea metodele de rasterizare necesită procese de ordonare pe fiecare pixel, în algoritmi ce consumă foarte mult spațiu de stocare şi bandwidth. Teza introduce algoritmul de Transparenţa Independenta de Ordine Virtuala (VOIT), care modifică algoritmul de ultimă oră „A-Buffer". Asemănător cu algoritmul „Deferred Virtual", VOIT folosește texturare virtuală şi are scopul de a scădea spațiul de stocare şi bandwidth-ul, mai ales pentru scene cu multe obiecte complexe. Comparat cu soluțiile deja existente, performantele VOIT scalează mult mai bine cu complexitatea scenei. În afara de VOIT, teza mai prezintă un algoritm rapid şi aproximativ pentru rasterizarea de obiecte transparente, ce modifica algoritmul de ultimă oră bazat pe harți de ocupare. Metoda propusă îmbunătățește algoritmul de hărți de ocupare prin adăugarea unei distribuții de adâncime fiecărui pixel. Aceasta distribuție de adâncime este folosită pentru a ajusta distribuțiile eşantioanelor din harta de ocupare, crescând astfel virtual rezoluția hărții şi astfel crescând calitatea aproximării funcției de opacitate definite peste harta de ocupare. Rezultatul acestei modificări este un rezultat final de o calitate superioară, ce utilizează aceleași resurse ca algoritmul original. Metoda prezentată

este astfel mult mai potrivită pentru redarea scenelor cu distribuții neuniforme de adâncime, așa cum sunt marea majoritate a scenelor masive.

Voxelizarea Inexactă Conservativă (CIV) este un nou algoritm de voxelizare inexactă, care scade complexitatea operaţiei de voxelizare de la numărul de primitive din scenă la numărul de obiecte. Acest rezultat imperfect este obținut prin subdivizarea rapida a volumelor încadratoare de forma cuboidă, care sunt stocate într-o reprezentare pe voxeli ierarhica a scenei. Un proces bazat pe algoritmul „Push-Pull" este folosit pentru a împrăștia informaţia pe toate nivelurile structurii ierarhice. CIV este o metoda inexacta de voxelizare cu scopul de a oferi informaţie aproximativa dar conservativa asupra structurii geometrice a scenei. CIV oferă informaţii suficient de exacte pentru diferite operaţii de determinare de vizibilitate. Această proprietate e folosita pentru a modifica ecuaţia de redare, schimbând operatorul exact de vizibilitate într-unul inexact dar conservativ. Acest operator inexact este apoi folosit într-o metodă bazată pe radiozitate instantanee, care creează un număr mare de lumini virtuale, folosind mai multe drumuri aleatoare prin structura bazata pe voxeli a scenei. După ce un număr suficient de mare de lumini virtuale a fost generat, scena este iluminată cu acestea, într-un proces analog algoritmilor „Deferred". Comparat cu metodele de ultimă oră de voxelizare, algoritmul prezentat permite transportul rapid al luminii de frecvenţă joasă fără nici un fel de calcule a priori sau cazuri speciale pentru obiecte animate. Cu toate că metoda prezentată este construită pentru transportul de lumină de joasa frecvenţă, ea poate fi folosită pentru a augmenta rezultatele metodelor aproximative de transport de lumină de înalta frecvenţă.

Familia de algoritmi „Path Tracing" se apropie de limita de viteză a redării interactive, şi cu toate că încă nu este folosibilă pentru redare în timp real, a atins un punct de relevanţă pentru domeniul tezei. Teza introduce o bandă alternativă de redare, ce redă imagini fotorealiste cu algoritmul „Path Tracing Bidirecțional", ce folosește un nou tip de eșantionare de importanţă numit „Light Flux Importance Sampling" (LFIS). LFIS este folosit pentru a ghida path-urile neproductive către surse de lumină care sunt foarte probabil vizibile. Astfel, procesul de redare converge semnificativ mai repede. Algoritmul „Path Tracing Bidirecțional" mai folosește voxelizarea inexactă conservativă pentru a amortiza complexitatea operației de determinare de vizibilitate. Astfel, banda de redare interactivă prezentată obține imagini fotorealiste mai repede decât metodele de ultimă oră.

De asemenea teza introduce mai mulți alți algoritmi, ce îmbunătățesc diferite aspecte ale procesului de redare: impostori ierarhici, generare de sarcini pe GPU, noi metode de decupare şi selecție de geometrie.

1.3. Structura Tezei

Teza este împărțită în cinci capitole: Introducere, State of the Art, Procesare de Geometrie, Iluminare şi Concluzii. Capitolul de introducere prezintă motivația, domeniul şi contribuțiile tezei. Capitolul de State of the art prezintă metode de ultimă oră folosite in redarea in timp real. Capitolul de Procesare de Geometrie prezintă diferiți algoritmi conceputi de autor, care sunt folosiți în rasterizarea de obiecte opace sau transparente. Capitolul de Iluminare discută algoritmii de transport de lumina de înaltă şi joasă frecvenţă, cât şi metodele de redare ce funcționează interactiv dar nu în timp real. Sunt prezentate contribuţii originale în acest suiect. Capitolul de concluzii prezintă contribuțiile originale cuprinse în teză şi impactul lor.

2. PROCESARE DE GEOMETRIE

Acest capitol descrie o parte a benzii de redare propuse în aceasta teză. Această parte se ocupă cu operațiile de determinare de vizibilitate, care sunt calculate cu ajutorul rasterizării. Algoritmii şi modulele prezentate în acest capitul sunt descrise în Figura 1. Modulele desenate cu verde includ algoritmi noi, introduși în aceasta teză.

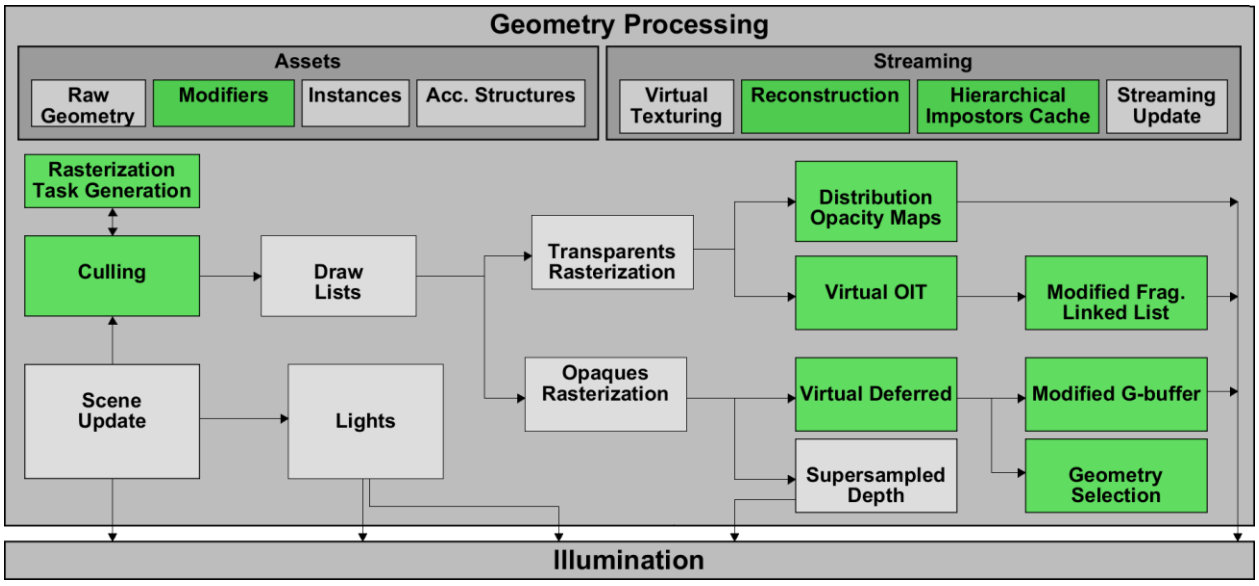

Figura 1 Modulele de Procesare de Geometrie**.**

2.1. Streaming

Pentru a calcula toate operațiile necesare procesului de redare, toate proprietățile obiectelor trebuie citite de pe hard disk, într-un proces de streaming. Din cauza aceasta una din contribuțiile minore ale tezei este de a rescrie toata definiția scenei prin modificatori, care iau un număr de obiecte de baza şi le modifică, instantiază, deplasează sau le dau traiectorii. Toată informația descrisa prin harți poate fi ușor controlată printr-un sistem de paginare, ca texturarea virtuală. Texturarea virtuală aplică principiul datelor virtuale atât texturilor cât şi nivelurilor de detaliu. Cum texturarea implica un proces de reconstrucție, în care o citire de textura implică citirea mai multor niveluri de detaliu şi interpolarea rezultatelor, algoritmul de texturare virtuala trebuie să țină cont de acest lucru. Un sistem de texturare virtuală este prezentat în Figura 1.

Informația geometrică poate folosi sistemul corespondent pentru geometrie, numit plase poligonale virtuale, unde plasele poligonale şi nivelurile lor de detaliu sunt tăiate în dimensiuni standard şi desenate indirect.

Streaming-ul este un proces scump, deoarece toate citirile de date de pe hard disk sunt operații foarte lungi. Pentru a minimiza dimensiunea datelor citite, se folosesc algoritmi de compresie, cei mai folosiți fiind algoritmii pe blocuri, care sunt direct implementați în hardware. Acești algoritmi reduc dimensiunea texturilor fără a introduce artefacte vizuale perceptibile.

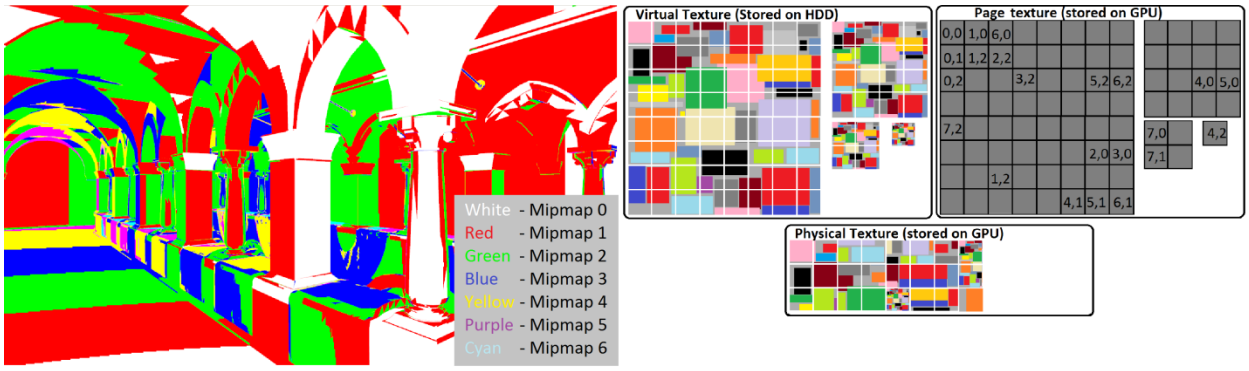

Figura 2 Texturare Virtuală**.**

Aplicațiile de texturare virtuala folosesc de obicei un sistem de graniţă pentru marginile paginilor, pentru a preveni artefactele generate de eșantionări greșite. Mărimea graniţei poate fi relaxată printr-o inovație recentă, ce implementează texturarea virtuala ca un sistem de pagini cu filtrare automata în hardware.

2.1.1. Redare Indirectă

Redarea indirectă este procesul în care datele nu sunt stocate într-o stare direct desenabilă, și dea aceea ele trebuie transformate într-un format compatibil cu algoritmii de redare. Redarea indirectă nu trebuie confundată cu desenarea indirectă, care este o funcționalitate a hardware-ului modern, cu care se pot genera comenzi de redare în mod indirect, fără vreun fel de interferentă din partea CPU-ului. În general, redarea indirectă este folosită în redarea în timp real pentru norii de puncte sau pentru redările bazate pe voxeli, care sunt mai întâi reconstruite în plase poligonale triangularizate şi apoi desenate normal. Acest proces este numit reconstrucție sau extragere de suprafață, și este operația inversă de la eșantionare, în cazul norilor de puncte și de la voxelizare, în cazul reprezentărilor volumetrice.

Extragerea de isosuprafețe este importantă în mod particular pentru redarea şi vizualizarea științifică pentru că este ușor de paralelizat şi este rezistentă la zgomot cu un număr relativ scăzut de eșantioane. Cele mai mari probleme ale algoritmilor de extragere de isosuprafețe sunt cerințele foarte mari de stocare cauzate de necesitatea de a lucra cu un număr mare de eșantioane cauzat de dimensiunea mare a seturilor reconstruite. Acest lucru se întâmplă des, mai ales la algoritmii ce lucrează pe seturi de date foarte mari, cum sunt cele folosite în vizualizarea medicală.

Teza prezintă o noua adaptare pe GPGPU a algoritmului „Marching Cubes”, care minimizează memoria consumată, făcând posibilă reconstrucția seturilor de date foarte mari în timp real. Algoritmul prezentat împarte datele volumetrice în sub volume, fiecare limitat la o capacitate maximă, numite chunk-uri. Acestea sunt reconstruite în mod serial pe GPU. Aceasta metoda are nevoie de mai putină memorie decât varianta standard GPGPU a algoritmului „Marching Cubes”. Algoritmul este vizual prezentat în Figura 3 si este publicat în [**Pet11**].

O alta proprietate utilă a algoritmului prezentat este că doar părțile relevante din setul de date volumetric sunt reconstruite la schimbarea datelor, astfel costul de reconstrucție este adaptiv în loc să fie constant.

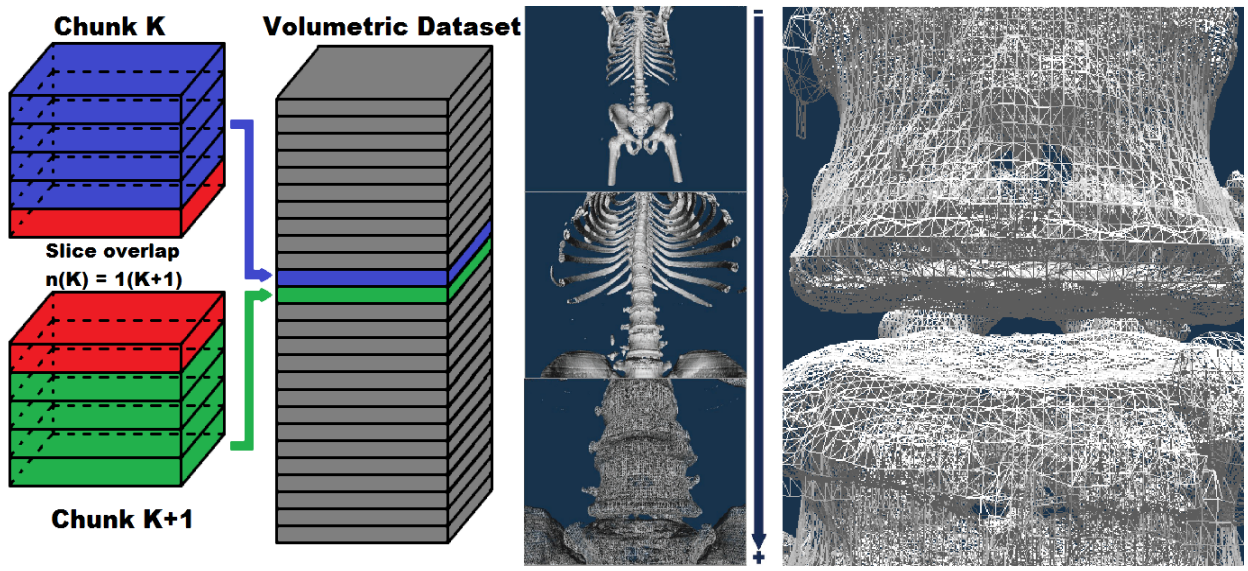

Figura 3 Algoritmul „Chunked Marching Cubes”**.**

Implementarea algoritmului prezentat a fost testată pe o placă video cu 1.5 GB de memorie GPU. Diferite mărimi au fost testate pentru chunk-uri, pentru a compara nivelurile de utilizare de memorie şi de a găsi dimensiunea optimă. Ocuparea de memorie obținută cu aceste diferite mărimi este prezentată în Tabelul 1.Deoarece ocuparea poate fi calculată în timp real, ea este folosită ca o metrică pentru a determina care este cea mai bună mărime pentru reconstrucție.

Mărime Chunk (nr subdiviziuni)	RAM (MB)	Ocuparea memorie
32	1495	99.6
16	1440	96
8	1432	95.4
4	>1500	>100

Tabel 1 Ocuparea de memorie pentru algoritmul „Chunked Marching Cubes”**.**

Acest algoritm este utilizat într-o aplicație medicală, „3D for Medicine”, ca parte a proiectului european SABIMAS, PNCDII-Joint Applied Research Projects, 2008-2011), http://se.cs.pub.ro/SABIMAS/ [**SAB15**].

Tehnica prezentata a fost adaptata pentru benzi de redare non fotoreliaste in [**Mor13**].

2.1.2. Impostori Ierarhici

Impostorii reprezintă o soluție parțială la două probleme din redarea în timp real: scalabilitatea şi antialiasing-ul. Redarea cu impostori scade dramatic complexitatea geometrică a scenei desenate şi de aceea scade semnificativ timpul de redare. Impostorii sunt reprezentări supra eșantionate ale obiectelor, stocațe în imagini, şi de aceea ele nu suferă de artefacte de reconstrucție, fiind reconstruite ca texturile. În plus, pentru că impostorii sunt stocați ca texturi, ei sunt foarte ușor de integrat într-un sistem de streaming ca texturarea virtuală.

Metoda nouă prezentată, impostori ierarhici, diferă de metodele de ultimă oră prin integrarea directă cu texturarea virtuală, prin abilitatea de a fi redați cu efecte de paralaxă şi prin faptul că sunt aplicabili explicit pe grupuri de obiecte oricât de mari. Fiecare impostor ierarhic

reprezintă un grup de obiecte, şi conține informație despre adâncime, normale şi culori, ce poate fi folosită apoi pentru o redare de înalta calitate a obiectelor distante. Când distanta de redare şi unghiul de redare cu care un impostor a fost creat diferă peste un prag fată de caracteristicile de redare actuale ale obiectului, impostorul este actualizat.

Algoritmul este construit în special pentru scene foarte mari, în care nu toată informația necesara redării poate fi stocată în memoria video. Întreaga scenă este încărcată pe hard disk într-o structură de accelerare. În funcție de poziția şi orientarea camerei, nodurile scenei care sunt mapate sus în structura de accelerare şi care sunt mai departe de cameră decât un anumit prag au impostorii calculați şi încărcați. Astfel, impostorii vor fi încărcați sau descărcați prin streaming pentru toate obiectele vizibile şi distante. Impostorii sunt reactualizați în funcție de unghiul de vedere şi de distanta de vedere. Din cauza aceasta, o a doua structură de accelerare poate fi folosită pentru obiecte dinamice.

Metoda propusă redă impostorii cu efecte de paralaxă. Fiecare impostor este trimis la banda grafica ca un punct, care este apoi extins la un dreptunghi aliniat în spațiul ecranului numit „billboard". Apoi billboard-ul este ajustat cu adâncimea impostorului şi pentru fiecare fragment rezultat din rasterizarea billboard-ului se efectuează o dubla căutare, pentru a determina exact punctul de intersecție între cameră şi suprafața. Prima dată se folosește o căutare liniară sau de secantă, ce determina aproximativ punctul de intersecție, care este urmată de o căutare binara, ce determină cu mare precizie punctul de intersecție. Informația de la punctul de intersecție e folosită în citirea din hărțile de culori şi normale, prin care este reprezentat impostorul.

Metoda de impostori ierarhici propusa diferă de metodele de ultimă oră prin integrarea directă în texturare virtuală şi prin construcția explicită peste multiple noduri de scenă. Deoarece metoda este ierarhică, utilizarea ei scade complexitatea procesului de redare de la $O(obiecte)$ la $O(\log(obiecte))$.

2.2. Generare de Task-uri în Rasterizare

În acest subcapitol este introdus un generator de task-uri care lucrează sub planificatorul de rasterizare. În comparație cu celelalte generatoare şi planificatoare de taskuri de ultimă oră, metoda introdusă nu funcționează în mod GPGPU ci funcționează în paralel cu procesul de rasterizare, de aceea face posibilă utilizarea de task-uri pe GPU pentru algoritmi de redare implementați în rasterizare.

Metoda prezentata se bazează pe ideea de a folosi funcţionalităţile de amplificare geometrica ale benzii de rasterizare hardware pentru a genera task-uri pe GPU. Acest lucru este obținut prin utilizarea modelului de shader 5, şi implementarea de shadere de teselare, geometrie şi fragmente. Metoda a fost publicată în [**Pet14**].

Primitivele originale sunt trimise la o banda de redare modificata. Etapele de teselare şi geometrie sunt ori adăugate, în cazul în care lipsea, ori modificate, astfel încât să se potrivească cu necesitățile generatorului de taskuri. În shaderul de control de teselare, efortul computațional pentru primitiva curenta este calculat şi daca acesta depăşeşte un prag definit de algoritmul de redare, se generează grupuri de task-uri. Grupurile de task-uri sunt apoi evaluate în shader-ul de evaluare de teselare, iar în shader-ul de geometrie mai multe alte sub-grupuri sunt create, folosind mecanismul de instanţiere de shader de geometrie.

Apoi, în shader-ul de geometrie, pentru fiecare sub-grup se efectuează o analiza a efortului computațional, şi se generează un număr foarte mare de task-uri pentru fiecare sub-grup. În shader-ul de fragmente, fiecare din aceste task-uri este executat ca un fir de execuție GPGPU. Acest proces este reprezentat în Figura 4. De asemenea, figura prezintă faptul că generatorul de task-uri prezentat nu este recursiv.

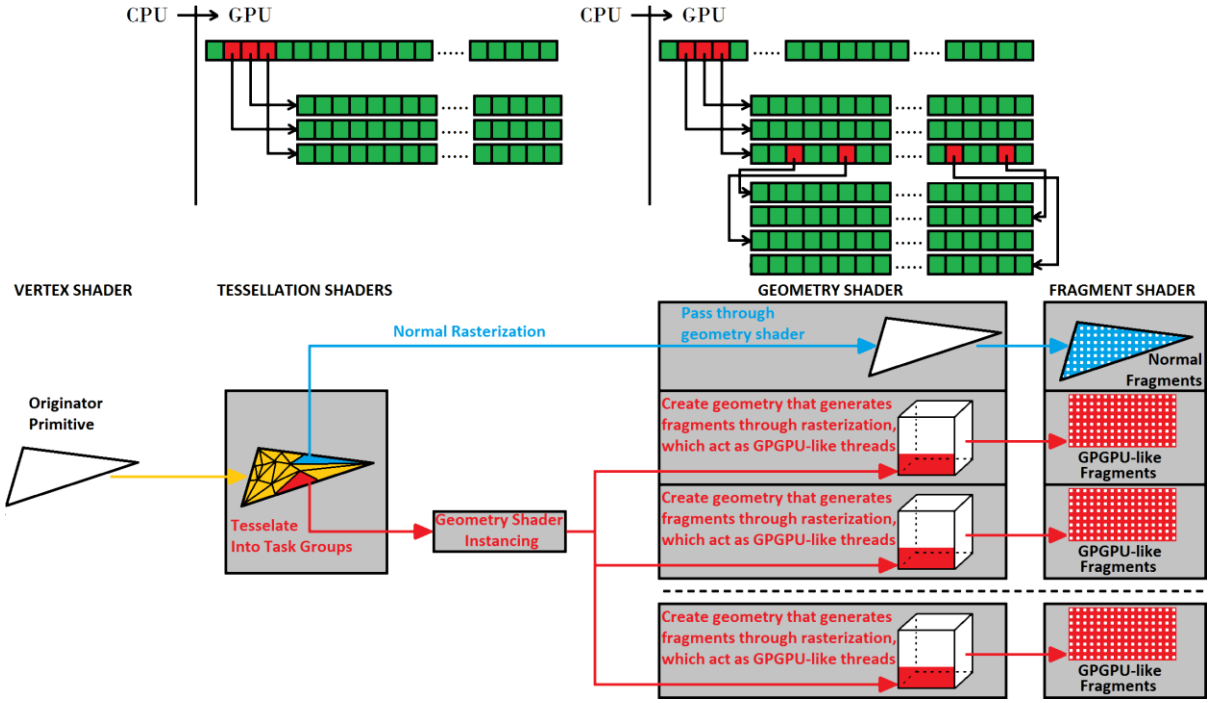

Figura 4 Generare de task-uri pentru rasterizare**.**

Primitiva originală este trimisă către rasterizare normală. Celelalte primitive sunt trimise către shader-ul de evaluare de teselare care decide pentru fiecare numărul de potențiale invocări a shader-ului de geometrie. Fiecare primitivă instanțiată în shader-ul de geometrie folosește o metrică dependentă de algoritmul de redare pentru a determina dimensiunea dreptunghiului aliniat la ecran, numit „billboard”, ce va fi desenat în spațiul de coordonate normalizate. Fiecare fragment obținut în urma rasterizarii billboard-ului este un fragment ce funcționează ca un fir de execuție GPGPU. Aceste fragmente nu scriu în zone de memorie ale ecranului ci în zona de memorie a rezultatelor de task-uri. Astfel, folosind generatorul de task-uri prezentat, marea majoritate a problemelor de redare ce se rezolva cu rasterizare hardware pot paraleliza eficient efortul computațional.

Deşi generatorul de task-uri prezentat nu este recursiv, marea majoritate a problemelor de redare ce necesita paralelizare în cadrul rasterizarii nu sunt recursive. Marile excepții, ca algoritmii de tip „Ray Tracing”, sunt eficient implementate prin alte metode, nu prin rasterizare.

Generatorul de task-uri prezentat pentru rasterizare este în întregime compatibil cu hardware-ul de rasterizare, lucrând prin planificatorul GPU pentru rasterizare. Generatorul poate genera task-uri în mod eficient, fără procesări a priori şi nu monopolizează GPU-ul aşa cum fac generatoarele şi planificatoare de task-uri de ultimă oră, ce controlează întreaga placă video.

Rezultatele de paralelizare obținute cu generatorul introdus sunt prezentate în Tabelul 2. DNP reprezintă non paralel dinamic.

Metoda Redare/ Task-uri	100k	200k	300k	400k	500k	600k	700k	800k	900k	1M	1.1M
DNP min	0.06	0.11	0.12	0.25	0.31	0.34	0.33	0.38	0.45	0.44	0.62
DNP med	0.07	0.15	0.25	0.30	0.38	0.35	0.35	0.48	0.60	0.63	0.76
DNP max	0.0	0.16	0.28	0.34	0.50	0.40	0.43	0.50	0.70	0.84	0.93
Metoda Prezentata min	0.00	0.00	0.01	0.01	0.01	0.01	0.02	0.02	0.02	0.02	0.03
Metoda Prezentata med	0.00	0.00	0.01	0.01	0.01	0.01	0.02	0.02	0.03	0.03	0.04
Metoda Prezentata max	0.00	0.00	0.01	0.01	0.02	0.02	0.02	0.03	0.04	0.04	0.06

Tabel 2 Rezultate de paralelizare pentru generatorul de task-uri**.**

Deşi generatorul de task-uri nu a fost construit pentru probleme recursive, el poate fi configurat ca să ruleze ca un generator recursiv. Acest lucru poate fi obținut prin înlănțuirea a multiple rulări cu generatorul prezentat. Totuşi, aceasta metoda implica un anumit nivel de control din partea CPU-ului.

2.3. Decupare Ierarhica pe GPU

Metodele de decupare de ultimă oră fie folosesc impostori geometrici fie nu sunt ierarhice sau sunt dependente de operații extrem de scumpe de sincronizare între CPU şi GPU. Din cauza aceasta, exista potențialul de îmbunătățire a algoritmilor de decupare. Decuparea ierarhica pe GPU este o metoda recurentă de decupare la nivel de volum vizual, care rulează în întregime pe GPU, fără nici un fel de interferenţă de la CPU şi fără calcule a-priori. Decuparea ierarhică pe GPU nu este limitată de impostori geometrici, dar poate beneficia de aceştia dacă sunt disponibili. Algoritmul este bazat pe generator-ul de task-uri prezentat anterior şi introduce de asemenea conceptul de decupare pe cadre multiple, în care un obiect poate fi decupat pentru mai multe cadre în funcție de poziția sa relativa fata de camera.

Metoda de decupare scrie toți indecşii obiectelor vizibile într-un buffer, care este apoi desenat cu funcționalitatea de desenare indirectă, care este implementată în driverul plăcii video. Această metoda poate fi de asemenea implementată cu paralelism dinamic.

Metoda de decupare prezentata utilizează principiile introduse de generatorul de task-uri, generând task-uri pentru fiecare nod din scenă vizitat de parcurgerea arborelui de scenă. Algoritmul începe cu nodurile superioare ale arborelui şi generează task-uri noi pentru copii fiecărui nod din arbore care sunt potențial vizibili. Nodurile vizibile au indexul adăugat la o listă de obiecte vizibile. Daca un nod este decupat, atunci şi toți copii săi sunt decupați şi nu se mai generează task-uri pentru aceştia.

Algoritmul poate fi implementat cu paralelism dinamic, dacă acesta este disponibil. Un avantaj al acestui algoritm este că nu foloseşte operații de sincronizare şi de aceea interacțiunea între CPU şi GPU este minimală, CPU-ul fiind folosit doar pentru inițializarea algoritmului.

Decuparea pe cadre multiple este contribuția finala a algoritmului de decupare introdus. În marea majoritate a scenelor, când un obiect este decupat, el nu este decupat pentru un singur cadru ci pentru mai multe. După cum este prezentat în Figura 5, un obiect poate fi decupat pentru mai multe cadre printr-o analiză a vitezei de rotație maxime a camerei şi a poziției obiectului. Daca direcția camerei este descrisă ca un unghi solid şi daca se consideră că în aplicațiile în timp

real există limite pentru viteza maximă de rotație a camerei, atunci întreaga scenă poate fi partiționată pe zone. Fiecare zonă de partiție e bazată pe ideea că ea nu poate fi vizibilă pentru cameră decât în minim un număr de cadre, bazat pe viteza maximă de rotație a camerei. Daca un obiect se află într-una din zone şi este decupat, atunci el este decupat pentru minim numărul de cadre necesar camerei pentru a se învârti către zona respectivă.

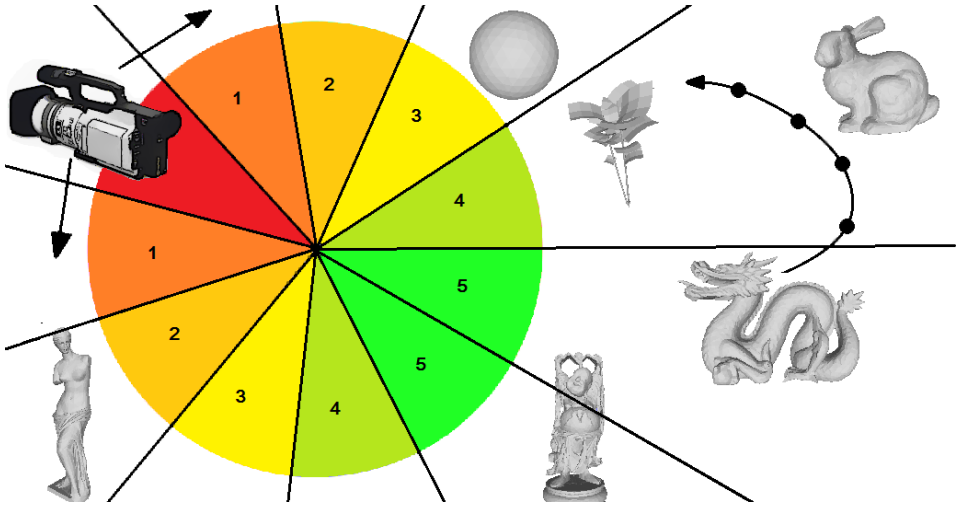

Figura 5 Decupare pe cadre multiple**.**

2.4. Rasterizare de Obiecte Opace

Singurii algoritmi de ultimă oră ce decuplează complet citirea de texturi şi operațiile de determinare de vizibilitate sunt toți foarte scumpi din punctul de vedere al procesării, pentru că ei folosesc ori operații foarte scumpe de sincronizare la nivel de GPU pentru a menține un cache, sau reconstruiesc geometria la nivel de fragment. În următorul subcapitol este prezentat un nou algoritm, numit „Deferred Virtual” care decuplează citirea texturilor de operațiile de determinare de vizibilitate, dar care are costuri comparabile cu alți algoritmi „Deferred” cu o singura etapă de procesare de geometrie.

2.4.1. Deferred Virtual

O problemă comună a tuturor algoritmilor de ultimă oră este că ori decuplează colorarea de operațiile de determinare de vizibilitate şi au mai multe etape de geometrie ori cuplează bandwidth-ul de colorare cu operațiile de determinare de vizibilitate şi folosesc o singura etapă de geometrie. Nu există nici o tehnică „Deferred” care să decupleze operațiile de determinare de vizibilitate de bandwidth-ul de texturi şi de colorare şi care să aibă doar o singură etapă de geometrie. Această teză introduce „Deferred Virtual”, un nou algoritm „Deferred”, publicat în [**Pet151**], care este capabil să facă această decuplare într-o singură etapa de geometrie.

Algoritmul introdus este o combinație intre metodele de date virtuale şi algoritmii „Deferred”, folosind mecanismul de texturare virtuală pentru a tine în buffer-ul de geometrie doar informațiile critice din punct de vedere geometric, adică coordonatele de texturare şi derivatele acestora. Astfel, algoritmul consumă bandwidth de textura doar atunci când acesta afectează procesul de redare, de exemplu pentru obiecte redate cu decupare alfa sau cu hărți de

deplasare. Făcând asta algoritmul „Deferred Virtual” garantează decuplarea completă între operațiile de determinare de vizibilitate, bandwidth-ul de colorare, bandwidth-ul de texturare, iluminare şi colorare. „Deferred Virtual” oferă aceste proprietăți într-o singura etapă de geometrie, combinand astfel cele mai bune metrici pentru algoritmii deferred, definite in [**Pet15**].

„Deferred Virtual” este bazat pe ideea de a utiliza texturarea virtuala pentru a maximiza operațiile de amânare. În loc să stocheze valorile citite din texturi în buffer-ul de geometrie, numit „G-Buffer”, metoda prezentată stochează coordonatele de texturare şi derivatele acestora. Astfel, în loc să se consume bandwidth de texturare pentru fiecare primitiva ocludată, „Deferred Virtual” salvează doar datele critic necesare procesului de redare. În funcție de configurația scenei redate, derivatele coordonatelor de texturare pot fi comprimate, așa cum este arătat în Figura 6.

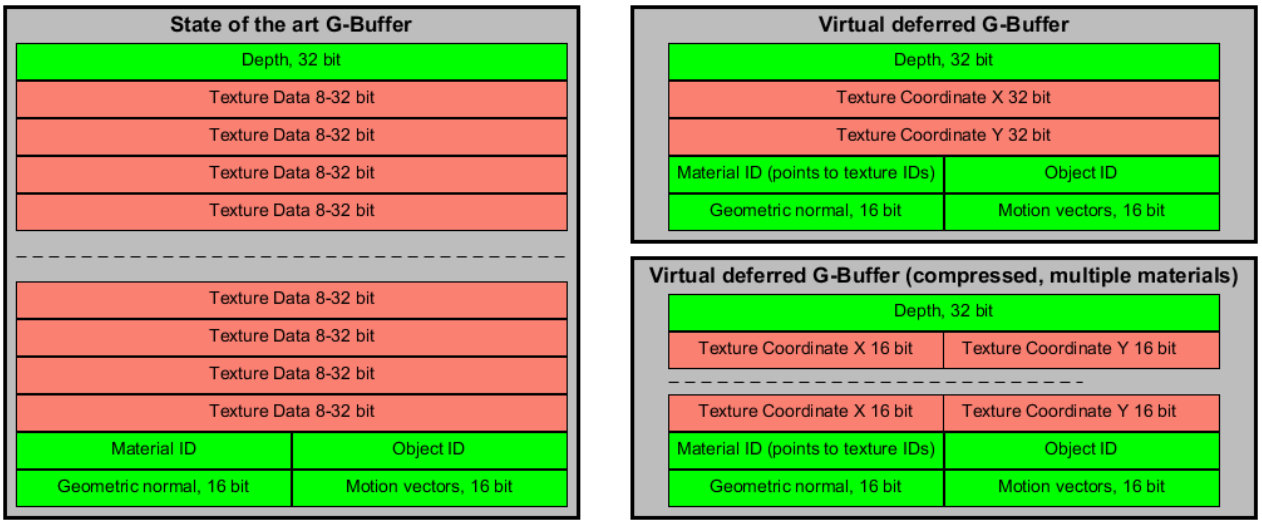

Figura 6 Buffer-ul de geometrie pentru algoritmul „Deferred Virtual”**.**

Etapa de iluminare de la „Deferred Virtual” este identică cu etapa de iluminare de la algoritmul de tip „Deferred” cu grid tridimensional, unde luminile sunt întâi stocate într-o structura de accelerare de tip grid 3D şi apoi sunt intersectate cu „G-Buffer”. În etapa de colorare, în loc ca datele de texturare să fie încărcate din „G-Buffer”, „Deferred Virtual” încărca doar coordonatele de texturare şi derivatele lor şi le folosește pentru a citi datele de texturare în textura virtuala. Deoarece „Deferred Virtual” e bazat pe texturare virtuală, este de asemenea foarte ușor de integrat într-un sistem de streaming bazat pe texturare virtuală.

Algoritmul „Deferred Virtual” poate fi combinat cu algoritmul de antialiasing sub geometric decuplat (DSRAA), un algoritm de antialiasing în stil de post procesare, prezentat în capitolul de Iluminare.

Etapa de iluminare de la „Deferred Virtual” e implementata ca un proces GPGPU, complet independentă de etapa de rasterizare. Colorarea se face folosind cache-ul fiecărui grup de fire de execuție GPGPU.

Din punct de vedere al complexității, „Deferred Virtual” scade complexitatea spațială de stocare şi a bandwidth-ul de texturare, așa cum este arătat în Tabelul 3 şi în Figura 7.

Algoritm\Criteriu	Redare „Deferred” cu o singura etapa de Geometrie	Redare „Deferred” cu mai multe etape de Geometrie	Redare „Deferred” Decuplata	Deferred Virtual
Procesare de Geometrie	1x	2x	1x	1x
Schimbări de Stare	1x	2x	1x	1x
Bandwidth Textura	T*TS*D	T*TS	T*TS	T*TS
Spaţiu stocare pentru Geometrie	Informaţie Geometrică + T*TS	Informaţie Geometrică	Informaţie Geometrică + memoizare	Informaţie Geometrică + 2

Tabel 3 Algoritmul „Deferred Virtual” fată de metodele de ultimă oră.

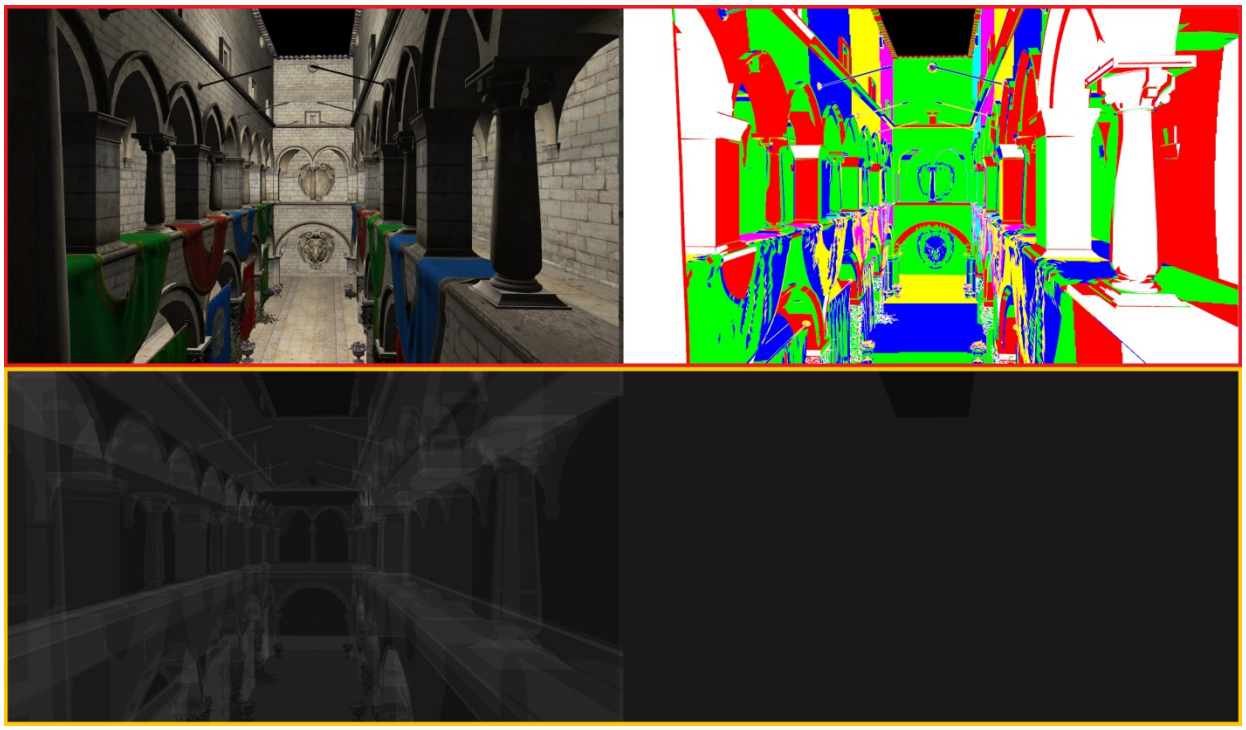

Figura 7 Rezultate algoritm „Deferred Virtual”**.**

Algoritmul „Deferred Virtual” poate rula în mod comprimat cu o optimizare în spațiu ecran, în care derivatele coordonatelor de textura nu sunt stocate, şi sunt calculate prin reconstrucție, folosind diferențe finite şi informațiile de culoare ale pixelilor vecini. Această optimizare scade şi mai mult consumul de bandwidth şi spațiu de stocare al algoritmului.

2.5. Rasterizare de Obiecte Transparente

Rasterizarea obiectelor transparente are similarități şi deosebiri cu rasterizarea de obiecte opace. Procesarea de triunghiuri, metodele de suprafața şi colorarea materialelor sunt ori identice ori foarte similare. De partea cealaltă, procesul de determinare de vizibilitate este diferit, determinat de lipsa de ordine a interacțiunilor suprafață-camera oferite de rasterizare.

Problema rasterizarii obiectelor transparente este foarte dificilă pentru că este o problema dependentă de ordine. Exista două metode de abordare: exactă sau aproximativă cu operator de vizibilitate aproximat. Diferența între aceste metode şi necesitatea pentru algoritmi exacți este prezentată în Figura 8.

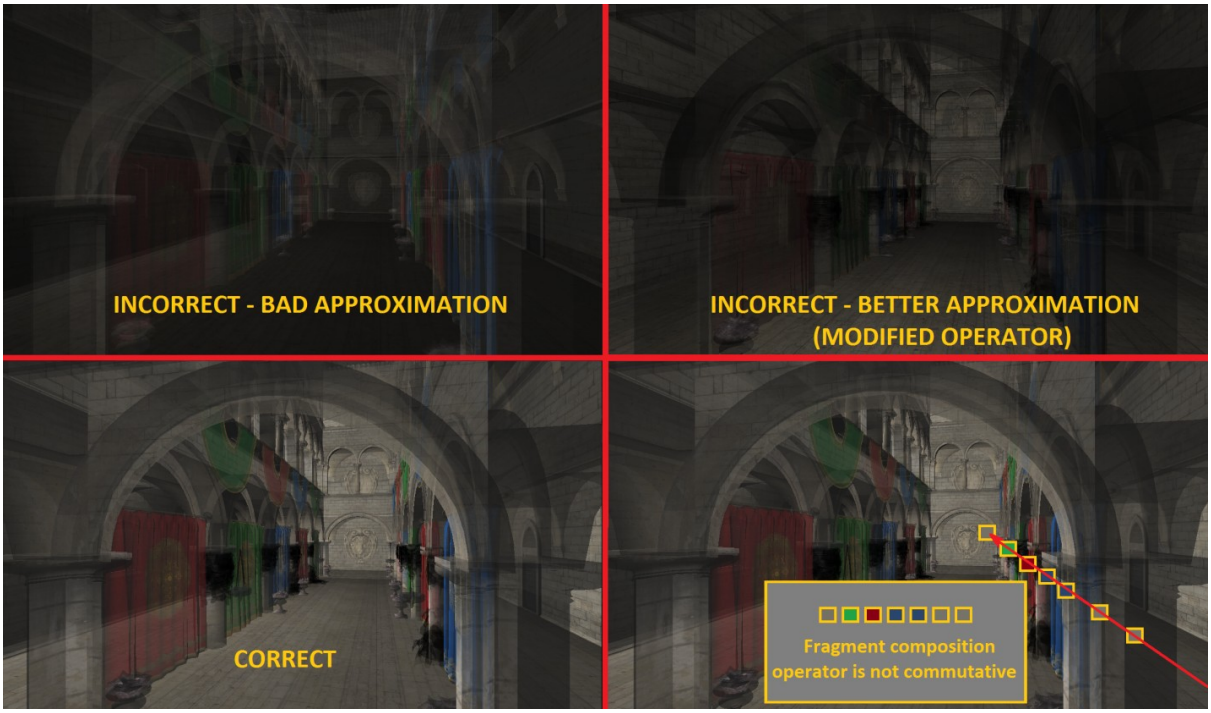

Figura 8 Transparenţă Independentă de Ordine**.**

2.5.1. Transparenţa Independenta de Ordine Virtuala

Unul din cei mai de succes algoritmi de rasterizare de obiecte transparente este metoda „A-Buffer” implementata pe GPU, singurele detrimente ale acesteia fiind costurile foarte mari de stocare şi bandwidth. Metoda „A-Buffer” poate fi transformata într-o metodă aproximativă, de calitate ridicată, prin folosirea de stocare stocastică, doar că în aceste cazuri metoda devine predispusă la artefacte temporale cauzate de diferitele aproximări. De aceea, în probleme în care precizia şi interactivitatea sunt critice, precum vizualizarea ştiinţifica, algoritmul original produce rezultatele cele mai bune.

Teza introduce o variaţie a algoritmului „A-Buffer”. Algoritmul de Transparenţă Independentă de Ordine Virtuala (VOIT, sau VA-Buffer) scade costurile excesive de stocare şi bandwidth-ul în comparaţie cu metoda „A-Buffer”. Metoda prezentata foloseşte aceleaşi principii de date virtuale care au fost aplicate şi pentru „Deferred Virtual” în sub-capitolul precedent.

Metoda „A-Buffer” colorează toate fragmentele generate de rasterizare înainte de a stoca culoarea în lista de noduri pe care o tine per pixel. În comparaţie, VOIT decuplează calculele de colorare de construcţia listei, de aceea VOIT se adaptează la situaţia de redare a fiecărui pixel, consumând doar strictul necesar. După ce lista per pixel este formata, algoritmul VOIT o sortează per-pixel, după care începe procesul de colorare, care se face de la camera în scena. Când VOIT determina că există suficientă ocluzie în canalul alfa, procesul de colorare ia sfârşit. Astfel, algoritmul VOIT colorează şi consuma bandwidth de texturare în mod adaptiv, încercând şi procesând date doar când acestea sunt garantate să aibă un impact vizual. În comparaţie „A-Buffer” încărcă şi colorează toate fragmentele.

Din cauza aceasta de performanta dar şi din cauza de decuplare, VOIT este un algoritm superior algoritmului „A-Buffer”. VOIT este prezentat vizual în Figura 9.

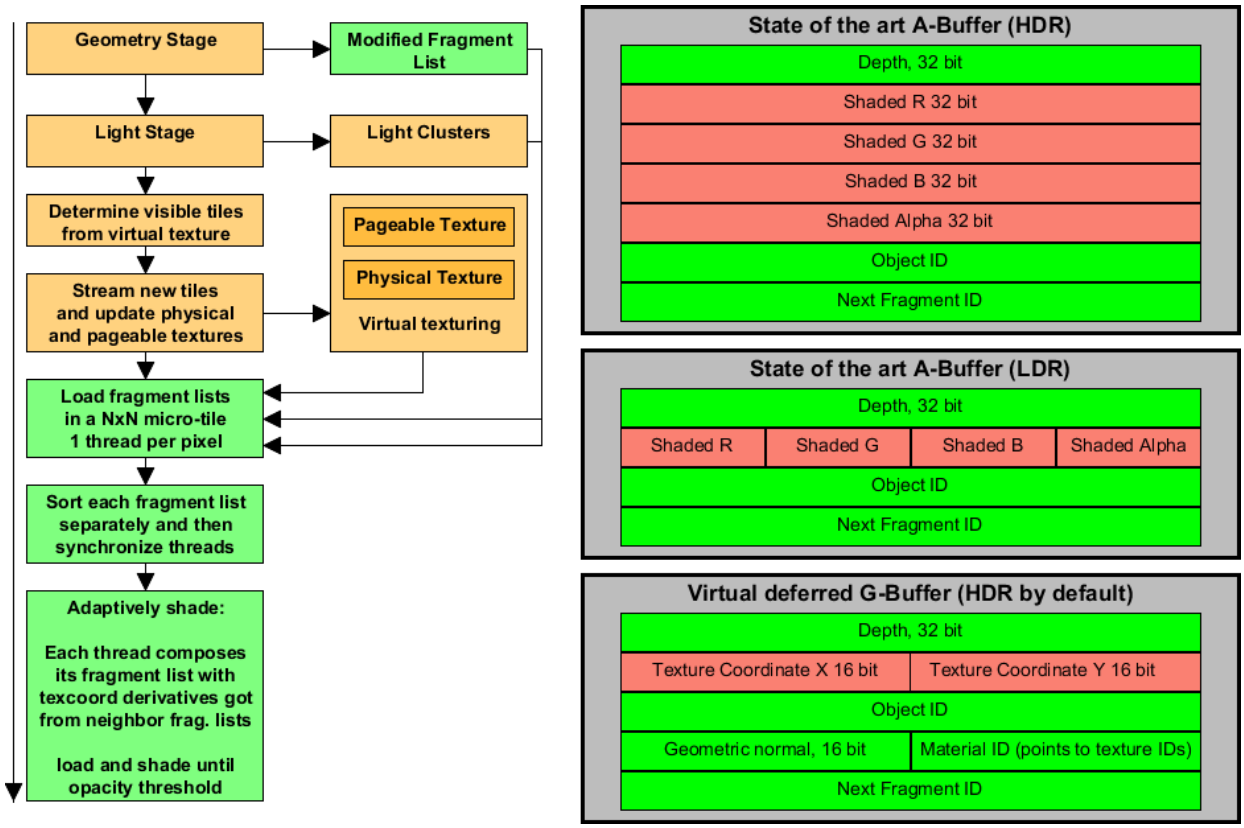

Figura 9 Transparență Independentă de Ordine Virtuala**.**

Algoritmul introdus funcționează în etape multiple, așa cum este descris în Figura 9. În prima etapa rasterizarea este folosita pentru a crea lista de fragmente per pixel. În comparație cu „A-Buffer”, VOIT nu stochează culorile fragmentelor ci coordonatele de texturare, în format comprimat. Din aceasta cauza VOIT are performante mai bune de stocare, în special pentru problemele de redare ce necesita HDR. În următoarea etapa se executa operațiile de integrare cu sistemul de texturare virtuala. Pentru a nu parcurge toate listele pentru a determina texturile referenţiate se folosesc doua buffer-uri în care starea texturilor e încodată binar.

Etapa de colorare a algoritmului VOIT este similara cu cea din „Deferred Virtual”, pentru că se bazează pe reconstrucție de derivate de coordonate de textura, obținute prin informația de la coordonatele de textura ale vecinilor.

Figura 10 prezintă o comparație între algoritmul VOIT şi algoritmul „A-Buffer”. Deși algoritmul „A-Buffer” are performante un pic mai bune de stocare, acestea sunt valabile doar pentru redarea în intervale joase (LDR) nu şi pentru intervale înalte dinamice (HDR). Mai mult, bandwidth-ul folosit de VOIT este mai mic, iar complexitatea adaptiva a VOIT este net superioara celei a algoritmului „A-Buffer”.

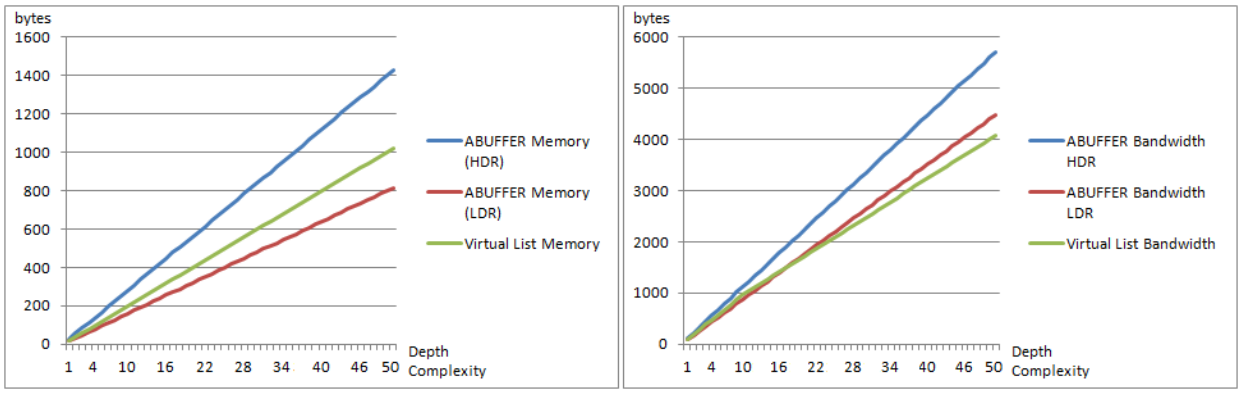

Figura 10 Transparență Independentă de Ordine Virtuala comparata cu metode de ultimă oră**.**

2.5.2. Hărţi de Ocupare cu Distribuții

Metodele aproximative de ultimă oră pentru algoritmi de rasterizare de obiecte transparente nu sunt suficient de rapide pentru fi utilizate în timp real pe obiecte cu geometrie de frecvenţă înaltă. Aceste metode ori aproximează [**Pet152**] ori redefinesc operatorul de compoziție şi sunt ori prea inexacte ori prea rigide pentru scenarii complexe de redare. În general aceste metode sunt folosite pentru a desena geometrie de frecvenţă joasa, cum sunt obiectele de tip fuzzy, pentru că algoritmii pot reda aceste obiecte cu rezultate vizuale acceptabile cu doar o mica parte din resursele folosite de algoritmii exacți.

În cazul harților de ocupare se face presupunerea adițională ca opacitatea este de un alfa constant. Aceasta presupunere folosește la redefinirea operatorului de compoziție într-unul care este o funcție peste adâncime. Comparat cu operatorul exact, acest operator este comutativ. Aceasta presupunere poate fi folosită în redarea în timp real, unde sunt multe obiecte transparente ce necesită redare, dar calitatea redării poate fi mai puțin exactă. Aceste proprietăți de redare sunt comune obiectelor fuzzy, ca fumul, norii sau lichidele. Pe baza acestor presupuneri teza introduce o varianta îmbunătățită a algoritmului hărți de ocupare. Hărțile de ocupare distribuite sunt hărți de ocupare care folosesc distribuții per pixel. Distribuțiile fac hărțile de ocupare adaptive prin schimbarea punctelor de eșantionare. În loc de a eșantiona uniform întreg intervalul de adâncime, hărțile de ocupare distribuite eșantionează ne-uniform pe baza unor distribuții ce depind de distribuția obiectelor pe axa de adâncime. Distribuțiile folosite sunt uniformă, şi gaussiană cu un singur sau mai mulți poli. Din cauza modificării eșantionării, hărțile de ocupare distribuite sunt adaptive, deci practic au o rezoluție crescută care se potrivește situației de la nivelul pixelului.

Hărțile de ocupare distribuite folosesc un buffer adițional, buffer-ul de distribuție, în care se menține ocuparea întregului interval de adâncime pe un număr foarte mic de biți, funcționând practic ca o hartă de ocupare de rezoluție foarte mica. Algoritmul stochează acest buffer adițional pentru cadrul curent cât şi pentru cadrul precedent. Biții buffer-ului de distribuție sunt folosiți pentru a partiționa întreaga rezoluție a hărții de ocupare în mod adaptiv, în funcție de distribuția fragmentelor pe axa de adâncime a pixelului. Fiecare partiție a hărții de ocupare folosește diferite strategii de eșantionare. Strategiile de eșantionare sunt uniformă, uniformă cu poli multipli, Gaussiană, Gaussiană cu poli multipli, sigmoida şi sigmoida cu poli multipli. Pentru un buffer de

distribuție de adâncime de opt biți toate modificările de eșantioane pot fi ținute într-o tabela de ofset-uri, în memoria GPU. Hărțile de ocupare distribuite sunt prezentate în Figura 11.

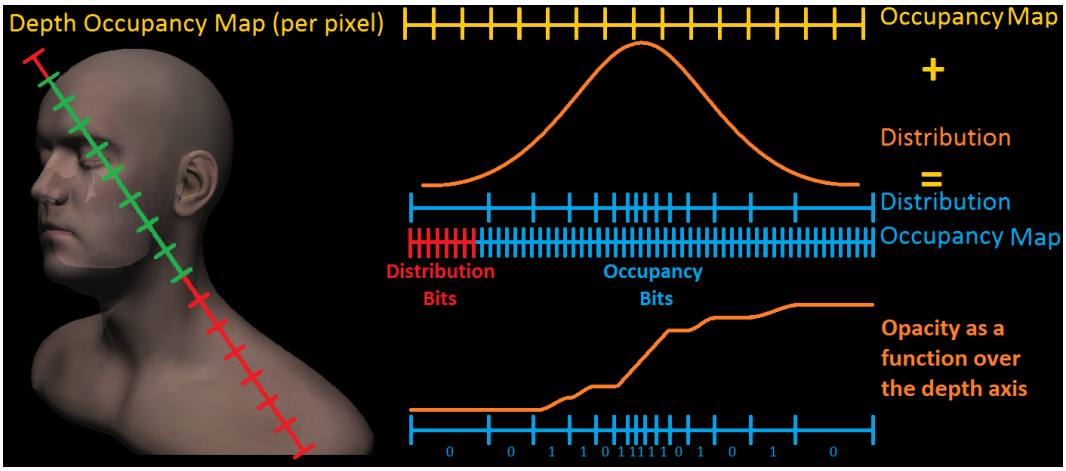

Figura 11 Harți de ocupare distribuite**.**

Înainte de redare, algoritmul aloca patru buffer-e: doua buffere de distribuție de adâncime (unul pentru cadrul precedent și unul pentru cadrul curent), un buffer de ocupare și o harta de ofset-uri. Bufferul de ocupare stochează ocuparea pe axa de adâncime a pixelului. Eșantioanele de ocupare sunt controlate de distribuția de adâncime de la cadrul precedent, prin ofset-urile citite din harta de ofset-uri.

Cele doua buffere de distribuție de adâncime stochează biții distribuției de adâncime pentru cadrul curent și precedent, pentru că algoritmul prezentat folosește una din cele doua distribuții de adâncime pentru a descrie cadrul curent dar folosește distribuția de adâncime de la cadrul precedent pentru a citi din harta de ofset-uri. Valorile citite din harta de ofset-uri sunt folosi pentru a modifica procesul de eșantionare.

Harta de ofset-uri stochează ofset-urile pentru punctele de eșantionare ale buffer-ului de ocupare, pentru fiecare dintre toate distribuțiile de adâncime reprezentabile în 8 biți. Harta de ofset-uri este împărțita în zone, una pentru fiecare distribuție de adâncime posibila. Fiecare zonă conține valorile necesare pentru modificarea procesului de eșantionare.

Algoritmul rulează în doua etape, etapa de geometrie și etapa de colorare și este un algoritm cuplat, care leagă procesul de colorare de determinarea de vizibilitate și de citirea texturilor.

În etapa de geometrie, geometria scenei este rasterizată normal, pentru a se obține adâncimile fragmentelor. Adâncimea fiecărui fragment e utilizată împreună cu distribuția de adâncime de la cadrul precedent pentru a seta biții din harta de ocupare cu metoda de eșantionare de la cadrul precedent. Distribuția de adâncime curenta este doar setată.

În etapa de colorare, geometria scenei este rasterizata normal, doar că de data aceasta este desenată cu un proces de amestecare aditiv. Culorile fiecărui fragment sunt calculate cu funcția de opacitate aproximata obținută în urma citirii hărții de ocupare de la etapa precedenta, folosind eșantionarea de la cadrul precedent.

2.6. Selecție de Geometrie cu Operații Atomice

În acest capitol o noua metoda pentru selecție este introdusa. Mai mulți algoritmi care rezolvă problema selecției există în contextul rasterizarii hardware, dar fiecare dintre ei nu are în totalitate funcționalitățile oferite de soluția prezentată şi toți au complexități mai slabe. Metoda introdusă este capabilă să selecteze corect nu doar primitive cât şi orice tip de obiect care ar putea apărea pe ecran la nivel de fragment, ca de exemplu obiecte rezultate din instanţiere hardware, obiecte decupate cu alfa, obiecte teselate hardware, obiecte animate hardware şi obiecte fuzzy. Tehnica propusă are consum de stocare şi bandwidth optime şi oferă oportunitatea de a selecta micro geometrie, nefiind limitată la prima interacțiune camera suprafață, oferind întreg setul de interacțiuni, daca este nevoie de așa ceva. Tehnica a fost publicata în [**Pet13**].

Tehnica propusă oferă alte oportunităţi unice cum ar fi selecție flexibilă de obiecte fuzzy şi tine cont de acumularea de opacitate de la obiectele transparente, neselectând obiecte care sunt ocludate de alfa acumulat. Unele din cele mai interesante cazuri de selecție suportate de algoritmul introdus sunt arătate în Figura 12.

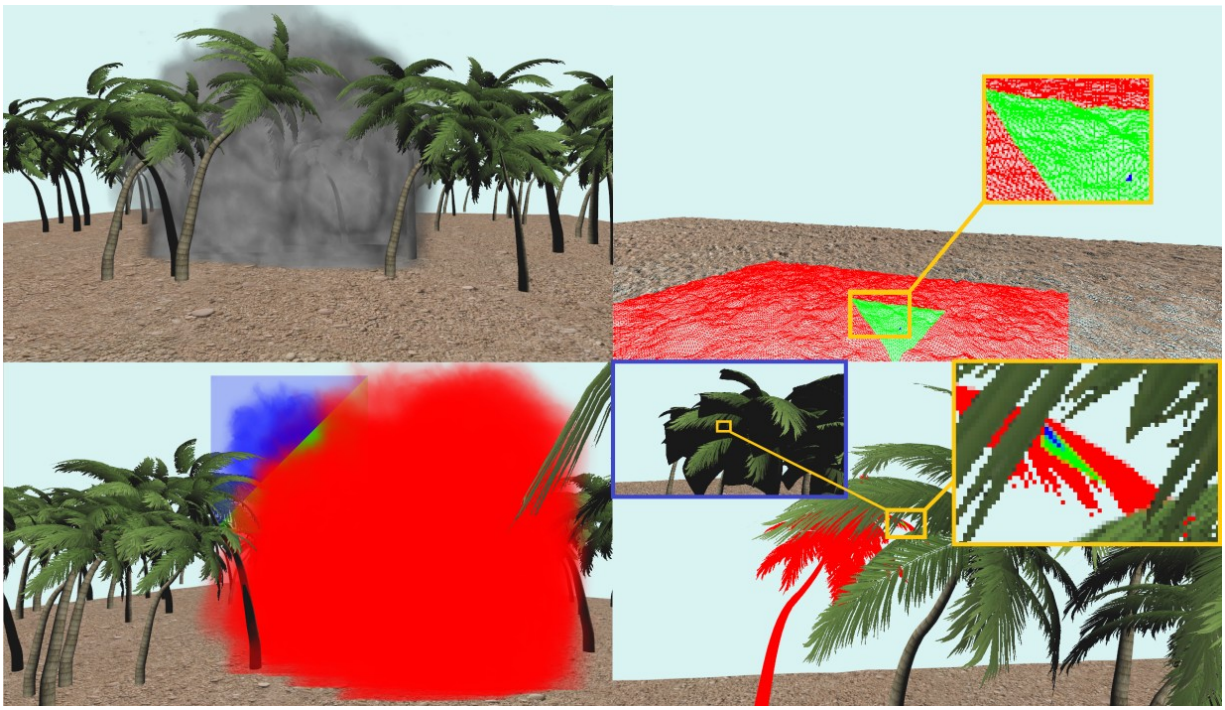

Figura 12 Selecție de Geometrie cu Operaţii Atomice – Cazuri dificile**.**

Contribuția principală a metodei de selecție prezentate este folosirea unui buffer sincronizat cu operații atomice, în care toate fragmentele care sunt rasterizate în zona de selecție sunt adăugate. Întreaga operație de salvare a punctelor de intersecție între camera şi suprafață este efectuata la nivel de fragment, de aceea metoda prezentată are detecție exactă la nivel de pixel şi poate selecta orice obiect desenabil, ca geometrie decupată cu alfa, obiecte fuzzy, sau micro geometrie generată cu teselare hardware, după cum este prezentat în Figura 12.

Comparat cu metodele de ultimă oră, tehnica prezentată nu aloca mai mult spațiu de stocare decât ceea ce este necesar şi procesează doar punctele de intersecție relevante între suprafeţe şi cameră, fără a desena geometria scenei de mai multe ori.

Algoritmul prezentat funcționează pe o zonă de selecție, care poate fi ori un singur pixel sau un număr mare de pixeli într-o zona dreptunghiulară. Algoritmul începe la nivelul vertex shader-ului şi trimite mai departe indexul obiectului, indexul vârfului şi indexul de instanță. Daca geometria desenată are etape de teselare hardware algoritmul trimite mai departe în banda grafica un index unic obținut printr-o funcție hash peste coordonatele baricentrice de teselare. Acest index unic este folosit pentru a identifica geometria temporară, generata de teselatorul hardware. Astfel, algoritmul prezentat poate selecta şi geometrie care nu este stocată permanent în memoria video. Metoda folosește un shader de geometrie pentru a determina indexul primitivei redate, care este şi el trimis mai departe în banda grafica. Fragment shader-ul este ultimă etapă programabilă din banda grafică şi colectează toate datele trimise de etapele precedente.

Daca fragmentul se află într-o zona de selecție, fragment shader-ul stochează punctul de intersecție, alături cu toate datele colectate de la etapele precedente. Punctele de intersecție sunt stocate într-un buffer, numit buffer de selecție, care este o listă înlănțuita per zona de selecție. Accesul la aceasta lista înlănțuită se face cu operații sincronizate atomic.

Daca cel mai apropiat fragment stocat în lista înlănțuită este opac, atunci algoritmul se termină cu selecția acestui fragment. Daca exista fragmente transparente în zona de selecție, şi daca acestea se află în fata celui mai apropiat fragment opac, metoda de selecție se termina cu o etapa de compute shader, care sortează toate fragmentele pe axa de adâncime a pixelului şi calculează acumularea de alfa, pentru a selecta doar obiectele ce au impact vizual.

Algoritmul de selecție este comparat cu metodele de ultimă oră în Tabelul 4. Se poate observa cum algoritmul prezentat combină toate proprietățile utile ale algoritmilor de ultimă oră şi în același timp adaugă alte proprietăţi utile cum ar fi selecția de micro primitive, selecția de obiecte fuzzy, ocluzia alfa şi selecția pe zone.

Algoritm de Selecţie	Proprietăți ale Algoritmilor de Selecție							
	Selecție Corecta	Capabil Ocluzie Alfa	Selecție în adâncime	Memorie 10 frag. la @1080p	Număr etape de geometrie	Instant-iere Hardwa re	Tesela-re Hardwa re	Selecție Fuzzy
Intersecție cu Ray Casting	Nu	Nu	Da	Nu se aplica	Nu se aplica	Nu	Nu	Nu
Intersecție cu Buffer de Adâncime	Da	Nu	Nu	66.35 mb	1	Da	Nu	Nu
Micro-Raster	Da	Nu	Da	320 b	2	Da	Nu	Nu
Selecție de Culoare	Da	Nu	Nu	49.76 mb	1	Da	Nu	Nu
Selecție atomica	Da	Nu	Nu	320 b	1	Da	Nu	Nu
Transform feedback	Nu	Nu	Da	320 b	1	Da	Da	Nu
AGS (acest algoritm)	Da	Da	Da	320 b	1	Da	Da	Da

Table 4 Comparație între algoritmi de selecție**.**

3. ILUMINARE

Acest capitol descrie a doua parte e benzii de redare propuse, care conține algoritmi pentru iluminare globala şi colorare. Modulele şi algoritmii prezentați în acest capitol sunt ilustraţi succint în Figura 13. Modulele colorate cu verde sunt contribuții personale.

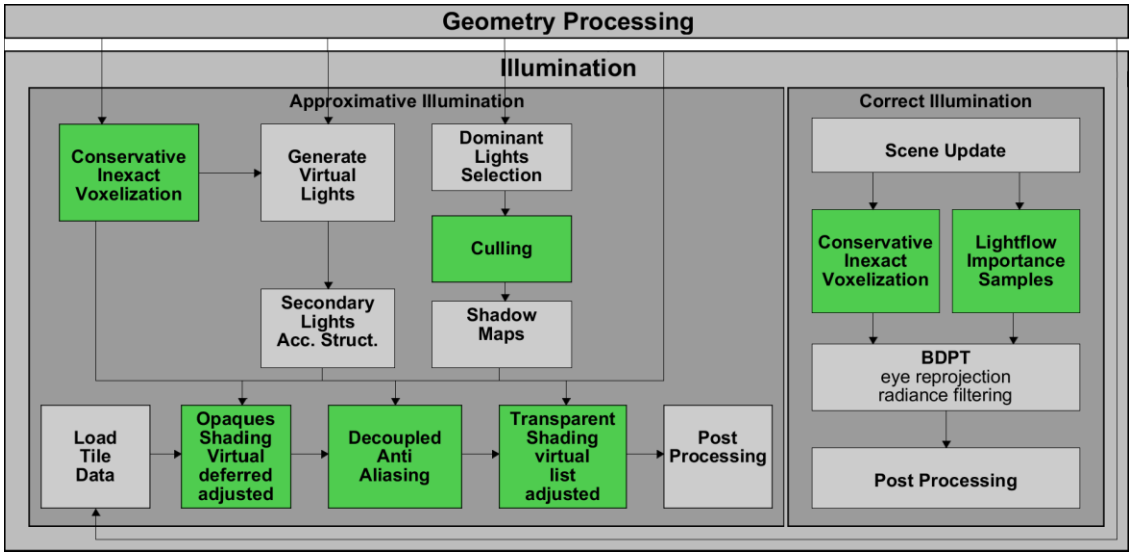

Figura 13 Modulele de Iluminare**.**

Acest capitol este împărțit în doua mari sub-capitole, care oferă soluții pentru problema iluminării globale din perspective diferite: primul sub-capitol prezintă o soluție aproximativă care rulează în timp real şi a al doilea prezintă o soluție corecta ce rulează parțial în mod interactiv. Cu excepția algoritmului de mapare de umbre, care este folosit în sub-capitolul de iluminare aproximativă, întreaga bandă de iluminare este independentă de rasterizare, fiind un proces complet GPGPU.

Subcapitolul de iluminare aproximativă oferă o soluție iluminării globale din perspectiva timpului real, folosind operatori de vizibilitate aproximați pentru a propaga lumina prin scenă. Modulele subcapitolului folosesc rezultatele obținute în capitolul de procesare de geometrie, cum ar fi buffer-ul de geometrie virtuală şi listele per pixel de la transparenta independentă de ordine. Procesul de iluminare e decuplat în mai mulți algoritmi şi căi de redare, fiecare dintre ele lucrând un anumite tipuri de propagări de lumină. Deşi rezultatele vizuale sunt plăcute şi apropiate de fotorealism, sub-capitolul de iluminare aproximativă nu poate simula toate tipurile de transporturi de lumina, mai ales transporturile pe căi cu multe interacțiuni cu un nivel de ridicat de specularitate.

Modulele din iluminarea aproximativă estimează contribuția potențială de la luminile scenei şi selectează un număr redus dintre acestea, care sunt considerate lumini dominante, pentru care se calculează hărți de umbră. Pentru restul luminilor, considerate lumini secundare, se utilizează o structura inovativa, numita voxelizare inexacta conservativă.

Modulele din iluminarea corectă calculează exact transportul de lumina, utilizând operatori de vizibilitate exacți şi simulând în detaliu interacțiunile cu suprafețele.

3.1. Iluminare Aproximativă

Iluminarea aproximativă este o problemă de redare bazată pe percepție, care încearcă să creeze imagini aproape fotorealiste fără nici un fel de preprocesare. Iluminarea aproximativă folosește diferiți operatori de vizibilitate aproximativi pentru transportul indirect de lumină difuză şi pentru transportul indirect de lumină speculară. În același timp folosește operatori exacți pentru transportul direct de lumină, maximizând eficienta şi corectitudinea luminii care este cel mai ușor de perceput. Cea mai mare problema a benzii de redare cu iluminare aproximativă este transportul de lumină pe căile speculare. Aceasta este o problema comună pentru redarea în timp real, pentru că operatorii de vizibilitate exacți nu pot fi implementați decât cu raze, path-uri sau fotoni, care reprezintă soluții mult prea scumpe din punct de vedere computațional pentru a fi implementate în timp real.

Soluția de redare propusa combina rasterizarea implementata pentru hărțile de umbre folosite pentru determinarea vizibilității la luminile dominante cu soluțiile bazate pe multe lumini care sunt implementate pentru luminile secundare. Soluțiile pe multe lumini folosesc operatori de vizibilitate aproximativi pe o structura de accelerare creată per cadru, în care se efectuează algoritmul „Ray Tracing". Structura de accelerare folosita e implementată ca o voxelizare inexactă conservativă (CIV), care este creata din înfășurători convexe ale obiectelor în loc de înfășurături convexe ale primitivelor obiectelor, şi este din acest motiv extrem de rapidă de construit. Soluția de multe lumini se bazează pe un algoritm de drumuri aleatoare înăuntru structurii CIV, în care la fiecare contact între drumul aleator şi datele din CIV se creează noi lumini, numite lumini virtuale. Transportul de lumină specular se efectuează cu algoritmul „Cone Tracing" în spațiu ecran, care are cazurile de eşec augmentate cu „Ray Tracing" în structura CIV.

Banda de redare aproximativă propusă calculează întâi transportul de lumină pentru luminile dominante cu hărți de umbră de ultimă oră. Apoi, construiește structura CIV peste obiectele scenei. Luminile secundare sunt folosite pentru a genera căile aleatoare prin CIV, care sunt folosite pentru a genera mii de lumini virtuale.

3.1.1. Transport de Lumină pentru Lumini Dominante

Luminile dominante sunt rezolvate cu hărți de umbră pentru că operatorii de vizibilitate implementați prin rasterizare sunt de calitatea cea mai înaltă care poate fi folosită pentru redarea în timp real. Deşi razele pot fi de asemenea folosite pentru operatorii de determinare de vizibilitate, aceasta sunt semnificativ mai încete. Pentru a implementa „Ray Tracing" de calitate în timp real este nevoie de o structura de accelerare coerentă, similară cu cea folosită în „Cone Tracing" peste reprezentări volumetrice rare, dar costurile de calcul pentru o asemenea metodă sunt foarte ridicate, iar metoda este complet dependenta de preprocesare. Mai mult, metoda este extrem de scumpă pentru scene dinamice, deoarece părți sau chiar întreaga structură de accelerare trebuie reconstruită la fiecare cadru. Din aceste motive hărțile de umbră sunt metoda cu cel mai bun raport dintre performantă şi cost.

Banda de redare prezentată folosește hărți de umbră în cascadă peste care se executa „Ray Tracing". Hărțile de umbră sunt filtrate într-o etapă la nivel de ecran, cu o dimensiune de kernel ajustabilă, în funcție de distanta la ocludator. Soluția prezentată desenează scena din punctul de vedere al luminii şi în loc să stocheze adâncime stochează indicii primitivelor ce sunt

rasterizate în maparea de umbre standard. În etapa următoare, de redare, primitivele din vecinătatea pixelului colorat sunt încărcate, se calculează intersecțiile între razele de la camera la primitive şi se obțin diferite valori de umbră, care sunt apoi filtrate în vecinătatea pixelului. Acest proces permite generarea de umbre de calitate foarte ridicată, cu un număr de eșantioane foarte scăzut. Deși aceasta metoda poate suferi de aliasing, acesta poate fi scăzut prin folosirea nivelurile de detaliu pentru geometrie şi a impostorilor ierarhici.

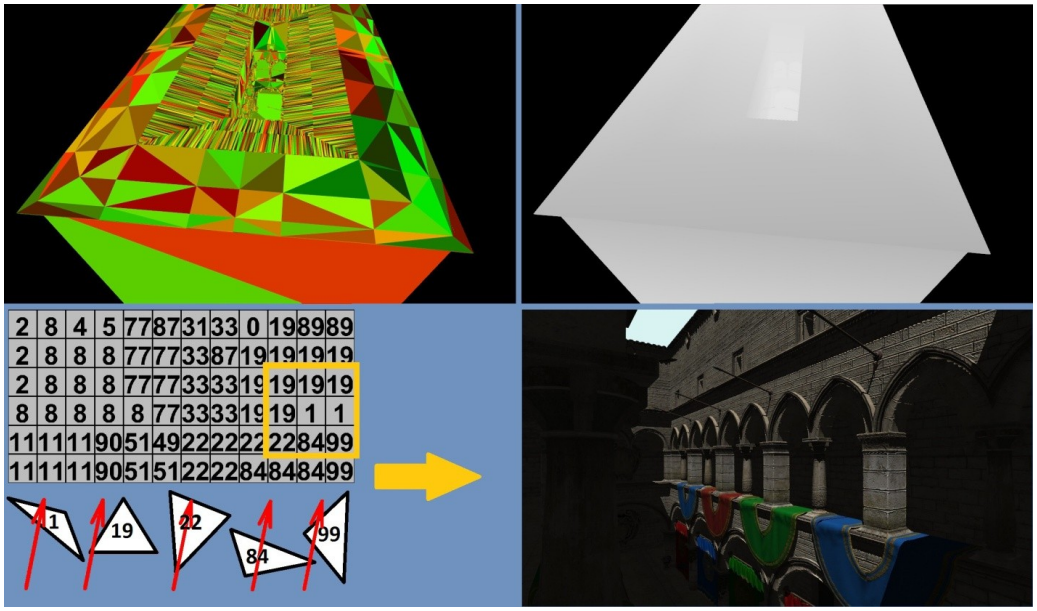

Figura 14 Harţi de umbră cu „Ray Tracing”**.**

3.1.2. Voxelizare Inexactă Conservativă

Din cauza obiectelor dinamice din scene, voxelizarea scenelor trebuie să fie calculată la fiecare cadru, şi din această cauză complexitatea de construcție trebuie să fie foarte scăzută. Problema cu metodele de voxelizare existente este ca nu există nici una cu o complexitate suficient de scăzută, dar care să ofere o reprezentare pentru întreaga scenă. Din acest motiv, toate metodele de voxelizare cu informații despre toata scena sunt metode interactive, fiind mult prea scumpe pentru a fi folosite în redarea în timp real pe scene masive.

În această teză este prezentat un nou algoritm de voxelizare, care diferă de metodele de ultimă oră în mai multe moduri : este o reprezentare inexactă dar conservativă a scenei şi are o complexitate de construcție foarte scăzută. Aceasta metodă, numita voxelizare inexactă conservativă (CIV), este construită pentru a fi folosita în metodele cu multe lumini, așa cum este prezentat în sub-capitolul următor, unde aceasta reprezentare inexactă a geometriei scenei este folosita pentru transportul global de lumina difuza. Acolo CIV este folosit pentru a relaxa costul operatorului de determinare de vizibilitate, şi astfel accelerează întreg transportul de lumina obținut prin generarea a mii de lumini virtuale.

Comparat cu metodele de ultimă oră, CIV pornește cu o altă perspectivă asupra procesului de voxelizare, lucrând de sus în jos. Aceasta alegere, de a voxeliza direct obiecte şi nu geometria acestora, conferă viteza algoritmului, pentru că garantează o complexitate de

construcție foarte scăzută, $O(nr\ obiecte)$, în comparaţie cu metodele de ultimă oră care lucrează în complexitate de $O(nr\ primitive)$, sau chiar $O(nr\ vertecsi)$. Metoda prezentată utilizează de asemenea toate datele deja existente ce sunt găsite în mod uzual în redarea în timp real cu metode „Deferred", proiectând toate datele din bufferul de geometrie în structura de accelerare şi garantând astfel un operator exact de transport de lumină în interiorul volumului vizual.

Astfel, ideea principala a CIV este de a crea o reprezentare volumetrică ierarhică exactă în interiorul volumului vizual şi inexactă în afara acestuia.

Intrările în structura de date se fac într-o textura tridimensională. Algoritmul parcurge arborele scenei şi determina ce obiecte sau noduri de scena trebuie voxelizate. Obiectele care au fost decupate în afara volumului vizual sunt extinse pe axele cubului încadrator, care e salvat ca o singura intrare în textura ce conține reprezentarea volumetrica a scenei. Astfel pentru întreg cubul încadrator este nevoie de o singura operație de scriere, efectuată la cel mai jos nivel aplicabil din structura ierarhica. Pentru obiectele elongate pe una din axe se aplica un proces de diviziune pe axa în cauză, rezultând un set de cuburi, pentru care se aplică metoda normală. Această etapă este urmată de reproiecția geometriei stocate în buffer-ul de geometrie creat de algoritmul „Deferred" folosit la etapa de procesare de geometrie.

Un proces tridimensional al algoritmului „Push-Pull" este folosit pentru a transfera informația stocată în structura de date ierarhică. Algoritmul „Push-Pull" folosește informație de la toate nivelurile ierarhice pentru a oferi cunoștințe aproximative asupra spațiilor în care nu există informație geometrică, pe baza informației stocate la alte niveluri de detaliu ale texturii CIV. Algoritmul CIV este prezentat vizual în Figura 15.

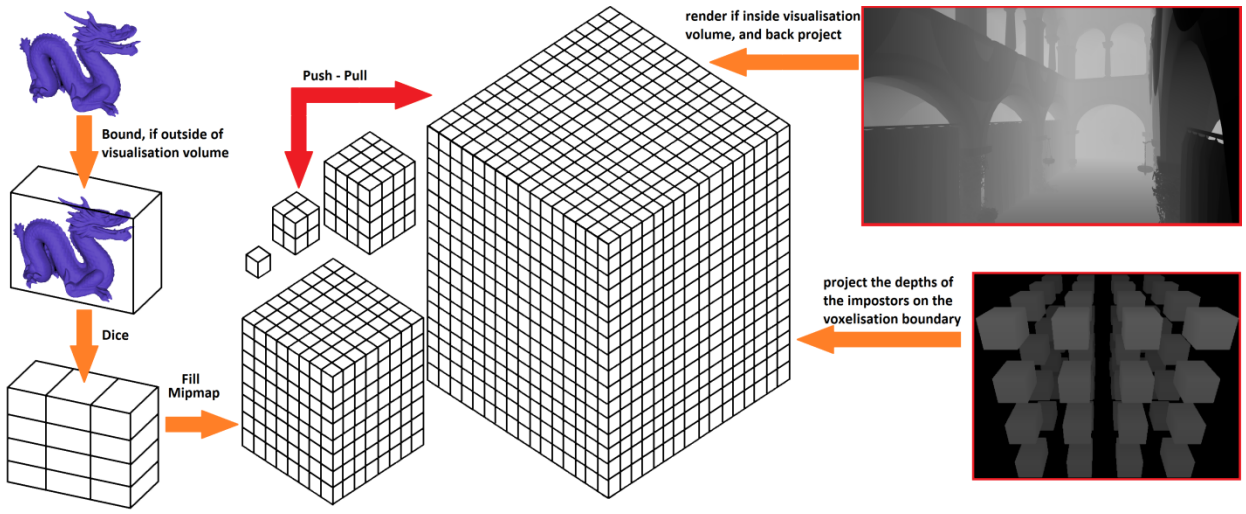

Figura 15 Voxelizare Inexactă Conservativă**.**

Structura CIV este apoi folosită ca un operator de vizibilitate pentru un proces de „Ray Tracing".

Structura de accelerare prezentată consumă memorie în funcție de strategia de redare utilizată, putând să stocheze informații adiționale despre normale sau natura materialului geometriei aproximate. O comparație cu metodele de ultimă oră de voxelizare e prezentată în Tabelul 5.

Algoritm de Redare \ Criteriu de comparație	Toata scena	Conservativ	Complexitate	Suport pt. Ray Tracing	Suport Împrăștiere
Reconstrucție de Nori de Puncte	da	nu	O(vârfuri)	nu	nu
Înfășurătoare Convexa cu Nori de Puncte	da	da	O(vârfuri)	nu	nu
Harți de Umbră Imperfect Voxelizate	da	da	O(lumini* vârfuri)	nu	parțial
Voxelizare în Spațiul Ecranului	nu	da	O(primitive)	parțial	parțial
Volume Imperfecte	da	nu	O(primitive)	da	da
Octree rari	da	da	O(primitive)	da	da
Voxelizare Inexacta Conservativa (acest algoritm)	da	da	O(obiecte)	da	da

Tabel 5 CIV fată de celelalte structuri de voxelizare**.**

O proprietate cheie a structurii de date introduse este acea că oferă un operator de vizibilitate variabil pentru „Ray Tracing", care se adaptează în funcție de camera. Operatorul de vizibilitate este exact în interiorul volumului de vizualizare şi inexact în afara acestuia. Acest lucru este prezentat în Figura 16. Aceasta proprietate este foarte utilă pentru transportul de lumină aproximativ, pentru că se adaptează la zonele perceptual importante ale scenei şi poate fi folosita atât pentru lumina de frecvenţă joasă cât şi de frecvenţă înalta.

$$L(x_1 \to x_0) = L_e(x_1 \to x_0) + \int_A f_r(x_2 \to x_0) G(x_1, x_2) V(x_1, x_2) L(x_2 \to x_1) dA(x_2)$$

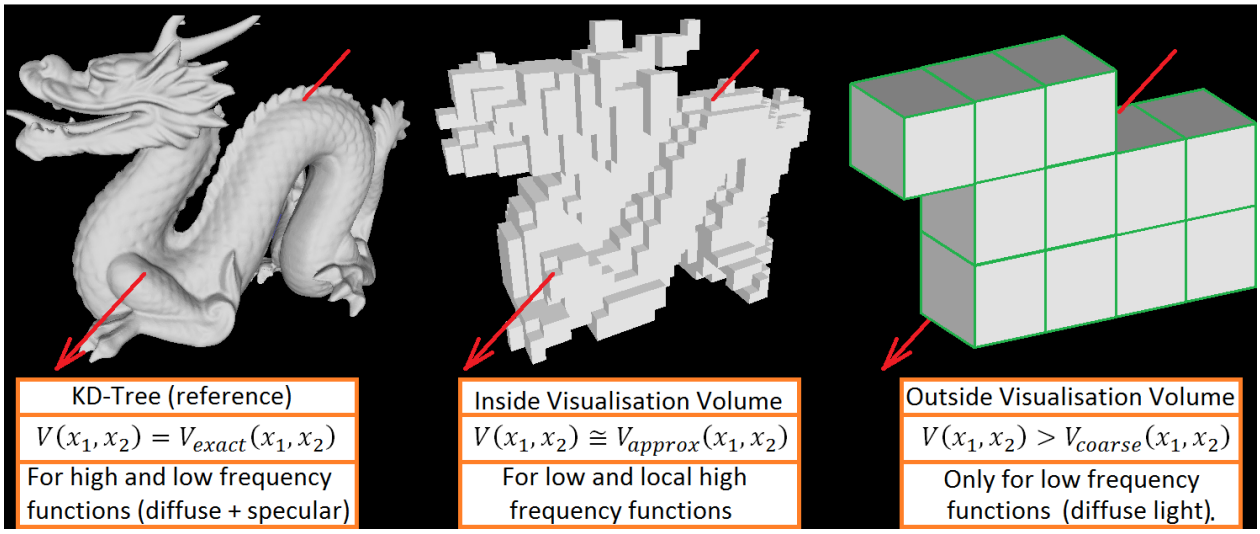

Figura 16 Operator de Determinare de Vizibilitate variabil cu Voxelizare Inexacta Conservativa**.**

Metoda de voxelizare prezentata nu oferă vizibilitate exactă dar oferă o aproximare conservativă a acesteia. Aceasta structură de accelerare este folosita în sub-capitolul următor într-o variantă modificată a algoritmului „Instant Radiosity". Structura de accelerare prezentată este de asemenea utilizată pentru a augmenta rezultatele algoritmului „Cone Tracing" în spațiu ecran.

3.1.3. Transport de Lumină pentru Lumini Secundare

Acest capitol foloseşte o soluţie de transport de lumină decuplată, utilizând diferiţi algoritmi pentru transportul de lumina de frecvenţă înaltă şi frecvenţă joasă. Pentru transport de lumină de frecvenţă joasă (difuză) se utilizează o metoda nouă, bazată pe algoritmul „Instant Radiosity" modificat. Pentru transportul de lumină de frecvenţă înalta se utilizează „Cone Tracing" în spaţiu ecran, cu cazurile de eşec augmentate cu structura de date prezentată la 3.1.2. Rezultatele celor doi algoritmi sunt apoi combinate, pentru a obţine transportul de lumină total.

3.1.3.1. Transport de Lumină de Frecvenţă Joasă

Metodele de iluminare bazate pe multe lumini au fost introduse cu algoritmul „Instant Radiosity", ce foloseşte drumuri aleatoare înăuntrul scenei pentru a transfera cantităţi mari de lumină în putini paşi. De fiecare dată când drumurile aleatoare prin scenă intra în contact cu suprafeţe ale obiectelor, se generează noi lumini, numite lumini virtuale. Din cauza naturii recursive a drumurilor prin scenă, algoritmul „Instant Radiosity" are nevoie de operaţii de vizibilitate recursive şi nu poate fi implementat eficient pentru un număr mare de lumini cu rasterizare.

Varianta algoritmului utilizata în acest sub-capitol pentru transportul de lumina de frecvenţă joasă foloseşte raze pentru a determina vizibilitatea între doua suprafeţe. Această decizie face ca „Instant Radiosity" să fie dificil de implementat fără o structură de accelerație care să crească rata de eşantionare a scenelor dificile, în care multe drumuri aleatoare sunt necesare pentru a transporta lumina.

În metoda prezentată, drumurile aleatoare sunt implementate folosind „Ray Tracing" peste o reprezentare aproximativă a scenei, oferită de voxelizarea conservativă inexactă prezentată în sub-capitolul precedent. Din cauza acestei decizii, ecuaţia de redare este modificată, utilizând acum un nou operator de vizibilitate în locul operatorului exact original. Operatorul de vizibilitate introdus este atât coerent din punct de vedere al accesului la memorie cât şi conservativ, şi este în acelaşi timp adaptiv din punct de vedere al percepţiei scenei. Operatorul este prezentat în Figura 16. Toate aceste proprietăţi fac acest operator ideal pentru transportul de lumina necesar în generarea rapidă de lumini virtuale pentru iluminare globala. Operatorul este folosit în drumurile aleatoare ce generează luminile virtuale, aşa cum este arătat în Figura 17, cât şi pentru „Ray Tracing", necesar în determinarea factorului de umbră.

Procesul de drum aleator generează lumini virtuale. Un număr mic de eşantioane este generat pentru fiecare lumină din scenă. Fiecare eşantion urmează un proces de „Ray Tracing", similar cu etapa de „Light Tracing" de la algoritmul de „Ray Tracing Bidirecţional" sau „Path Tracing Bidirecţional". Strategii de mutaţie a path-ului rezultat pot fi folosite. Fiecare vârf al path-ului este tratat ca un o lumină virtuală punctiformă (VPL) potenţială. În funcţie de structura de stocare CIV, variaţia prezentată de „Instant Radiosity" poate sau nu să calculeze transferul de culoare indirect. Nu toate VPL-urile generate sunt salvate, criteriul de rejecţie pentru VPL-uri este bazat pe eşantionare de importanţă. Eşantionarea de importanţă tine cont de poziţia VPL-ului relativ la volumul vizual şi rejectează toate VPL-urile care nici nu sunt poziţionate în volumul vizual şi nici nu sunt vecine în path cu un alt VPL care să fie poziţionat în volumul vizual.

Procesul de generare de VPL-uri este prezentat în Figura 17.

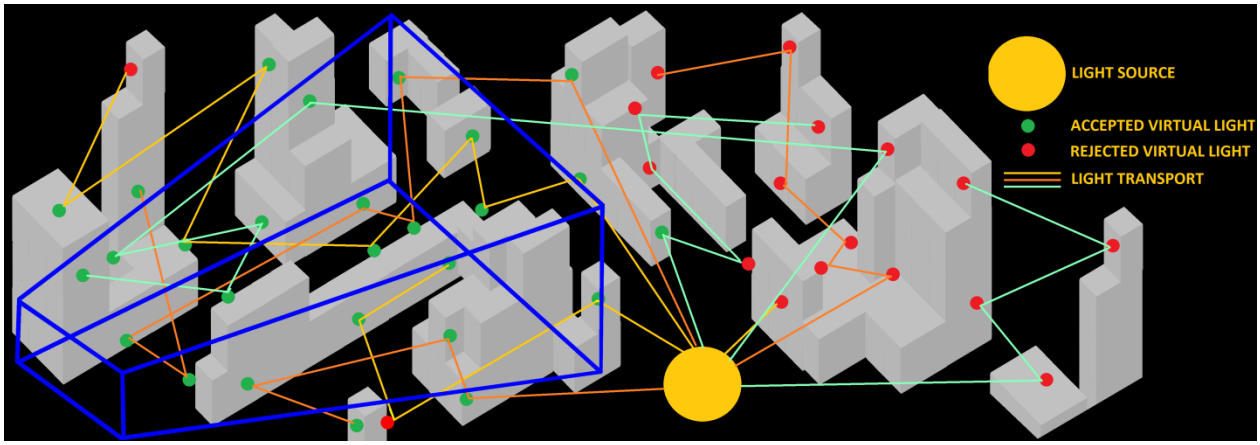

Figura 17 Generare de lumini virtuale**.**

Toate luminile secundare, luminile virtuale şi luminile scenei sunt stocate într-o structură de accelerare de tip grid tridimensional, numita cluster. Redarea cu toate aceste lumini se executa într-un proces de iluminare identic cu cel din algoritmul „Deferred Clustered", unde luminile din structura de accelerare sunt intersectate cu buffer-ul de geometrie creat de algoritmul de tip „Deferred" utilizat. Fiecare contact potențial între o lumină de orice tip şi o suprafață stocată în buffer-ul de geometrie este analizat. Pentru luminile dominante se folosesc hărțile de umbră create anterior. Pentru toate celelalte lumini se utilizează structura CIV ca suport de operator de vizibilitate, în care sunt trasate raze de determinare de umbră. Tot procesul este ilustrat în Figura 18.

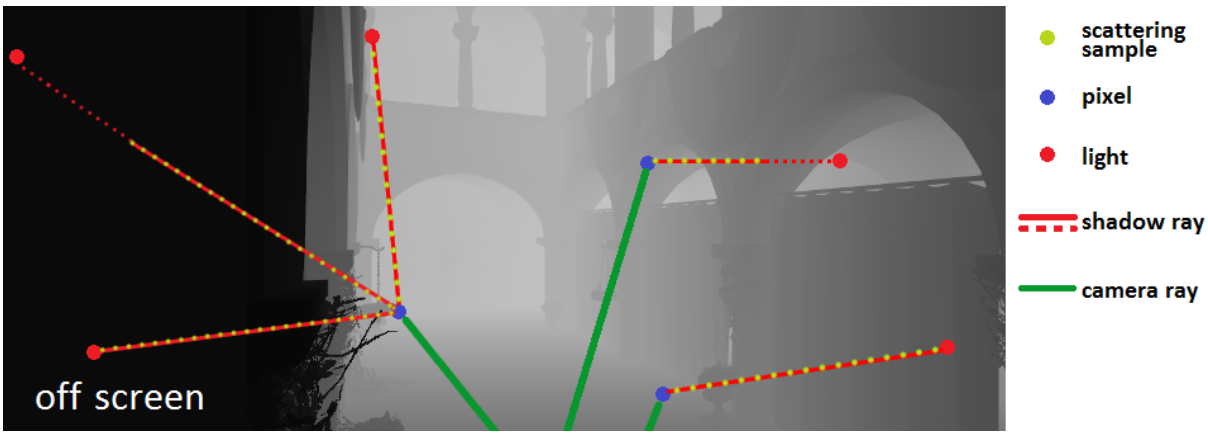

Figura 18 Iluminare şi Vizibilitate pentru lumini secundare**.**

Iluminarea cu lumini virtuale nu este lipsita de artefacte, deoarece artefactele apar ca zone de acumulare de lumina, în contrast cu artefactele din algoritmul „Path Tracing", care apar ca zgomot. Artefactele din iluminarea cu lumini virtuale pot fi observate în Figura 19. O soluție pentru această problemă este limitarea energiei luminilor virtuale de la un anumit grad de proximitate şi filtrarea iluminării în spațiul ecran, similar cu algoritmul SSPCSS. Figura 19 prezintă diferite aspecte din iluminarea cu şi fără artefacte folosind lumini virtuale generate cu metoda de „Instant Radiosity" modificată, așa cum a fost prezentată în acest capitol.

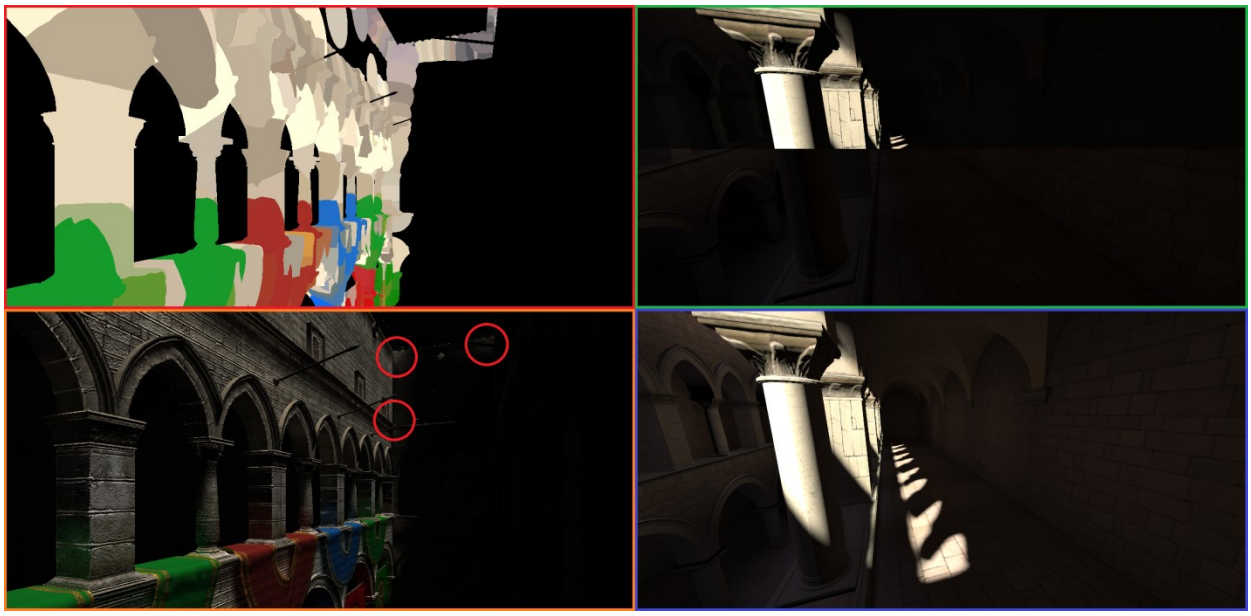

Figura 19 Rezultate iluminare cu lumini virtuale**.**

3.1.3.2. Transport de Lumină de Frecvenţă Înaltă

Transportul de lumină de înaltă frecventă este o problemă dificilă pentru redarea în timp real în scene dinamice, şi din această cauză este rezolvată în mod decuplat în această teză, deoarece metodele cu multe lumini nu sunt potrivite pentru transportul luminii speculare.

Deși transportul de lumina specular de înaltă calitate este posibil în timp real în scene statice cu multa pre procesare, algoritmul care oferă aceasta funcționalitate, „Cone Tracing" pe reprezentare volumetrică rară, are nevoie de o reprezentare volumetrica foarte detaliată, pentru a funcționa pe scene complexe. Din cauza aceasta, reprezentarea ocupă foarte multa memorie pe GPU şi mult mai important, trebuie reconstruită la fiecare cadru pentru scenele cu obiecte dinamice. Din această cauză algoritmul este doar interactiv pentru transportul de lumină specular în scene masive generale şi de aceea aceasta teza folosește proiecția algoritmului în spațiul ecran, numită „Cone Tracing" în spațiu ecran. Acest algoritm este executat după ce lumina difuză este propagată cu metoda prezentată ulterior.

„Cone Tracing" în spațiu ecran generează unul sau mai multe eșantioane per pixel, care trasează conuri peste ecran, folosind informația din buffer-ul de geometrie creat de algoritmul de de tip „Deferred" folosit în capitolul de procesare de geometrie. Direcția conurilor reflectate este obținuta prin reflectarea conurilor actuale cu suprafețele intersectate. Unghiul conurilor variază în funcție de tipul de material intersectat. Dimensiunea conurilor este dată de distanta deja trasată de fiecare eșantion. Pentru a minimiza numărul de citiri din texturile ce mențin informația ecranului, acestora li se construiesc niveluri de detaliu, din care se citește în funcție de mărimea conului.

Algoritmul „Cone Tracing" în spațiu ecran are multe cazuri de eșec, datorate următoarelor cauze: lipsa de informație cauzată de suprapunere de suprafețe în proiecția geometriei pe ecran, trasare în afara spațiului ecranului sau trasare prin spațiul din spatele

informației stocate pe ecran. Structura de accelerare CIV poate fi folosită pentru a augmenta algoritmul „Cone Tracing” pe cazurile sale de eșec.

3.1.4. Colorare de Obiecte Opace

Acest subcapitol discută operațiile de colorare de obiecte opace, în care luminile scenei şi luminile virtuale generate şi stocate în structura de accelerare de tip grid sunt folosite pentru a colora obiectele opace reprezentate în buffer-ul de geometrie obținut cu algoritmul „Deferred Virtual”.

Această metodă începe prin a încărca buffer-ul de geometrie produs de „Deferred Virtual” în mai multe țigle de lucru, una pentru fiecare grup de fire de execuție GPGPU. Acest lucru este făcut pentru a determina proprietățile de bază ale obiectelor vizibile.

După încărcarea datelor din buffer-ul de geometrie, algoritmul are doua variante de rulare, în funcție de existenta derivatelor coordonatelor de textură în buffer-ul de geometrie. Varianta algoritmului cu un consum mai mare de bandwidth stochează derivatele coordonatelor de textură per pixel, iar variantă mai intensiva din punct de vedere computațional reconstruiește derivatele coordonatelor de textura. Deoarece derivatele coordonatelor de textură sunt critice procesului de texturare, acestea trebuie neapărat obținute într-un mod sau altul.

Varianta mai intensivă computațional include operațiile variantei mai intensive din punct de vedere al bandwidth-ului, şi va fi varianta prezentată. Varianta începe cu un proces de reconstrucție al derivatelor de coordonate de texturare, care calculează diferențe finite ale coordonatelor de textura pentru fiecare pixel. Acest lucru se face prin citirea coordonatelor de texturare dintr-o anumită vecinătate, cu atenție de a nu include în proces coordonate de texturare ce provin de la obiecte care au alt index decât cel al pixelului în cauza. Astfel derivatele de coordonate de textura sunt reconstruite în același mod în care au fost inițial generate în banda grafică, prin diferențe finite la nivel de fragment. Algoritmul folosește derivatele coordonatelor de textură pentru a determina nivelurile de detaliu ale texturii ce sunt necesare pentru a efectua citirea din textură.

După ce texturile au fost citite şi rezultatele au fost stocate într-un cache la nivelul grupului de fire de execuție GPGPU, algoritmul calculează iluminarea. Mai întâi determină nivelul de iluminare dobândit de la luminile dominante, folosind operatorul de determinare de vizibilitate oferit de hărțile de umbră. Pentru luminile secundare ale scenei şi pentru luminile virtuale se folosește operatorul de vizibilitate oferit de structura de voxelizare inexactă conservativă (CIV). Astfel, pentru fiecare potențială interacțiune între o suprafața şi o lumină secundară se trasează o raza prin CIV care determină contribuția luminii.

După această etapă se calculează contribuția de iluminare provenita din transportul de lumină specular. Aceasta este o etapă opțională şi poate fi nefolosită daca nu exista potențialul de transport specular în scenă. Această etapă este calculată în spațiu ecran, bazat pe rezultatele de colorare obținute de la prima etapă de colorare cu lumina directă. Iluminarea indirectă speculară este calculată cu algoritmul „Cone Tracing” în spațiu ecran, ce folosește structura CIV pentru a ameliora artefactele rezultate din multele cazuri de eșec ale algoritmului.

3.1.5. Antialiasing Sub Pixel Decuplat

Aceasta teza prezintă o îmbunătățire pentru metodele hibride de antialiasing, care decuplează complet bandwidth-ul de colorare de operațiile de determinare de vizibilitate, totul din punctul de vedere al procesului de antialiasing. Metoda prezentată, antialiasing sub pixel decuplat, colorează o singura data per pixel şi foloseşte rezultatele colorate pentru a reconstrui o mulțime de eșantioane de determinare de vizibilitate ce lucrează la nivel sub pixel. Algoritmul apoi foloseşte eșantioanele reconstruite într-un proces de filtrare. Metoda prezentată este o îmbunătățire peste algoritmul de ultimă oră SRAA, din cauză că etapa de reconstrucție este bazată pe un proces mai exact de mapare a eșantioanelor la vecini, şi nu este doar un simplu filtru bilateral.

Antialiasing sub pixel decuplat (DSRAA) poate fi uşor de utilizat cu orice alt algoritm de tip „Deferred". Metoda prezentată are două etape. În prima etapă, numita etapă de eşantionare, algoritmul este integrat în banda de redare cu „Deferred Virtual", pe care o modifica puțin pentru a genera mai multe eşantioane de determinare de vizibilitate per pixel. Astfel, algoritmul „Deferred" modificat salvează toate datele per pixel o singură dată în afara de determinarea de vizibilitate care este salvată la o rată semnificativ mai mare, deci algoritmul are doar o creştere mică în spațiu de stocare şi bandwidth consumat faţă de un algoritm „Deferred Virtual" normal.

În cea de doua etapa, numită etapa de reconstrucție, algoritmul ia toate eşantioanele de vizibilitate necolorate şi încearcă să le mapeze întâi la eşantionul colorat al pixelului şi apoi la eşantioanele colorate ale pixelilor vecini. Maparea se face ținând cont de index-ul materialului şi de direcția normalei. Daca un eşantion nu poate fi mapat la nici un vecin colorat, atunci este mapat la fundal. În final, toate eşantioanele sunt colorate prin mapare şi culoarea finală a pixelului este obținută printr-o medie ponderată.

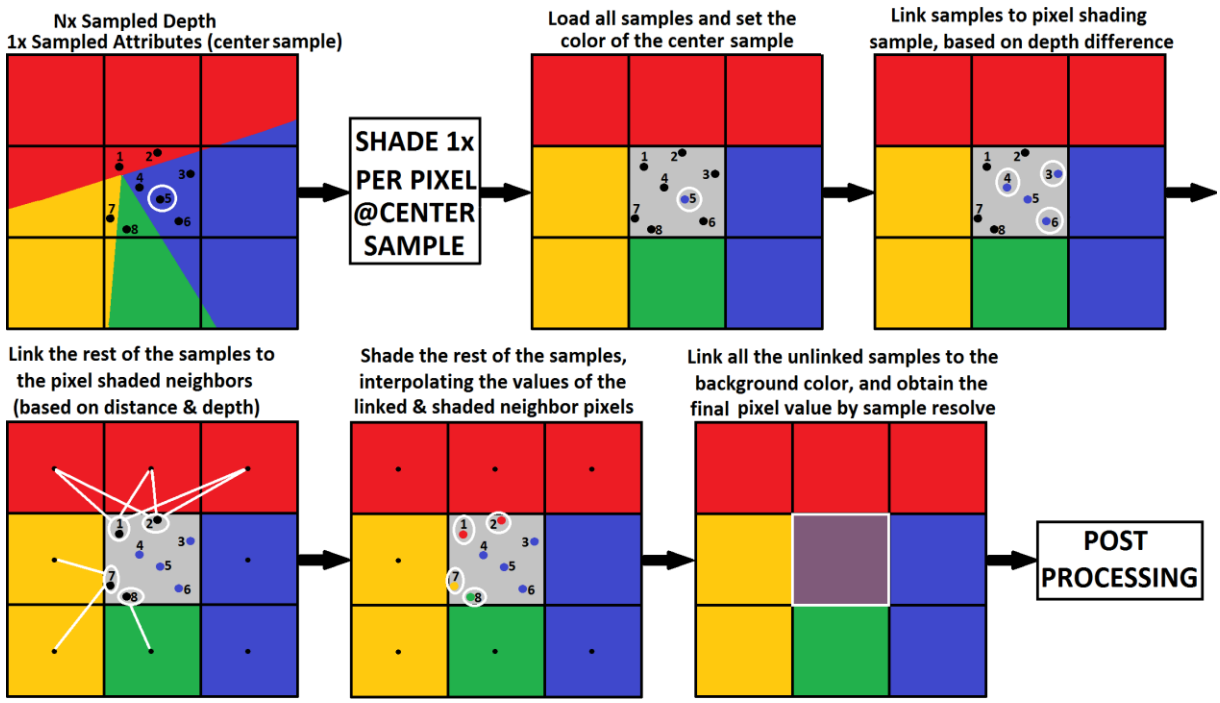

Figura 20 Antialiasing reconstruit sub geometric decuplat**.**

O comparație a algoritmului introdus cu metodele de ultimă oră de antialiasing este oferita în Tabelul 6. În Tabel se poate observa cum algoritmul introdus, DSRAA, combina marea majoritate a celor mai utile proprietăți ale algoritmilor de antialiasing pentru metode de tip „Deferred”.

Algoritm\Cantitate per pixel	Eșantion Adâncime	Eșantion Acoperire	Date Geometrie	Date Colorare	Spațiu stocare	Bandwidth
Fără Antialiasing	1	0	1	1	1	1
Multisampling Antialiasing (MSAA)	+++	+++	1	1	+	+
Coverage Sampling (CSAA/EQAA)	+++	+++++	1	1	+	+
Supersampling Antialiasing (SSAA)	+++++	0	+++++	+++++	+++++	+++++
Deferred MSAA	+++++	0	+++++	+++++	+++++	+++++
Fast Approximate Antialiasing (FXAA	1	0	1	1	1	+
Morphological Antialiasing (MLAA)	1	0	1	1	1	+
Subpixel Morphological Antialiasing (SMAA)	+	0	+	+	+	+++
Directionally Localized Antialiasing (DLAA)	1	0	1	1	1	+
Geometry Buffer Antialiasing (GBAA)	1	0	+++++	1	+	+
Distance to Edge Antialiasing (DEAA)	1	0	+++	1	+	+
Phone Wire Antialiasing (PWAA)	1	0	+++	1	+	+
Temporal Antialiasing (TXAA)	1	0	1	1	+++	+++
Aggregate G-Buffer Antialiasing (AGAA)	1	+	1	1	1	+
Surface Based Antialiasing (SBAA)	+++	+++	+++	1	+++	+++
Subpixel Reconstruction Antialiasing (SRAA)	+++	0	1/+++	1	+/+++	+/+++
Resampling Antialiasing (RSAA)	+++	+++	+++	1	+	+
Decoupled Subpixel Reconstructed Antialiasing (DSRAA) – acest algoritm	+++	0	1/+++	1	+/+++	+/+++

Tabel 6 DSRAA și algoritmii de antialiasing de ultimă oră pentru metode „Deferred”**.**

În comparație cu algoritmul de ultimă oră SRAA, metoda prezentată are o etapă de reconstrucție superioară, în care sunt create mapări exacte între eșantioanele de determinare de vizibilitate şi eșantioanele din vecinătatea pixelului care sunt colorate, adică eșantioanele generate de algoritmul „Virtual Deferred”. Din cauza aceasta în loc să se procedeze ca în algoritmul SRAA, unde fiecare eșantion de determinare de vizibilitate este aproximat cu un filtru bilateral, algoritmul prezentat utilizează procesul de mapare prezentat ce aproximează mult mai exact valoarea fiecărui eșantion necolorat. De aceea, metoda prezentată produce rezultate de o calitate mai bună, foarte apropiate de antialiasing-ul care ar fi obținut daca s-ar folosi un număr mare de eșantioane colorate per pixel (MSAA).

Figura 21 Rezultate antialiasing reconstruit sub geometric decuplat**.**

3.1.6. Colorare de Obiecte Transparente

Acest subcapitol discută operațiile de colorare ale obiectelor transparente. Acest subcapitol prezintă algoritmul de colorare al metodei de transparentă independentă de ordine. Metoda colorează în stilul algoritmului original „A-Buffer", doar că este adaptivă din punct de vedere al costului de procesare şi din punct de vedere al bandwidth-ului consumat.

Daca derivatele coordonatelor de texturare nu sunt stocate în lista modificată de fragmente per pixel atunci acestea trebuie reconstruite. Algoritmul prezentat încărca lista de fragmente în micro țigle, care sunt similare cu țiglele folosite de „Deferred Virtual", doar că sunt mult mai mici, din cauza limitărilor în spațiul de stocare local unui grup de fire de execuție pe GPU.

Micro țiglele folosite sunt limitate la dimensiunea de 2x2 pixeli, care este dimensiunea minimă necesară pentru reconstrucția derivatelor coordonatelor de texturare. După ce fragmentele sunt încărcate în micro țigle, ele sunt sortate pe axa de adâncime a pixelului. Astfel fiecare pixel din țiglă își sortează propria lista şi apoi, unul din pixeli sortează lista parțial sortată.

Aceasta operație de sortare per-pixel şi per-țigla poate fi implementată cu algoritmi de tip „Merge Sort" sau „Quick Sort", sau cu orice alt algoritm de sortare atât timp cât sortarea este făcută fără alocări de memorie. Fiecare pixel parcurge apoi lista sortata şi lucrează doar cu fragmentele ce au fost rasterizate peste el. Pentru fiecare fragment procesat se citesc vecinii săi din lista şi, în cazul în care vecinii sunt din acelaşi obiect, se calculează diferențele finite necesare determinării derivatelor coordonatelor de texturare.

În cazul în care derivatele coordonatelor de texturare sunt stocate atunci algoritmul de transparentă independentă de ordine virtuala prezentat nu are nevoie să le reconstruiască. Algoritmul doar încărcă fragmentele fiecărui pixel fără a folosi țigle, şi sortează fragmentele încărcate.

După ce derivatele coordonatelor de texturare sunt obținute într-un mod sau altul, algoritmul parcurge listele sortate, la nivel per pixel. Parcurgerea este făcută din fată în spate, încărcând datele de texturare şi efectuând operațiile de colorare pentru fiecare nod nou. Algoritmul este adaptiv pentru că nu încărcă date de texturare şi nu procesează operații de colorare decât pentru fragmentele ce au un impact vizual garantat. O imagine redată cu algoritmul prezentat este ilustrată în Figura 22.

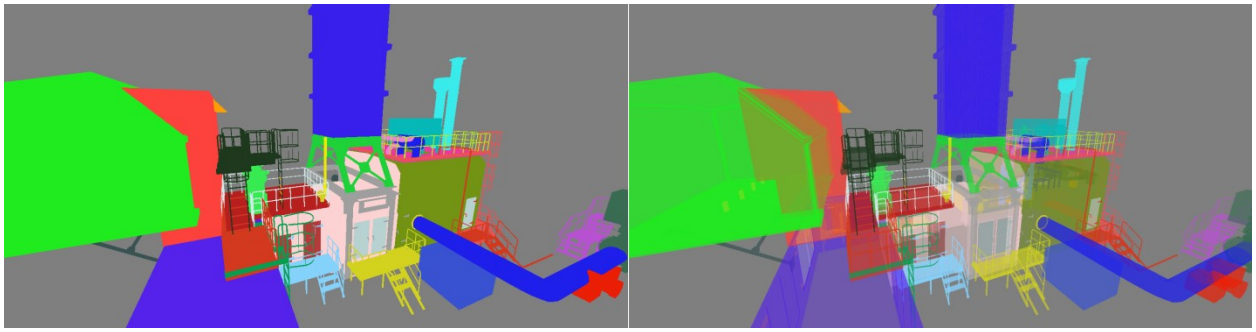

Figura 22 Colorare de Obiecte Transparente**.**

3.2. Iluminare Corectă

Modulul de iluminare corectă este un modul de redare interactiv, care conține şi prezintă algoritmi pentru care hardware-ul curent este prea puțin puternic pentru rulări în timp real. Cu toate acestea, algoritmii prezentați în acest modul sunt din ce în ce mai relevanți pentru domeniul redării în timp real.

Modulul de iluminare corecta folosește structuri de accelerare pentru „Ray Tracing", pentru a scădea complexitatea trasării de raze prin scenă, operație ce domina timpul de redare pentru algoritmii folosiți în iluminarea globală corectă. Transportul de lumina este efectuat într-un mod corect, în limitările redării fotorealiste pe calculator. Algoritmii prezentați în acest modul pot rula atât pe CPU cât şi pe GPU.

În acest modul imaginile sunt redate cu un algoritm de tip „Path Tracing Bidirecțional" modificat care trasează raze cu o complexitate amortizată. Algoritmul folosește o metodă inedită de eșantionare de importanţă, numita „Light Flux" (LFIS), care aproximează fluxul transportului de lumină în scenă şi folosește această aproximare pentru a ghida path-urile neproductive către vârfuri din path-uri de lumină. Comparat cu metodele de ultimă oră, LFIS este mai rapidă şi stochează semnificativ mai putina memorie. Algoritmul „Path Tracing Bidirecțional" folosește atât metoda inedită introdusă cât şi alte metode de accelerare de ultimă oră.

3.2.1. Determinare de Vizibilitate Amortizată

Conceptul de vizibilitate amortizată este bazat pe idea de a combina doua structuri de accelerare pentru intersecție de suprafețe cu raze: una exactă, bazată pe structura „Bounding Interval Hierarchy" şi altă inexactă, bazată pe un grid tridimensional. Se efectuează trasare de raze în structura exactă doar atunci când este dovedit că raza este un potențial contribuitor de către structura aproximativă. Razele aproximative nu pot determina niciodată exact dacă are loc o intersecție între ele şi suprafețe, dar pot aproxima potențialul acestei intersecții.

De aceea metoda de aproximare de vizibilitate nu poate fi folosita pentru a crea noi grupuri de segmente în path-uri, pentru că aceasta operație poate fi efectuată doar analitic corect. Totuși, vizibilitatea amortizata poate fi folosita pentru determinarea potențialului de legare între diverse segmente de path deja existente, operație care trebuie execută de foarte multe ori în algoritmul „Path Tracing-ul Bidirecțional", când rezultatele etapelor de „Light Tracing" şi „Camera Tracing" sunt legate.

În cazul în care o raza aproximativă arată un potențial ridicat de intersecție între rază şi suprafață atunci raza este intersectată cu structură de accelerație exactă. Intersecția cu structura de accelerație exactă nu pleacă de la zero, ci pleacă de la un interval definit de raza aproximativă care este folosit în partiționarea razei de intersecție pentru a minimiza spațiul de căutare exact. De aceea, în timpul traversării structurii de accelerație exacte nu toate segmentele razei de intersecție sunt intersectate cu nodurile structurii „Bounding Interval Hierarchy".

Deși razele productive suferă astfel de o creștere în complexitate, cum marea majoritate a razelor nu sunt productive şi sunt terminate rapid prin intersecție cu structura aproximativă, costul de intersecție per rază este amortizat.

3.2.2. Eşantionare de Importanţă Light Flux

Problema principală a algoritmilor bazați pe trasare de raze nu este neapărat complexitatea de trasare a razelor cât numărul foarte mare de raze trasate, independent de soluția aleasă de algoritmul de redare. De aceea, marea majoritate a cercetării este bazata pe reducerea numărului de raze cu metode de eșantionare avansate numite eșantionare de importanţă. Eșantionarea de importanţă generează raze de determinare de vizibilitate ce au o probabilitate mult mai mare de a explora spațiu productiv, relevant pentru interacțiunile între suprafețele şi luminile scenei.

Eșantionarea de importanţă de tip „Light Flux” (LFIS), este un nou algoritm de eșantionare de importanţă, prezentat în aceasta teza şi este construit astfel încât să fie folosit cu algoritmul de redare „Path Tracing Bidirecțional”. Idea este inspirată din hărțile de flux, folosite pentru fluide. LFIS foloseşte o hartă foarte similara ca şi concept cu hărțile de flux pentru fluide, doar că această hartă descrie fluxul de lumină în scenă. Această hartă poate fi citită pentru a determina rapid sursa de lumină cea mai relevantă pentru fiecare punct din scenă. Sursa de lumină poate fi orice lumină din scenă, cât şi vârfurile din path-urile construite la etapa de „Light Tracing” din cadrul algoritmului „Path Tracing Bidirecțional”.

Harta „Light Flux” este implementată ca un grid tridimensional. Este populata prin generarea unui număr de eșantioane pentru fiecare lumină şi generarea de path-uri de lumină pentru aceste eșantioane. Path-urile de lumina sunt apoi voxelizate în interiorul grid-ului tridimensional din harta „Light Flux”. Fiecare intrare din grid peste care path-ul de lumină este voxelizat salvează o referința la lumină sau vârful din path-ul de lumină care a catalizat intrarea în grid. Daca o intrare este deja populată, se utilizează un proces de determinare a celui mai important contribuitor de lumină, pe baza radiantei stocate în grid şi radiantei provenite de la sursa de lumină nouă. Harta „Light Flux” conține astfel informație parțiala despre transportul de lumină din scenă. Se foloseşte un proces „Push-Pull” tridimensional, similar cu cel folosit în Voxelizarea Conservativă Inexactă în capitolul 3.1.2, cu care este populată întreaga hartă, care garantează că fiecare intrare din harta „Light Flux” referențiază o sursa de lumină. Pentru intrările aproximate din harta „Light Flux” se utilizează o mască, care setează un bit special.

Astfel, harta „Light Flux” permite eșantionare de importanţă prin faptul că este capabilă să lege orice vârf creat în etapa de „Camera Tracing” a algoritmului de „Path Tracing Bidirecțional” cu o sursa de lumină. Astfel, harta „Light Flux” poate fi folosita pentru a construi legături rapide între path-uri de explorare a scenei din perspectiva camerei ce nu au găsit o sursă de lumină după un număr relativ mare de iterații.

În etapa de „Light Tracing” a algoritmul de „Path Tracing Bidirecțional”, harta de „Light Flux” este actualizată. Intrările care au bitul special de aproximare setat sunt actualizate în cazul în care vârfuri create de path-urile construite la aceasta etapă ajung să fie generate în spațiul mapat către intrarea aproximată din hartă. În acest mod, harta originală aproximată converge relativ rapid la o hartă exactă a transportului de lumină în scenă. Din acest motiv, „Light Flux” permite conexiuni rapide intre etapele algoritmului „Path Tracing Bidirecțional”, făcându-l mai eficient şi astfel mai apropiat de performantele necesare pentru execuția în timp real.

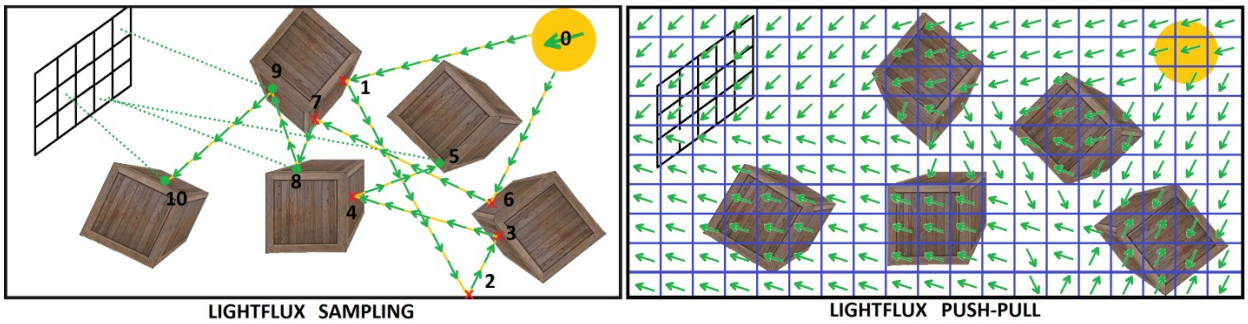

Figura 23 Eşantionare de Importanţă Light Llux**.**

Eşantionarea de Importanţă cu „Light Flux" este prezentată pe scurt în Figura 23, unde în stânga sunt prezentate putinele eşantioane necesare pentru construcţia hărţii, iar în dreapta este prezentat procesul de tip „Push – Pull" tridimensional, în urma căruia rezultatele obţinute cu puţine eşantioane sunt interpolate pentru a oferi o aproximare a transportului de lumină la nivel global al scenei.

Metoda de Eşantionare de Importanţă &	Rezultate bune cu eşantion slab	Complexita-te	Spaţiu Stocare	Necesita Mutaţii pe Vârfuri	Explorare de path-uri speculare	Cunoştinţe explicite asupra luminilor
Eşantionare adaptivă	Slabe	Mică	Putină	Nu	Nu	nu
Eşantionare BSDF	Nu	Mică	Deloc	Nu	Da	N
Iluminare Directă	Nu	Mică	Deloc	Nu	Nu	Da
Eşantionare de Importanţă re-Eşantionata	Nu	Mică	Putină	Nu	Nu	Nu
Eşantionare volumetrica	Nu	Medie	Deloc	Nu	Nu	Nu
Re-proiecţie la camera	Nu	Mică	Deloc	Nu	Nu	Nu
Filtrare de Radianţă	Nu	Mică	Medie	Nu	Nu	Nu
Reconstrucţie de câmp de lumină temporală	Nu	Ridicată	Medie	Da	Nu	Nu
Transport de lumină Metropolis	Medii	Ridicată	Ridicată	Da	Medie	Nu
Transport de lumină Metropolis în spaţiu primar	Ridicate	Ridicată	Ridicată	Da	Medie	Nu
Redistribuie de energie	Ridicate	Medie	Ridicată	Da	Medie	Nu
Explorare de Manifold	Medii	Ridicată	Ridicată	Da	F. Bună	Nu
Metropolis în domeniu gradient	Ridicate	Ridicată	Ridicată	Da	Bună	Nu
Metropolis multiplexat	Ridicate	Ridicată	Ridicată	Da	F. Bună	Nu
Legare de conexiuni de vârfuri	Ridicate	Ridicată	Ridicată	Nu	Bună	Nu
Regularizare în spaţiu path	Ridicate	Ridicată	Ridicată	Nu	Bună	Nu
Bidirecţional	Slabe	Medie	Ridicată	Nu	Nu	Da
Skeleton	Medii	Slabă	Medie	Nu	Nu	Nu
Light Flux - acest algoritm	Ridicate	Slabă	Medie	Nu	Nu	Da

Tabel 7 Strategii de eşantionare pentru algoritmul „Path Tracing"**.**

Algoritmul „Light Flux" de eşantionare de importanţă este comparat cu alţi algoritmi de eşantionare de importanţă de ultimă oră în Tabelul 7. În tabel se poate observa cu „Light Flux" are rezultate bune cu eşantioane slabe în condiţiile în care nu necesita un număr ridicat de operaţii sau un spaţiu de stocare ridicat. Mai mult, „Light Flux" nu produce mutaţii la nivel de path şi este singurul algoritm de eşantionare de importanţă global care oferă informaţii directe asupra iluminării. Celalalt algoritm de eşantionare de importanţă globală oferă informaţii indirecte, prin faptul că speculează prezenta luminii prin absenta geometriei.

3.2.3. Path Tracing Bidirecțional

Algoritmul „Path Tracing Bidirecțional” (BDPT) este o variantă a algoritmului „Path Tracing” care oferă cele mai bune rezultate fără a utiliza mutații pe path-urile create. Din această cauza este una dintre cele mai bune variante pentru transportul fotorealistic de lumină pe arhitecturile de tip „many core”, cum sunt GPU-urile. Algoritmul BDPT rulează în doua etape: etapa de „Light Tracing” și etapa de „Camera Tracing”. În etapa de „Light Tracing” fiecare lumină din scenă este eșantionată, și pentru fiecare eșantion generat se creează un path construit din vârfuri, obținute la contactul dintre transportul de lumină și suprafețele scenei. Același proces se aplică pentru etapa de „Camera Tracing”. BDPT este integrabil cu algoritmii de tip „Deferred”. Path-urile generate în cele două etape sunt apoi legate, creând transport continuu de la lumini la cameră. Conexiunea între cele doua etape este de obicei făcuta cu o structură de tip hash ierarhic peste un grid spațial.

Algoritmul BDPT poate utiliza metoda de eșantionare de importanță prezentată în sub-capitolul anterior. Așa cum este arătat în Figura 23, procesul de construcție al hărții „Light Flux” poate fi folosit pentru a aproxima cea mai productivă direcție de eșantionare pentru fiecare lumină a scenei. Acesta direcție este folosita în algoritmul BDPT cu „Light Flux”, astfel implementându-se eșantionare de importanță pe eșantionarea luminilor.Banda de redare pentru iluminare corecta prezentată mai folosește eșantionare de importanță pentru funcțiile de distribuție de împrăștiere bidirecționale (BSDF), care sunt eșantionate la fiecare contact intre lumină și o suprafață a scenei. În plus, BDPT-ul din teză folosește eșantionare adaptivă, reproiecție către cameră, filtrare de radiantă și metoda de ruletă rusească pentru terminarea path-urilor. Algoritmul „Path Tracing Bidirecțional cu Light Flux” este prezentat pe scurt în Figura 24.

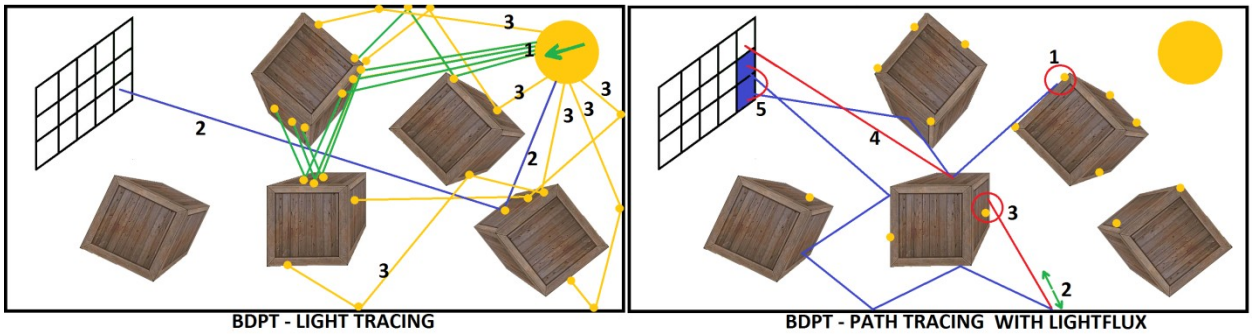

Figura 24 Path Tracing Bidirecțional cu eșantionare de importanță Light Flux**.**

BDPT-ul prezentat poate folosi doua strategii pentru a eșantiona din harta „Light Flux”. Prima strategie este de a umple harta aproximată „Light Flux” cu intrări în vecinătatea vârfurilor de lumină generate în etapa de „Camera Tracing”. A doua strategie, mai exactă dar mai scumpă, este de a utiliza harta inițială aproximativa și a ii aduce eșantioane adiționale, în timpul procesului de „Light Tracing”. Acest lucru duce la convergența rapidă a hărții. Etapa de „Camera Tracing” din BDPT eșantionează scena ca algoritmul „Path Tracing” normal, dar în același timp încearcă să conecteze path-urile generate la această etapă cu path-urile generate la etapa de „Light Tracing” prin utilizarea hărții „Light Flux” pentru a găsi rapid vârfuri ce aparțin de path-urile create la etapa de „Light Tracing”. Diferențele de rulare cu și fără „Light Flux” sunt prezentate în Figura 25. Se poate observa cu algoritmul BDPT are o convergență superioară dacă este augmentat cu harta „Light Flux”, în special în scene cu transfer dificil de lumină.

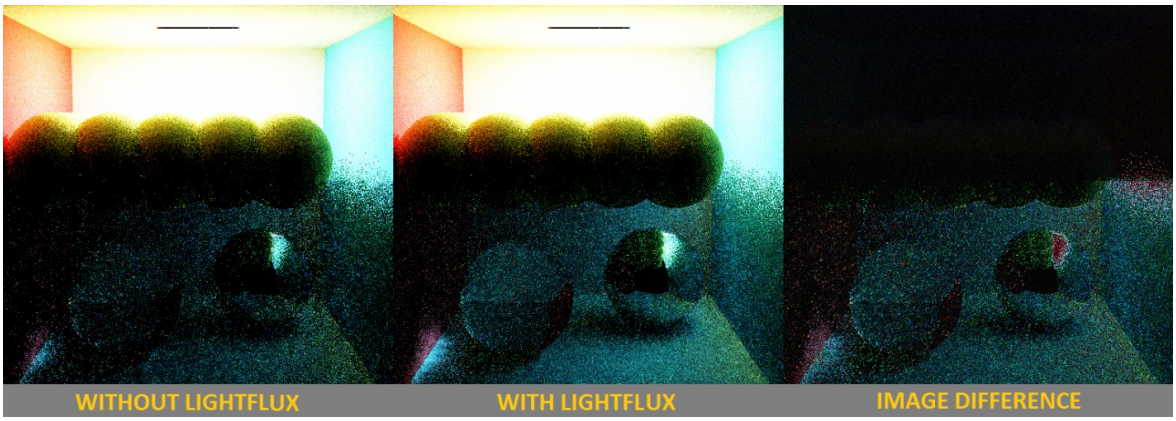

Figura 25 Rezultate BDPT interactiv cu eşantionare de importanţă Light Flux**.**

4. CONCLUZII

Această teză a introdus o bandă de redare modulară, bazată pe mai mulți algoritmi inediți, care funcționează în timp real sau interactiv. Algoritmii prezentați sunt bazați pe principiile de decuplare şi de reducere de bandwidth, şi pot simula multe tipuri de transport de lumină.

Algoritmii decuplați prezentați în această teză separă complet procesul de determinare de vizibilitate, citirea din texturi şi metodele de colorare, toate acestea în contextul desenării cu rasterizare. Acest lucru este efectuat fără stocarea unui număr mare de eșantioane şi fără a sincroniza soluții bazate pe programare dinamică. Soluțiile decuplate oferite cresc ușurința analizei surselor de aliasing, stabilitatea redării şi dezvoltarea şi menținerea software-ului care le implementează. Principiile decuplării sunt aplicate pentru redarea cu rasterizare a obiectelor opace şi transparente, în doi noi algoritmi: „Deferred Virtual" şi „Transparentă Independentă de Ordine Virtuală", care oferă rezultate superioare fată de algoritmii de ultimă oră. De asemenea, un nou algoritm de anti aliasing este introdus în teza, bazat pe aceleaşi principii de decuplare. Algoritmul este special construit pentru metodele de tip „Deferred", şi oferă un proces de reconstrucție superior metodei de ultimă oră pe care o îmbunătățește.

Voxelizarea conservativă inexactă este un tip nou de voxelizare şi se distinge de metodele de ultimă oră prin faptul că are o complexitate de construcție de $O(obiecte)$ în loc de $O(primitive)$. Această nouă structură de date poate fi folosită în transportul aproximativ de lumină în timp real, mai ales pentru lumina de frecvenţă joasă. Metoda rezolvă problema dificilă a redării cu mult umbre în timp real, prin faptul că poate fi folosită ca un suport pentru trasare de raze de vizibilitate. Voxelizarea inexactă conservativă este adaptată în această teză pentru a fi folosită împreună cu metode de transport aproximativ de lumina specular, augmentând cazurile de eşec ale algoritmilor de ultimă oră. Acest rezultat este obținut fără consumurile foarte mari de spațiu de stocare şi de bandwidth ce sunt prezente în soluțiile de ultimă oră pentru transportul de lumină de înaltă frecvenţă.

Teza introduce de asemenea conceptul de operator de vizibilitate adaptiv, care scade ca şi calitate în zonele cele mai dificil de perceput ale scenei şi care permite simularea transportului de lumină în timp real cu rezultate comparabile cu tehnici de ultimă oră, cu costuri computaționale mult mai mari.

Eşantionarea de importanţă „Light Flux” este o altă contribuție a tezei, ce accelerează algoritmul „Path Tracing Bidirecțional”. Comparat cu celelalte metode de eșantionare de importanţă de ultimă oră, „Light Flux” oferă informații directe asupra fluxului global de lumina. Astfel „Light Flux” ghidează explorările de lumină neproductive fără a specula natura transportului de lumina din scenă, așa cum este făcut în metodele de ultimă oră. „Light Flux” creşte rata de convergenţă a algoritmului „Path Tracing Bidirecțional”.

Teza mai introduce şi alte contribuții precum un algoritm de reconstrucție de seturi de date volumetrice masive, impostori ierarhici pentru scene foarte mari, generare de sarcini şi metode noi de paralelizare pe GPU, algoritmi de decupare pe cadre multiple, metrici de analiză a metodelor de „Deferred” şi hărți de ocupare cu distribuții. Acestea sunt descrise pe larg în teza.

BIBLIOGRAFIE

[Pet11] L. Petrescu, A. Morar, F. Moldoveanu, and V. Asavei, "Real time reconstruction of volumes from very large datasets using CUDA", in *15th International Conference on System Theory, Control, and Computing (ICSTCC)*, pp. 1-5, 2011.

[SAB15] SABIMAS. (2015, Aug.) Personalized implants for hip arthroplasty, (SABIMAS, PNCDII-Joint Applied Research Projects, 2008-2011). [Online]. HYPERLINK "http://se.cs.pub.ro/SABIMAS/" http://se.cs.pub.ro/SABIMAS/

[Mor13] A. Morar, F. Moldoveanu, V. Asavei, L. Petrescu, A. Moldoveanu, and A. Egner, "GPGPU Based Non-photorealistic Rendering of Volume Data", *Control Engineering and Applied Informatics (CEAI)*, vol. 15, no. 1, pp. 45-52, 2013.

[Pet14] L. Petrescu, F. Moldoveanu, V. Asavei, A. Moldoveanu, and O. Ferche, "A GPU Task Generator for Rendering", in *ICSTCC 2014 - 18th International Conference On System Theory, Control and Computing*, pp. 562-567, 2014.

[Pet151] L. Petrescu, F. Moldoveanu, V. Asavei, and A. Moldoveanu, "Virtual deferred rendering", in *20th International Conference on Control Systems and Computer Science (CSCS)*, pp. 373-378, 2015.

[Pet152] L. Petrescu, F. Moldoveanu, V. Asavei, and A. Moldoveanu, "Guarded Order Independent Transparency", *Scientific Bulletin of University POLITEHNICA of Bucharest, Series C, Electrical Engineering and Computer Science*, vol. 77, no. 1, pp. 3-14, Apr. 2015.

[Pet13] L. Petrescu, F. Moldoveanu, A. Moldoveanu, A. Morar, and V. Asavei, "Efficient Picking Through Atomic Operations", in *19th International Conference on Control Systems and Computer Science (CSCS)*, pp. 66-70, 2013.

[Pet15] L. Petrescu, F. Moldoveanu, V. Asavei, and A. Moldoveanu, "Analyzing Deferred Rendering Techniques", *Control Engineering and Applied Informatics (CEAI)*, accepted, to be published, 2015.