

UNIVERSITATEA POLITEHNICA DIN BUCUREŞTI
FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DEPARTAMENTUL DE CALCULATOARE



PROIECT DE DIPLOMĂ

Implementarea algoritmului Ray Tracing
folosind arhitectura DirectX Raytracing

Alex-Andrei Cioc

Coordonator științific:

Conf. Dr. Ing. Victor Asavei

BUCUREŞTI
2024

UNIVERSITY POLITEHNICA OF BUCHAREST
FACULTY OF AUTOMATIC CONTROL AND COMPUTERS
COMPUTER SCIENCE AND ENGINEERING DEPARTMENT



DIPLOMA PROJECT

Implementation of the Ray Tracing algorithm
using the DirectX Raytracing architecture

Alex-Andrei Cioc

Thesis advisor:

Conf. Dr. Ing. Victor Asavei

BUCHAREST

2024

CUPRINS

1 Introducere	1
1.1 Context	1
1.2 Problema	1
1.3 Obiective	3
1.4 Soluția propusă	4
1.5 Rezultatele obținute	5
1.6 Structura lucrării	6
2 Motivație de cercetare	7
3 Metode Existente	9
3.1 Fundamente și Terminologie	9
3.1.1 Metode Monte Carlo	9
3.1.2 Integrare Monte Carlo	9
3.1.3 Importance Sampling	10
3.1.4 Eșantionare Stratificată	11
3.2 Ecuația transportului luminii	12
4 Solutia Propusă	13
5 Detalii de implementare	14
5.1 Indicații formatare tabele	14
6 Evaluare	15
7 Concluzii	16
Anexe	18

SINOPSIS

Algoritmul Ray Tracing este o tehnică de randare a imaginilor care simulează propagarea și comportamentul razelor de lumină într-o scenă tridimensională. Acesta este adesea folosit în industria cinematografică pentru a obține imagini fotorealiste. Până de curând, natura computațională intensivă a acestui algoritm a limitat utilizarea sa în aplicații interactive, precum jocurile video. Totuși, cu avansul tehnologiei, utilizarea acestuia a devenit tot mai accesibilă și pentru aceste aplicații. Suport hardware pentru Ray Tracing în contextul consumatorilor a fost introdus de NVIDIA în 2018, prin intermediul arhitecturii Turing.¹ Tot în același an, Microsoft a anunțat DirectX Raytracing² (DXR), o extensie a API-ului DirectX 12 care permite programatorilor să folosească Ray Tracing în aplicațiile lor, utilizând hardware-ul compatibil.

Lucrarea de față își propune să studieze și să implementeze algoritmul Ray Tracing pe un sistem de calcul modern, folosind accelerarea hardware oferită de arhitectura DirectX Raytracing. Acest algoritm va fi folosit pentru randarea iluminării unor scene arbitrate, în timp real, oferind o reprezentare fotorealistă a acestora. În cadrul lucrării se va realiza o analiză a performanțelor implementării curente, atât a fidelității imaginilor generate, cât și a eficienței spațio-temporale a implementării. Se vor explora și posibilitățile de optimizare a algoritmului, precum și modul în care acestea pot fi folosite pentru a îmbunătăți performanțele sistemului.

ABSTRACT

The Ray Tracing algorithm is an image rendering technique that simulates the propagation and behavior of light rays in a three-dimensional scene. It is often used in the film industry to achieve photorealistic images. Until recently, the computationally intensive nature of this algorithm has limited its use in interactive applications, such as video games. However, with technological advances, its use has become increasingly accessible for these applications as well. Hardware support for Ray Tracing in the consumer context was introduced by NVIDIA in 2018, through the Turing¹ architecture. In the same year, Microsoft announced DirectX Raytracing² (DXR), an extension of the DirectX 12 API that allows programmers to use Ray Tracing in their applications, using compatible hardware.

This paper aims to study and implement the Ray Tracing algorithm on a modern computing system, using the hardware acceleration provided by the DirectX Raytracing architecture. This algorithm will be used for rendering the lighting of arbitrary scenes, in real-time, providing a photorealistic representation of them. The paper will perform an analysis of the current implementation's performance, both in terms of the fidelity of the generated images and the spatio-temporal efficiency of the implementation. It will also explore the possibilities for optimizing the algorithm, as well as how these can be used to improve the system's performance.

¹<https://images.nvidia.com/aem-dam/en-zz/Solutions/design-visualization/technologies/turing-architecture/NVIDIA-Turing-Architecture-Whitepaper.pdf>. Accesat 10.06.2024.

²<https://devblogs.microsoft.com/directx/announcing-microsoft-directx-raytracing/>. Accesat 10.06.2024.

MULTUMIRI

Adresez mulțumiri coordonatorului meu de proiect, Conf. Dr. Ing. Victor Asavei, pentru îndrumarea și sprijinul acordat pe parcursul realizării acestei lucrări, dar și pentru inspirația și motivația oferită în cadrul cursurilor de Elemente de Grafică pe Calculator. De asemenea, mulțumesc familiei și prietenilor pentru susținere și încurajare.

1 INTRODUCERE

1.1 Context

Industria jocurilor video este una dintre cele mai mari și mai profitabile industrii de divertisment din lume. În 2021, piața jocurilor video era evaluată la aproximativ 202.64 miliarde de dolari și este estimat să se extindă la o rată anuală compusă de creștere de 10.2% în perioada 2022-2030.¹ Această industrie este alimentată de cererea pentru experiențe interactive și captivante, care să ofere o experiență de joc cât mai realistă și cât mai imersivă. Toate studio-urile de dezvoltare de jocuri video AAA (i.e., jocuri cu bugete mari și echipe de dezvoltare extinse) investesc resurse semnificative în dezvoltarea de tehnologii care să le permită să creeze jocuri cu grafică de înaltă calitate. Aceste tehnologii includ motoare grafice puternice, care să permită randarea unor scene complexe, cu iluminare realistă și efecte speciale impresionante. Multe studio-uri folosesc propriile motoare dezvoltate in-house (e.g., Frostbite de la EA, CryEngine de la Crytek, Anvil de la Ubisoft), dar există și motoare comerciale, precum Unreal Engine și Unity. Aceste motoare oferă un set de instrumente și funcționalități care permit dezvoltatorilor să creeze jocuri video de înaltă calitate, fără a fi nevoie să dezvolte de la zero toate componentele necesare. O componentă critică a acestor motoare este motorul grafic, care se ocupă de randarea scenei jocului, de la geometria obiectelor până la iluminare și efecte speciale. Astfel, programatorii, artiștii, animatorii și designerii de jocuri pot să se concentreze pe crearea conținutului jocului, fără a fi nevoie să se ocupe de detalii tehnice ale randării grafice.

1.2 Problema

Tehnica tradițională și cea mai răspândită de randare a imaginilor în jocurile video este rasterizarea. Această tehnică se bazează pe proiecția obiectelor 3D pe un plan bidimensional, folosind o serie de algoritmi și tehnici pentru a simula iluminarea și efectele speciale. Rasterizarea este o tehnică eficientă și rapidă, care permite randarea unui număr mare de obiecte în timp real, dar are și limitări. Una dintre cele mai mari limitări ale rasterizării este incapacitatea de a simula iluminarea globală, care este esențială pentru obținerea unor imagini fotorealiste. De asemenea, reflexiile și refractiile pot fi doar approximate, de exemplu prin tehnici de cubemapping sau screen-space reflections. Aceste tehnici sunt eficiente, dar nu oferă rezultate realistice, iar în multe cazuri pot fi observate artefacte vizuale care afectează calitatea imaginii.

În continuare se evidențiază aceste limitări (care nu sunt deloc exhaustive) ale rasterizării, prin

¹<https://www.grandviewresearch.com/industry-analysis/gaming-industry>. Accesat 10.06.2024.

comparație cu tehnica de Ray Tracing (așa numita *RTX* în jocurile sponsorizate de Nvidia). Comparând Figurile 1 și 2, se observă neajunsul reflexiilor în screen space. Atâtă timp cât obiectele reflectate se află în viewport, reflexiile sunt corecte și realiste. Însă, dacă obiectele ies din viewport, reflexiile se pierd, ceea ce duce la o imagine nerealistă. Un alt exemplu și mai elovent este ilustrat în Figura 4, unde imaginea randată cu Ray Tracing redă reflexii ale exploziei care nu este vizibilă decât parțial în cadru.



Figura 1: Screen space reflections în Minecraft²

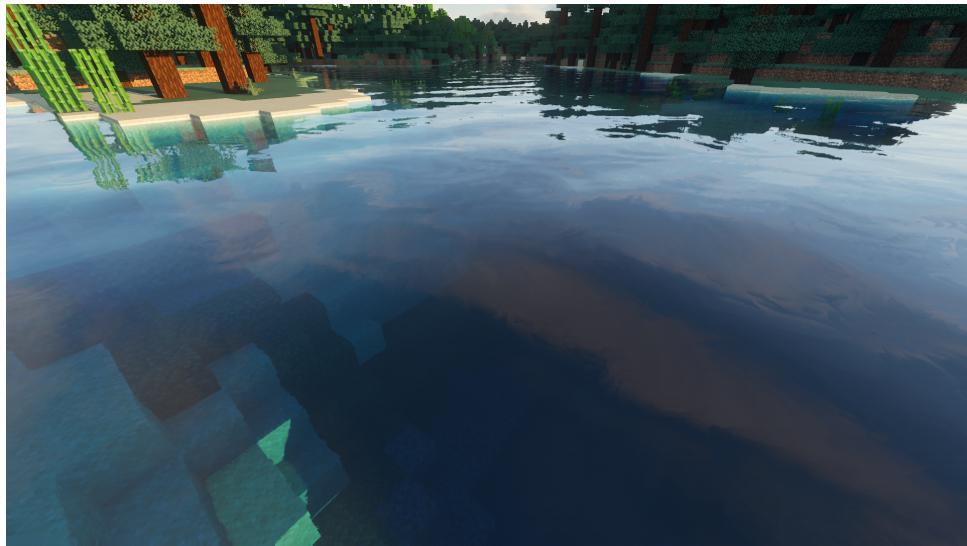


Figura 2: Artefacte vizuale în screen space reflections.² Se poate observa cum reflexiile se pierd dacă obiectele reflectate ies din viewport

Am văzut cum tehnica de Ray Tracing poate oferi rezultate mult mai realiste decât rasterizarea, dar această tehnologie vine cu un cost. Algoritmul Ray Tracing este computațional intensiv,

²Shader folosit: <https://continuum.graphics/>. Accesat 10.06.2024.

³©Nvidia Corporation: <https://blogs.nvidia.com/blog/geforce-rtx-real-time-ray-tracing/>. Accesat 10.06.2024.

⁴©Nvidia Corporation: <https://www.youtube.com/watch?v=WoQr0k2IA9A>. Accesat 10.06.2024.

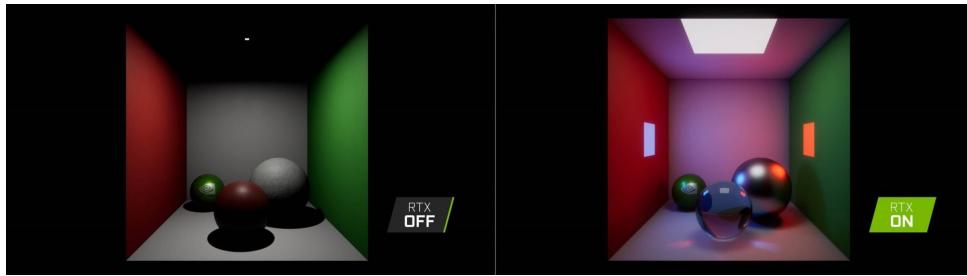


Figura 3: Iluminarea globală este absentă în imaginea randată cu rasterizare³

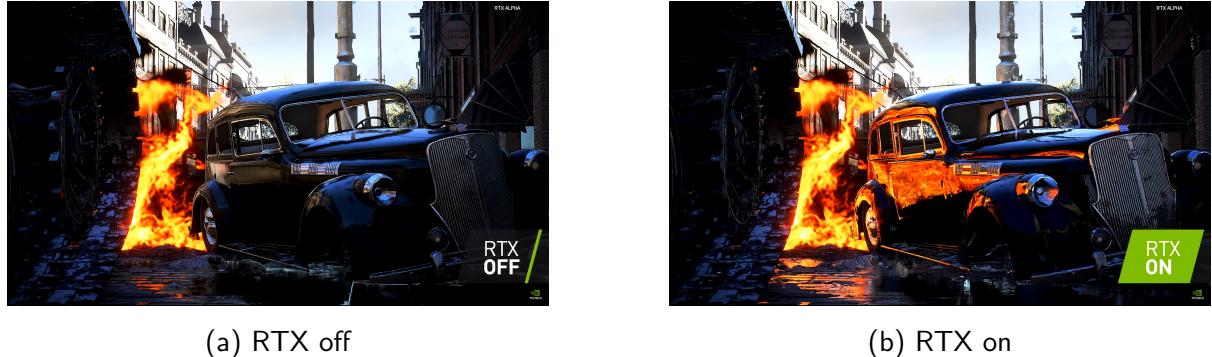


Figura 4: Comparație RTX on/off în Battlefield V⁴

deoarece necesită calcularea intersecțiilor mai multor raze de lumină pentru fiecare pixel cu obiectele din scenă și calcularea contribuției acestora la culoarea pixelului. Pentru a obține o imagine de calitate, este nevoie de un număr mare de raze de lumină și tehnici de denoising, ceea ce face ca algoritmul să fie greu de balansat între fidelitate și performanță.

1.3 Obiective

Scopul acestei lucrări este de a cerceta și implementa algoritmul Ray Tracing în context de timp real și a evalua performanțele acestuia, prin comparație cu implementări destinate producției cinematografice. Eforturile vor fi concentrate pe implementarea unui motor grafic simplu, cu câteva funcționalități de bază:

- Controale de cameră simple (mișcare, rotere)
- Randarea obiectelor definite ca mesh-uri triunghiulare
- Randarea obiectelor definite prin funcții implicate (e.g., sfere, planuri)
- Suport pentru materiale PBR (Physically Based Rendering)
- Suport pentru mai multe scene precum:
 - Cornell Box⁵

⁵<https://www.graphics.cornell.edu/online/box/>. Accesat 19.06.2024.

- Demonstrație a suprafețelor implicate
- Scenă de testare a materialelor PBR
- Meniu de configurare a mai multor parametri (e.g., pentru materiale, configurarea algoritmului etc.),

precum și a unor efecte de iluminare care să ofere o imagine fotorealistică a scenei:

- Iluminare globală
- Reflexii și refracții
- Umbre.

De asemenea, vom explora și implementa tehnici de optimizare a algoritmului propriu-zis, pe care le vom evalua în contextul de performanță și fidelitate obținute.

În final, dorim să obținem o implementare eficientă, care să țintească un frametime de 16.6ms (60 de cadre pe secundă) pentru hardware entry-level cu suport pentru DirectX Raytracing (e.g., Nvidia GeForce RTX 2060).

Adresându-mă direct cititorului, această lucrare este scrisă cu scopul de a desemna o introducere asupra conceptelor și metodelor prezentate. Nu se adresează experților în domeniu, ci mai degrabă celor care se află la primii pași de învățare. Sper ca explicațiile să fie intuitive și să ofere o privire de ansamblu, însă destul de detaliată pentru a înțelege lucrurile care stau la baza unui algoritm de Ray Tracing în timp real. În concluzie, lucrarea de față își propune să servească drept ghid pentru cititorii care vor să experimenteze tehnici de randare avansate, însă sunt copleșiti de complexitatea teoretică a conceptelor și de complexitatea practică a uneltelor precum DirectX 12.

1.4 Soluția propusă

Soluția propusă este un motor grafic simplu de utilizat. Din perspectiva utilizatorului, acesta are un meniu din care poate configura diversi parametri ai algoritmului de Ray Tracing, precum și ai scenei. Utilizatorul poate încărca scene predefinite și poate interacționa cu acestea folosind controalele de cameră.

La nivel de bază, motorul grafic conține două implementări din clasa algoritmilor de Ray Tracing. Prima este o versiune simplă a algoritmului original descris de Whitted (1980) [10], peste care se aplică modelul clasic de iluminare Phong (1975) [7]. Această implementare a fost adaptată din codul sursă suport oferit de Microsoft.⁶ A doua implementare este scrisă de la zero și folosește algoritmul de tip Monte Carlo Path Tracing pentru a rezolva ecuația de iluminare globală propusă de Kajiya (1986) [5]. Fără a intra în prea multe detalii tehnice

⁶<https://github.com/microsoft/directx-graphics-samples/blob/master/Samples/Desktop/D3D12Raytracing/src/D3D12RaytracingProceduralGeometry/readme.md>. Accesat 19.06.2024.

(vezi secțiunea 3), această ecuație descrie cantitatea de lumină emisă dintr-un punct de pe o suprafață, de-a lungul unei direcții de vizualizare, dându-se o funcție de distribuție a luminii și un BRDF (Bidirectional Reflectance Distribution Function) pentru materialul de pe suprafață. Ecuația conține o integrală de suprafață (peste emisfera unitate), care integrează contribuțiile din toate direcțiile. Pentru eficiență, această integrală este eșantionată folosind tehnici de Importance Sampling, descrise în secțiunea 3. Totuși, numărul de eșantioane per pixel rămâne în continuare foarte limitat, din cauza bugetului de calcul (nu depășește 16 eșantioane per pixel). Pentru a reduce în continuare varianța (manifestată prin zgomot în imaginea finală), se folosește un algoritm de denoising în faza de post procesare. Mai multe detalii tehnice despre implementare sunt prezentate în secțiunea 5.

Algoritmul de Path Tracing folosește un sistem de materiale diferit de cel folosit de algoritmul Whitted. Dacă acesta din urmă este limitat de modelul simplu de iluminare Phong, Path Tracing folosește un model PBR (Physically Based Rendering) inspirat de cel introdus de Burley în 2012 [1], pentru a fi folosit în producția filmelor marca Walt Disney Animation Studios. Varianta implementată în această lucrare este augmentată cu o componentă de transmisie, descrisă într-un curs organizat de Hill et al. la conferința SIGGRAPH din 2015 [4]. Acest model este mult mai complex, având la bază un BSDF (Bidirectional Scattering Distribution Function - oferă și o componentă de transmisie a luminii prin materiale) care descrie cum un material interacționează cu lumina incidentă. Fiind totuși folosit în producția cinematografică, modelul se concentrează pe a avea o interfață cât mai intuitivă pentru artistul grafic, deviând puțin de la un model fizic strict. Bazele teoretice și implementarea acestui model PBR sunt descrise în secțiunile 3 și 5.

Pentru evaluare se folosesc 3 scene distințe. Prima este celebra Cornell Box (o variantă se poate vedea în Figura 3), care este folosită pentru a evalua corectitudinea algoritmului de iluminare globală. Se va putea face o comparație între cei doi algoritmi de Ray Tracing. Se va mai observa și capacitatea algoritmului de Path Tracing de a simula surse de lumină de tip area light, care sunt dificil de modelat în algoritmul Whitted. A doua scenă este un test al materialelor PBR, exclusivă pentru algoritmul de Path Tracing. Aceasta conține mai multe sfere cu același material de bază, care diferă printr-un singur parametru. Utilizatorul poate alege care să fie acest parametru variabil și să seteze constante pentru restul parametrilor. Scena va fi folosită pentru a evalua fidelitatea sistemului de materiale. Ultima scenă testează performanțele de rulare a algoritmilor. Aceasta conține mai multe obiecte simple, definite prin suprafețe implicate, dar și obiecte complexe definite ca mesh-uri triunghiulare cu multe poligoane.

1.5 Rezultatele obținute

//TODO

Descriere pe scurt a rezultatelor obținute, eventual de ce acestea sunt importante față de alte

soluții sau studii.

1.6 Structura lucrării

În secțiunea 2 este descris contextul actual în care se plasează eforturile din domeniu, din perspectiva industriei de gaming și de cinematografie, și se prezintă motivațiile principale pentru a continua avansurile în cercetare.

În secțiunea 3 se analizează stadiul curent al cercetărilor în domeniul algoritmilor de Path Tracing, cu focus pe optimizări pentru timp real. De asemenea, se poziționează lucrarea de față în acest peisaj și se conturează aspectul didactic al acesteia.

În secțiunea 4 se prezintă bazele teoretice ale algoritmilor utilizați (Ray Tracing Whitted și Path Tracing), iar pentru cel din urmă se detaliază modelul de materiale PBR folosit și tehniciile de denoising. De asemenea, se descrie arhitectura generală a motorului grafic, cu accent pe concepție din API-urile DirectX 12 și DXR.

Secțiunea 5 detaliază utilizarea API-urilor DirectX 12 și DXR pentru implementarea algoritmilor de Ray Tracing cu accelerare hardware. Accentul este pus pe transpunerea aspectelor teoretice în cod și pe deciziile de design luate pentru a obține o implementare eficientă și ușor de înțeles. Tot aici se notează și dificultățile întâmpinate și compromisurile făcute pentru a obține un echilibru între fidelitate și performanță.

Secțiunea de evaluare (secțiunea 6) prezintă rezultatele obținute în urma testelor efectuate pe cele trei scene descrise anterior. Se analizează performanțele sistemului, fidelitatea imaginilor generate și se fac comparații între cei doi algoritmi de Ray Tracing. De asemenea, se analizează impactul pe care îl au diferențele optimizări asupra stabilității imaginilor generate.

Ultima secțiune (secțiunea 7) conține concluziile trase din rezultatele obținute și se analizează calitativ produsul final. De asemenea, se discută posibile direcții de dezvoltare viitoare și se oferă o perspectivă asupra importanței acestei lucrări în contextul cercetării în domeniul graficii pe calculator.

Extrase de cod și imagini detaliate sunt oferite în anexă.

2 MOTIVATIE DE CERCETARE

Proiectul de față cercetează tehnici de randare a imaginilor în timp real. Acst domeniu este de mare interes pentru industria jocurilor video, care investesc resurse semnificative în dezvoltarea de motoare grafice puternice.

Un studiu de caz recent¹ analizează costurile de dezvoltare ale unui joc video AAA. Două dintre jocurile cu cel mai mare buget sunt Grand Theft Auto V și Cyberpunk 2077, care investesc peste 270, respectiv 300 milioane de dolari în dezvoltare și marketing. O parte importantă a acestor bugete a fost alocată pentru dezvoltarea motoarelor grafice proprietare.

RAGE (Rockstar Advanced Game Engine) este motorul grafic folosit de Rockstar Games pentru jocurile sale de tip open-world, precum GTA V. Acesta a trebuit să fie adaptat pentru portarea jocului pe consolele de nouă generație (PS4 și Xbox One) și pentru PC, suportând rezoluții de până la 4K pe PC.² RAGE a primit o iterare semnificativă odată cu lansarea Red Dead Redemption 2 în 2018 (un alt joc cu un buget de dezvoltare mare, de peste 100 milioane de dolari¹), care a adus noi tehnici de randare precum suport PBR, nori volumetrici și iluminare globală precalculată.^{3,4}

Cyberpunk 2077 a început dezvoltarea folosind un nou motor REDEngine 3,⁵ creat special pentru a îmbina o lume de joc vastă și detaliată cu o poveste complexă, bazată pe deciziile jucătorului. Totuși, acest motor nu era destul de flexibil pentru a suporta toate cerințele jocului⁶ (precum first person shooting și condus de mașini), așa că au început lucrul la o nouă versiune, REDEngine 4, folosind un grant de 7 milioane de dolari de la guvernul polonez.⁷ Nici acest proces nu a fost fără dificultăți (lansarea jocului a fost dezastruoasă și plină de bug-uri,⁸ iar studio-ul a decis după lansare să tranzitioneze către Unreal Engine 5⁷ pentru jocurile viitoare), dar a permis jocului să fie primul care să ofere iluminare realizată integral cu

¹<https://vnextglobal.com/category/blog/game-development-cost-an-in-depth-analysis>. Accesat 19.06.2024.

²<https://www.eurogamer.net/digitalfoundry-2015-grand-theft-auto-5-pc-face-off>. Accesat 19.06.2024.

³<https://www.eurogamer.net/digitalfoundry-2017-red-dead-redemption-2-trailer-tech-analysis>. Accesat 19.06.2024.

⁴<https://www.eurogamer.net/digitalfoundry-2018-red-dead-redemption-2-tech-analysis>. Accesat 19.06.2024.

⁵<https://www.engadget.com/2013-02-01-cd-projekt-red-introduces-redengine-3-latest-iteration-of-in-ho.html>. Accesat 19.06.2024.

⁶<https://www.superjmpmagazine.com/why-cd-projekt-reds-switch-to-unreal-engine-is-a-big-deal/>. Accesat 19.06.2024.

⁷https://www.wipo.int/edocs/mdocs/en/wipo_smes_ge_20/wipo_smes_ge_20_p3.pdf. Accesat 19.06.2024.

⁸<https://gamerant.com/cyberpunk-2077-review-bombing-negative-impact-bad-example/>. Accesat 19.06.2024.

Path Tracing.⁹ Acest lucru a fost posibil datorită colaborării cu Nvidia,¹⁰ care a oferit suport în implementarea tehnologiei RTX în REDengine 4.

Am văzut astfel ce eforturi depun companiile mari pentru a aduce fidelitate grafică în jocurile lor. Un studiu realizat de Tondello și Nacke în 2019[9] pe două esanțioane de gameri a arătat că majoritatea jucătorilor sunt interesati de aspectele estetice ale jocurilor video, precum grafica și sunetul. În alt studiu realizat de Katja et al. în 2022[8], s-a analizat concepția literaturii asupra realismului în jocuri video. Deși s-a concluzionat că acest termen nu este bine definit de multe ori, cel mai adesea în literatură acesta se referă, printre altele, la fidelitatea grafică a jocului.

Așadar, unul dintre cele mai importante aspecte ale unui joc video, în relație cu experiența jucătorului, este fidelitatea grafică.¹¹ Aceasta este influențată de calitatea modelelor 3D, a texturilor, a animațiilor, a efectelor speciale, dar și de iluminare. Iluminarea este un aspect critic al fidelității grafice, deoarece aceasta influențează cum percepem obiectele din joc. O lume frumos modelată nu poate avea un impact vizual puternic dacă nu este pusă într-o "lumină bună". Această "lumină bună" izvorăște adânc din tehnologiile folosite de motorul grafic pentru a da valoare obiectelor în scenă. Deși există multe stiluri atractive de a prezenta o scenă (e.g., cel-shading, pixel art), realismul este unul dintre cele mai populare, deoarece oferă o experiență de joc mai imersivă. Clasa de algoritmi de Ray Tracing este una dintre cele mai bune tehnici de a obține realism în jocuri, însă aceasta vine cu un cost computațional ridicat. De aceea, eforturi de cercetare în domeniul său sunt necesare pentru a găsi soluții care să ofere un compromis între fidelitate și performanță.¹² Ca referință, NVIDIA oferă public multe studii de cercetare și articole științifice publicate de echipa lor de cercetare.¹³ Marea majoritate se concentrează pe optimizarea tehnicii de prezentare grafică (Path Tracing reprezentând o parte importantă a acestora) și oferă o privire de ansamblu asupra eforturilor de cercetare în domeniu.

Ca o ultimă observație, imaginăți-vă că fidelitatea cu care se realizează producția filmelor de animație ar putea fi adusă în jocurile video. Din cealaltă perspectivă, filmele video ar putea fi randate în timp real, fără consum enorm de energie. Aceste avantaje ar reduce costurile de producție și ar accelera procesul de dezvoltare a filmelor. Eforturile de cercetare în domeniu sunt necesare pentru a face aceste viziuni realitate.

⁹<https://www.tomshardware.com/news/cyberpunk-277-rt-overdrive-available-to-all>. Accesat 19.06.2024.

¹⁰<https://www.nvidia.com/en-us/geforce/news/cyberpunk-2077-nvidia-partnership-ray-tracing/>. Accesat 19.06.2024.

¹¹<https://goombastomp.com/why-good-graphics-matter-in-video-games-enhancing-the-visual-experience/>. Accesat 19.06.2024.

¹²<https://blogs.nvidia.com/blog/rtx-real-time-ray-tracing/>. Accesat 19.06.2024.

¹³<https://research.nvidia.com/labs/rtr/publication/>. Accesat 19.06.2024.

3 METODE EXISTENTE

Literatura de specialitate din domeniul graficii pe calculator este vastă. Există multe metode de randare a imaginilor și se dă o luptă constantă între a balansa performanța cu fidelitatea. Direcția de cercetare cea mai proeminentă se axează în jurul metodelor de tip Monte Carlo, care reprezintă state-of-the-art în domeniu. În continuare, vom prezenta fundamentele teoretice ale acestei clase de algoritmi și vom analiza cele mai importante metode existente.

3.1 Fundamente și Terminologie

Metodele prezentate în continuare sunt inerent metode care se bazează extensiv pe teoria probabilității. Pentru că scopul acestei lucrări nu este de a oferi o introducere în teoria probabilității, vom prezenta doar concepțele de bază necesare înțelegerea algoritmilor de Path Tracing. Pentru o introducere mai detaliată în acest domeniu, cititorul poate consulta lucrări de specialitate precum [2] și [3].

3.1.1 Metode Monte Carlo

Metodele de tip Monte Carlo sunt metode iterative care folosesc eșantionare aleatoare pentru a aproxima valoarea unei integrale sau pentru a simula comportamentul unui sistem complex determinist, modelat stocastic. Ele sunt utilizate pentru a găsi soluții aproximative, care să conveargă către soluția corectă, pentru probleme intractabile sau care nu pot fi rezolvate analitic.

Un exemplu de pași pe care îi urmează un algoritm de tip Monte Carlo este:

1. Definirea unui domeniu de eșantionare
2. Generarea unui număr de eșantioane în domeniu, folosind o distribuție de probabilitate
3. Efectuarea calculelor (deterministe) pentru fiecare eșantion, care să aproximeze soluția
4. Agregarea rezultatelor.

3.1.2 Integrare Monte Carlo

Relevant în particular pentru această lucrare este conceptul de integrare Monte Carlo. Aceasta se referă la aproximarea unei integrale definite folosind metode Monte Carlo de eșantionare.

Să luăm un exemplu de problemă. Dându-se o funcție

$$f : \mathbb{D} \rightarrow \mathbb{R}$$

și o variabilă aleatoare continuă X cu distribuția de probabilitate $p(x)$, vrem să calculăm valoarea medie (expected value)

$$\mathbb{E}_p(f(X)) = \int_{\mathbb{D}} f(x)p(x) dx. \quad (3.1.1)$$

Valoarea medie se poate aproxima prin eșantionare, folosind formula

$$\mathbb{E}_p(f(X)) \approx \frac{1}{N} \sum_{i=1}^N f(x_i), \quad (3.1.2)$$

aproximarea devenind mai bună cu creșterea numărului de eșantioane N .

Așadar, putem aproxima integrala (3.1.1) prin

$$\int_{\mathbb{D}} f(x)p(x) dx \approx \frac{1}{N} \sum_{i=1}^N f(x_i). \quad (3.1.3)$$

Partea dreaptă a ecuației (3.1.3) este estimatorul Monte Carlo. Teorema limitei centrale afirmă că, pentru un număr suficient de mare de eșantioane, distribuția de probabilitate a estimărilor se apropie de o distribuție normală în jurul valorii medii estimate. Acest lucru înseamnă că estimatorul Monte Carlo este imparțial (unbiased) și are o varianță redusă. Dacă notăm cu f_p estimatorul, atunci au loc următoarele relații:

$$\begin{aligned} \mathbb{E}_p(f_p) &= \mathbb{E}_p(f(X)), \\ Var_p(f_p) &= \frac{1}{N} Var_p(f(X)). \end{aligned} \quad (3.1.4)$$

3.1.3 Importance Sampling

În practică, distribuția de probabilitate $p(x)$ este adesea greu de eșantionat, sau nu produce varianță cea mai mică, ceea ce duce la o convergență lentă a estimării. Eșantionarea pe baza importanței (importance sampling) este o tehnică prin care se eșantionează variabila aleatoare X dintr-o altă distribuție de probabilitate, $q(x)$, cu scopul de a reduce varianță și a estima mai bine valoarea medie a funcției f .

Să introducем această distribuție în ecuația valorii medii (3.1.1):

$$\begin{aligned} \mathbb{E}_p(f(X)) &= \int_{\mathbb{D}} f(x)p(x) dx \\ &= \int_{\mathbb{D}} f(x) \frac{p(x)}{q(x)} q(x) dx \\ &= \mathbb{E}_q \left(f(X) \frac{p(X)}{q(X)} \right). \end{aligned} \quad (3.1.5)$$

Obtinem astfel un nou estimator Monte Carlo, notat cu f_q , care își păstrează imparțialitatea (căci valoarea medie rămâne aceeași) și care folosește distribuția $q(x)$:

$$\mathbb{E}_q(f_q) = \mathbb{E}_p(f(X)) \approx \frac{1}{N} \sum_{i=1}^N f(x_i) \frac{p(x_i)}{q(x_i)}. \quad (3.1.6)$$

Varianta acestui estimator, este dată de

$$Var_q(f_q) = \frac{1}{N} Var_q \left(f(X) \frac{p(X)}{q(X)} \right). \quad (3.1.7)$$

Scopul este de a alege distribuția $q(x)$ astfel încât varianta estimatorului să fie mai mică decât varianța inițială, i.e.

$$\frac{1}{N} Var_q \left(f(X) \frac{p(X)}{q(X)} \right) < \frac{1}{N} Var_p(f(X)). \quad (3.1.8)$$

Pentru a minimiza varianța noului estimator, în mod ideal, vrem ca funcția $f(x) \frac{p(x)}{q(x)}$ să fie constantă, pentru orice x eșantionat, ceea ce ar duce la o varianță nulă. Acest lucru se întamplă dacă

$$q(x) = c \cdot f(x)p(x), \quad (3.1.9)$$

unde c este o constantă de normalizare. Pentru a păstra imparțialitatea, c trebuie să fie egal cu $\frac{1}{\mathbb{E}_p(f(X))}$, însă această valoare este necunoscută (altfel nu am mai avea nevoie de estimator!). Așadar, nu este fezabilă alegerea optimă pentru $q(x)$. Totuși, ecuația (3.1.9) ne arată că distribuția optimă este proporțională cu funcția $f(x)p(x)$. Acest produs măsoară fix importanța fiecărui eșantion în estimarea mediei, de unde și numele tehnicii.

În Figura 5 se poate observa efectul eșantionării bazate pe importanță. În acest caz, funcția f are o regiune mică de unde vine majoritatea contribuției către valoarea medie. Dacă distribuția de probabilitate p nu eșantionează bine această regiune (în acest caz, este o distribuție uniformă), varianța estimării va fi mare. Distribuția q este aleasă astfel încât să eșantioneze mai mult din zonele importante și se poate vedea pe subgraficul din dreapta cum raportul $\frac{p(x)}{q(x)}$ încearcă să echilibreze contribuția fiecărui eșantion, reducând varianța estimării.

3.1.4 Eșantionare Stratificată

O altă metodă de a reduce varianța estimării este eșantionarea stratificată (Stratified Sampling). În această metodă se partionează domeniul de eșantionare \mathbb{D} în n subdomenii, numite straturi, iar evaluarea integralei se face pe fiecare strat în parte:

$$\int_{\mathbb{D}} f(x)p(x) dx = \sum_{i=1}^n \int_{\mathbb{D}_i} f(x)p(x) dx. \quad (3.1.10)$$

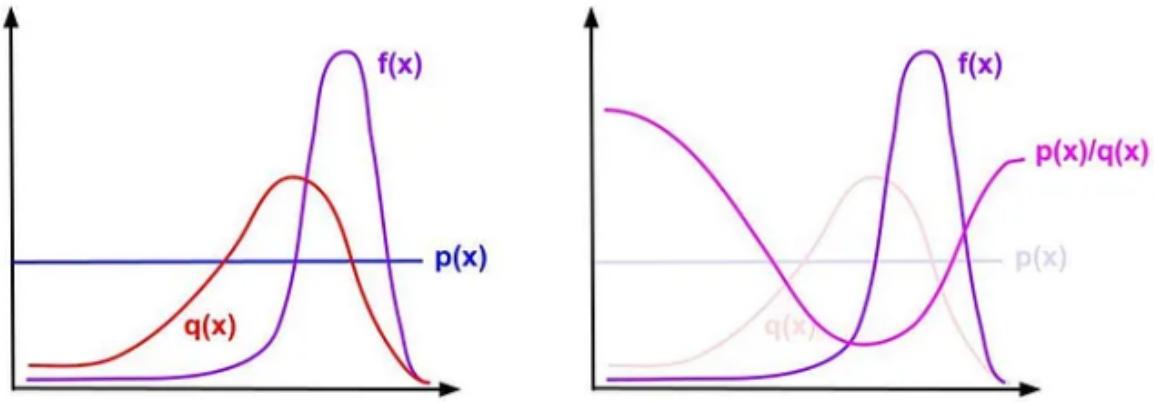


Figura 5: Efectul eșantionării bazate pe importanță. Credite: <https://medium.com/>

În acest caz, varianța estimării devine suma varianțelor pe fiecare strat:

$$Var(f_s) = \sum_{i=1}^n Var(f_i). \quad (3.1.11)$$

Se poate demonstra că, în cazul în care toate straturile au aceeași măsură, i.e.

$$\int_{\mathbb{D}_i} p(x) dx = \frac{1}{n} \int_{\mathbb{D}} p(x) dx, \quad \forall i \in \{1, 2, \dots, n\}, \quad (3.1.12)$$

atunci varianța estimării nu va fi mai mare decât fără eșantionare stratificată. Pentru mai multe informații se poate consulta cartea de specialitate a lui Kleijnen et al. [6].

Un exemplu de eșantionare stratificată este ilustrat sub tehnica de jittering în eșantionarea pixelilor, detaliată în secțiunea 5.

3.2 Ecuatia transportului luminii

4 SOLUȚIA PROPUȘĂ

//TODO

Capitolul conține o privire de ansamblu a soluției ce rezolvă problema, prin prezentarea structurii / arhitecturii acesteia. În funcție de tipul lucrării acest capitol poate conține diagrame (clase, distribuție, workflow, entitate-relație), demonstrații de corectitudine pentru algoritmii propuși de autor, abordări teoretice (modelare matematică), structura hardware, arhitectura aplicației.

Criterii pentru calificativul *Bine*:

- Descriere + diagrame de baze de date, workflow, clase, algoritmi + descrierea unui proces prin care s-a realizat arhitectura/structura soluției.

5 DETALII DE IMPLEMENTARE

//TODO

În plus fata de capitolul precedent acesta conține elemente specifice ale rezolvării problemei care au presupus dificultăți deosebite din punct de vedere tehnic. Pot fi incluse configurații, secvențe de cod, pseudo-cod, implementări ale unor algoritmi, analize ale unor date, scripturi de testare. De asemenea, poate fi detaliat modul în care au fost utilizate tehnologiile introduse în capitolul 3.

Criterii pentru calificativul *Bine*:

- Descrierea detaliată a algoritmilor/structurilor utilizati; Prezentarea etapizată a dezvoltării, inclusiv cu dificultăți de implementare întâmpinate, soluții descoperite; (dacă este cazul) demonstrarea corectitudinii algoritmilor utilizati.

5.1 Indicații formatare tabele

Se recomandă utilizarea tabelelor de forma celui de mai jos. Font size : 9. Orice tabel prezent în teză va fi referit în text; exemplu: a se vedea Tabel 1.

Tabela 1: Sumarizare criterii

Calificativ	Criteriu	Observații
Nesatisfacator	Sunt prezentate pe scurt scheme și pseudo-cod	
Satisfacator	Descriere sumară a implementării, prezentarea unor secvențe nerelevante de cod, scheme, etc.	
Bine	Descrierea detaliată a algoritmilor/structurilor utilizati; Prezentarea etapizată a dezvoltării, inclusiv cu dificultăți de implementare întâmpinate, soluții descoperite; (dacă este cazul) demonstrarea corectitudinii algoritmilor utilizati.	Pot fi incluse configurații, secvențe de cod, pseudo-cod, implementări ale unor algoritmi, analize ale unor date, scripturi de testare.

6 EVALUARE

//TODO

Acest capitol trebuie să răspundă, în principiu, la 2 întrebări și să se încheie cu o discuție a rezultatelor obținute. Cele două întrebări la care trebuie să se răspundă sunt:

1. **Merge corect?** (Conform specificațiilor extrase în capitolul 2); Evaluarea dacă merge corect se face pe baza cerințelor identificate în capitolele anterioare.
2. Cât de *Bine* merge / cum se compară cu soluțiile existente? (pe baza unor metrii clare). Evaluarea cât de *Bine* merge trebuie să fie bazată pe procente, timpi, cantitate, numere, **comparativ cu soluțiile prezentate în capitolul 3**. Poate fi vorba de performanță, overhead, resurse consumate, scalabilitate etc.

În realizarea discuției, se vor utiliza tabele cu procente, rezultate numerice și grafice. În mod obișnuit, aici se fac comparații și teste comparative cu alte proiecte similare (dacă există) și se extrag puncte tari și puncte slabe. Se ține cont de avantajele menționate și se demonstrează viabilitatea abordării / aplicației, de dorit prin comparație cu alte abordări (dacă acest lucru este posibil). Cuvântul cheie la evaluare este "metrică": trebuie să aveți noțiuni măsurabile și cuantificabile. În cadrul procesului de evaluare, explicați datele, tabelele și graficele pe care le prezentați și insistați pe relevanța lor, în următorul stil: "este de preferat ... deoarece ..."; explicați cititorului nu doar datele ci și semnificația lor și cum sunt acestea interpretate. Din această interpretare trebuie să rezulte poziționarea proiectului vostru printre alternativele existente, precum și cum poate fi acesta îmbunătățit în continuare.

Criterii pentru calificativul *Bine*:

- [Dezvoltare de produs] Teste unitare și de integrare, instrumente de punere în funcțiune (deployment) utilizate și care arată lucru constant de-a lungul semestrului, lucrare validată cu clienții / utilizatorii, produs în producție.
- [Cercetare] Componentele și soluția în ansamblu au fost evaluate din punct de vedere al performanței, folosind seturi de date standard și cu o interpretare corectă a rezultatelor.
- [Ambele] Discuție cu prezentarea calitativă și cantitativă a rezultatelor, precum și a relevanței acestor rezultate printr-o comparație complexă cu alte sisteme similare.

7 CONCLUZII

//TODO

În acest capitol este sumarizat întreg proiectul, de la obiective, la implementare, și la relevanța rezultatelor obținute. În finalul capitolului poate exista o subsecțiune de "Dezvoltări ulterioare".

Criterii pentru calificativul *Bine*:

- Concluziile sunt corelate cu conținutul lucrării, și se oferă o imagine precisa asupra relevantei și calității rezultatelor obținute în cadrul proiectului.

BIBLIOGRAFIE

- [1] Brent Burley. Physically-based shading at disney. 2012.
- [2] John H. Halton. A retrospective and prospective survey of the monte carlo method. *SIAM Rev.*, 12(1):1–63, jan 1970.
- [3] J. M. Hammersley and David Handscomb. Monte carlo methods. 1964.
- [4] Stephen Hill, Stephen McAuley, Brent Burley, Danny Chan, Luca Fascione, Michał Iwanicki, Naty Hoffman, Wenzel Jakob, David Neubelt, Angelo Pesce, and Matt Pettineo. Physically based shading in theory and practice. In *ACM SIGGRAPH 2015 Courses*, SIGGRAPH '15, New York, NY, USA, 2015. Association for Computing Machinery.
- [5] James T. Kajiya. The rendering equation. *SIGGRAPH Comput. Graph.*, 20(4):143–150, aug 1986.
- [6] Jack P. C. Kleijnen, Ad A. N. Ridder, and Reuven Y. Rubinstein. *Variance Reduction Techniques in Monte Carlo Methods*, pages 1598–1610. Springer US, Boston, MA, 2013.
- [7] Bui Tuong Phong. Illumination for computer generated pictures. *Commun. ACM*, 18(6):311–317, jun 1975.
- [8] Katja Rogers, Sukran Karaosmanoglu, Maximilian Altmeyer, Ally Suarez, and Lennart E. Nacke. Much realistic, such wow! a systematic literature review of realism in digital games. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, CHI '22, New York, NY, USA, 2022. Association for Computing Machinery.
- [9] Gustavo F. Tondello and Lennart E. Nacke. Player characteristics and video game preferences. In *Proceedings of the Annual Symposium on Computer-Human Interaction in Play*, CHI PLAY '19, page 365–378, New York, NY, USA, 2019. Association for Computing Machinery.
- [10] Turner Whitted. An improved illumination model for shaded display. *Commun. ACM*, 23(6):343–349, jun 1980.

ANEXE

Anexele sunt optionale. Ce poate intra în anexe:

- Exemplu de fișier de configurare sau compilare;
- Un tabel mai mare de o jumătate pagină;
- O figura mai mare mai mare de jumătate pagină;
- O secvență de cod sursă mai mare de jumătate pagină;
- Un set de capturi de ecran (“screenshot”-uri);
- Un exemplu de rulare a unor comenzi plus rezultatul (“output”-ul) acestora;
- În anexe intră lucruri care ocupă mai mult de o pagină ce ar îintrerupe firul natural de parcursere al textului.

A EXTRASE DE COD

...