https://doi.org/10.31649/1997-9266-2025-179-2-132-137

УДК 004.92

Р. М. Пікульський[1]

АЛГОРИТМ ТРАСУВАННЯ ПРОМЕНІВ НА ОСНОВІ МЕТОДУ SCREEN SPACE RAY MARCHING

[1]Київський національний університет імені Тараса Шевченка

Трасування променів є дуже важливою операцією в сучасній комп'ютерній графіці, яка дає змогу здійснювати фотореалістичний рендеринг віртуальних середовищ з високим рівнем точності моделювання взаємодії світла з геометричними об'єктами. Відстежуючи шлях окремих променів світла, бібліотеки рендерингу здатні обчислювати дзеркальні відображення, тіні, заломлення світла та глобальне освітлення, що забезпечує високу якість зображення. Ця технологія є невід'ємною частиною програмного забезпечення для комп'ютерного моделювання, створення анімаційних фільмів та кінематографічних ефектів, а також комп'ютерних ігор та інших інтерактивних графічних програм, де попит на реалістичні візуальні ефекти продовжує зростати. Розвиток апаратних можливостей графічних процесорів, зокрема поява підтримки апаратного прискорення трасування променів, значно підвищує продуктивність операції трасування. Проте трасування променів залишається слабкою ланкою у ефективності більшості алгоритмів, які використовуються в сучасних бібліотеках рендерингу. У зв'язку з цим оптимізація трасування променів є актуальним напрямком досліджень.

Метою цієї роботи є розробка реалізації методу маршування променів у просторі екрану, здатної визначити коректний результат трасування променя у разі коли буфер глибини містить необхідну для цього геометричну інформацію про середовище. Представлена реалізація цього методу є одночасно ефективною та забезпечує хорошу точність результатів. Для підтвердження результатів і вимірювання продуктивності отриманої реалізації проводиться експериментальне тестування. Дані, отримані на основі цих спостережень, доводять актуальність використання нашої реалізації в гібридній системі трасування променів, яка використовує метод маршування променів у просторі екрану, щоб уникнути виконання повільного апаратного трасування променів для частини запитів трасування.

Ключові слова: комп'ютерна графіка, інтерактивний рендеринг, трасування променів, маршування променів у просторі екрану.

Вступ

Комп'ютерна графіка — це важлива галузь комп'ютерних наук, яка вивчає методи побудови зображень з використанням комп'ютерної техніки. Однією з фундаментальних цілей комп'ютерної графіки є рендеринг — створення реалістичного зображення певного віртуального середовища на основі його аналітичного опису та конфігурації спостерігача. З огляду на це, важливою задачею є визначення видимого об'єкта для цієї позиції та напрямку спостереження, оскільки її розв'язання необхідне для фізичного моделювання руху світла у середовищі [1]. Задача виконання рендерингу стає дедалі складнішою у разі потреби роботи системи в режимі реального часу з використанням типового апаратного забезпечення комп'ютерних пристроїв загального користування, що накладає жорсткі обмеження на час виконання алгоритмів рендерингу та споживання відеопам'яті.

Найпоширенішими способами розв'язання задачі визначення видимості є растеризація [2] та трасування променів [3]. Метод растеризації набув широкого застосування, оскільки він передбачає прогнозований та впорядкований доступ до пам'яті, що є сприятливим для виконання на сучасному апаратному забезпеченні. Його поширення також зумовило впровадження його апаратної підтримки у всіх графічних адаптерах, які підтримують прискорення тривимірної графіки. Через це растеризація стає дуже ефективною для практичного використання. Проте по суті растеризація призначена для визначення видимості для променів, що відповідають пікселям зображення певного кадру, тому метод не може ефективно застосовуватися для довільної множини променів — до прикладу, променів дзеркальних відображень або глобального освітлення.

Отже, трасування променів є важливою процедурою, яка застосовується як універсальне рішення

для визначення видимості. Існує багато методів трасування променів, які використовують різні структури даних для представлення інформації про геометрію середовища. Основним методом загального призначення вважається трасування ієрархії граничних об'ємів (bounding volume hierarchy, BVH). У сучасному графічному апаратному забезпеченні впроваджено апаратне прискорення трасування BVH. Проте цей метод передбачає неефективні патерни доступу до пам'яті, що значно обмежує його ефективність. На практиці, сучасні графічні адаптери не мають достатньої обчислювальної потужності для забезпечення достатнього обсягу трасування променів, що необхідно для фотореалістичного рендерингу в реальному часі. Тому оптимізація трасування променів є популярним напрямком досліджень у комп'ютерній графіці.

Маршування променів у екранному просторі (screen space ray marching, SSRM) [4], [5] стало популярною альтернативою трасуванню променів для рендерингу в реальному часі. Цей метод знаходить проекцію променя на зображення основного виду та відслідковує його перетин з видимими поверхнями, використовуючи буфер глибини [6] основного виду як джерело геометричної інформації. Оскільки буфер глибини не містить повної інформації про геометрію віртуального середовища, метод SSRM не здатен правильно розв'язати задачу визначення видимості для довільного променя. Проте він часто може досягати високої частки коректно оброблених запитів для деяких типових технік рендерингу (наприклад, рендеринг дзеркальних відображень, невіддалених тіней тощо). Також, він має передбачуваний патерн доступу до пам'яті та може бути ефективно реалізований. Проте сучасні вимоги до якості рендерингу виключають можливість некоректних тестів видимості, через що метод SSRM неприйнятний для використання як самостійний алгоритм.

Метою роботи є розробка ефективної реалізації методу SSRM, яка уникає хибно-позитивних результатів трасування, тобто вважає результат дійсним лише за умови можливості гарантувати його коректність. З огляду на таку властивість, цю реалізацію можна використовувати у гібридній системі трасування променів [7], у якій усі запити трасування спочатку обробляються алгоритмом SSRM, а після цього ті запити, для яких не вдалося отримати дійсні результати, обробляються основним алгоритмом трасування, що гарантовано визначає дійсний результат. Незважаючи на додатковий крок у вигляді SSRM алгоритму, така схема може бути ефективнішою, оскільки кількість променів, які потребують виконання основного алгоритму, може бути значно меншою порівняно із загальною кількістю запитів. З метою аналізу ефективності практичного застосування гібридного трасування променів, проводяться експериментальні вимірювання швидкості розробленої реалізації SSRM та апаратного трасування променів для запитів трасування дзеркальних відображень. На основі отриманих даних можна оцінити актуальність застосування гібридного трасування у сучасних застосунках рендерингу в реальному часі.

Реалізація маршування променів у екранному просторі

Базові принципи методу маршування променів у екранному просторі запозичено з роботи [5]. Ключовою особливістю нашої реалізації є дотримання суворих критеріїв дійсності перетину променів з геометрією навколишнього середовища. Наприклад, на відміну від інших поширених варіантів SSRM, маршування променів здійснюється з обробкою усієї геометричної інформації вздовж променя без пропусків пікселів. Також, алгоритм не використовує жодних евристичних методів визначення взаємодії променів з геометричним представленням пікселя, натомість, якщо промінь проходить крізь ділянку, про яку немає геометричної інформації (або за видимою поверхнею, яка записана у буфері глибини, або взагалі позаду спостерігача), алгоритм закінчує роботу, вказуючи на неможливість визначення коректного результату трасування. З таким принципом роботи розроблена реалізація SSRM менш ефективна з погляду швидкодії та менш продуктивною з погляду кількості результативних маршувань, але вона мінімізує кількість хибних результатів, що дозволяє використовувати її у гібридній системі трасування променів з мінімальним погіршенням якості результатів трасування.

Процедура маршування променів потребує два буфери даних про геометрію середовища: ієрархічний буфер глибини та буфер похідних глибини. Ієрархічний буфер глибини (піраміда глибини) — це екранний буфер (двовимірна текстура, розміри якої збігаються з розміром екрану) з MIP-ланцюгом — послідовністю дочірніх буферів, кожний з яких удвічі менший за попередній вздовж обидвох осей екрану. Перший і найбільший буфер цієї послідовності є копією буфера глибини основного виду. Кожен з інших буферів є по суті грубою апроксимацією попереднього. Кожний піксель такого буфера ставиться у відповідність квадрату пікселів розміром 2×2 попереднього

буфера з відповідними координатами та містить значення глибини, що є найближчим до спостерігача (залежно від реалізації представлення глибини, це може бути арифметичний мінімум або максимум). Цей буфер використовується як основне джерело геометричної інформації для маршування. MIP-буфери дозволяють обробляти декілька пікселів одразу, зчитуючи лише одне значення глибини, що відображає спільну порожню ділянку простору — це зменшує кількість операцій необхідних для повного маршування. Кількість MIP-буферів залежить від ширини W та висоти H екрану та дорівнює $[\log_2(\min(W,H))]$. За такої кількості одна з розмірностей останнього MIP-буфера дорівнює одиниці. Таким чином, для роздільної здатності екрану 1920×1080 кількість MIP-буферів становить 11 елементів, для екрану 3840×2160 — 12. Побудова піраміди глибини відбувається аналогічно принципу роботи застосунку побудови MIP-ланцюгів текстур FidelityFX Single Pass Downsampler [8]. Ця бібліотека оптимізована для сучасного апаратного забезпечення і з використанням аналогічних підходів побудова піраміди глибини є незначною операцією порівняно з маршуванням променів.

Буфер похідних глибини — екранний буфер, кожен піксель якого містить двокомпонентний вектор, значення компонент якого дорівнюють зміні глибини видимого елемента поверхні вздовж горизонтальної та вертикальної осей екрану відповідно. Ці дані дозволяють інтерпретувати дані з буфера глибини як поверхню, що не є завжди перпендикулярною до напрямку зору спостерігача, а натомість спостерігається під певним кутом (звідки й зміна глибини). Таке представлення є набагато ближчим до дійсності та дозволяє значно підвищити точність тестів перетину променів з елементом видимої поверхні пікселя. Побудова буфера похідних глибини відбувається під час растеризації основного виду, де такі похідні можна отримати шляхом використання функцій ddx та ddy до системної змінної, що містить значення глибини, отримане з растрування геометрії; з огляду на це, побудова буфера похідних глибини також є практично безкоштовною операцією.

Дані про запити трасування, які має обробити SSRM, подаються у вигляді двох буферів екрану, які містять вершини та напрямки променів відповідно у просторі екранної проекції. Результати запитів трасування записуються у екранний буфер двокомпонентних векторів. У разі успішного виконання запиту результатом є координата пікселя, у якому відбувся перетин променя з поверхнею; інакше, використовується некоректне значення координати (–1, –1) як маркер неуспішного трасування. Така конфігурація вхідних та вихідних даних передбачає виконання одного запиту трасування для кожного пікселя екрану та підходить для експериментальних тестів, які здійснені у цій роботі. У разі коли конкретні алгоритми рендерингу передбачають нерівномірну кількість запитів трасування для пікселів екрану, конфігурацію вхідних та вихідних даних може бути модифіковано до використання лінійних масивів даних замість екранних буферів.

Сама процедура маршування променів відбувається шляхом виконання повноекранного обчислювального шейдера. Вона має декілька стадій: стадію ініціалізації, цикл маршування та стадію корекції результату.

Ініціалізація — це початкова стадія, у якій шейдер зчитує дані про промінь та обчислює декілька допоміжних констант необхідних для маршування. Наприклад, відбувається попереднє обчислення інвертованого значення напрямку вхідного променя, крок координат пікселя при маршуванні — ці та інші дані використовуються для обробки пікселів під час маршування променя. Також обчислюється максимальна дистанція ходу променя з огляду на перетин з об’ємом головного виду – якщо маршування виходить за його межі, де дані про геометрію відсутні, процедура маршування вважається неуспішною.

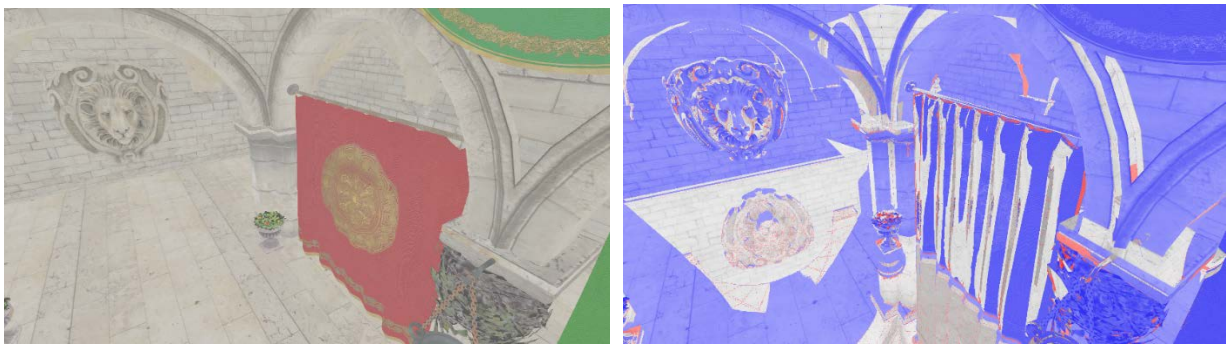
Цикл маршування — основна стадія алгоритму, де відбувається послідовне сканування геометричної інформації вздовж вхідного променя та пошук перетину. У тілі циклу алгоритм зчитує глибину видимої поверхні з піраміди глибини, визначає глибину променя у ділянці, що відповідає зчитаному значенню та порівнює отримані значення глибини між собою. Якщо промінь знаходиться далі від спостерігача, ніж видима поверхня — це зараховується як перетин, інакше вважається, що перетин відсутній. Ця стадія використовує класичну інтерпретацію глибини як площини, перпендикулярної до напрямку зору — це зумовлює ефективність тестів перетину. Кожна ітерація циклу обробляє чотири послідовні вибірки геометричної інформації з поточного MIP-буфера піраміди глибини вздовж променя, що дозволяє краще балансувати навантаження на блоки арифметичних обчислень та блоків доступу до пам’яті графічного процесора, підвищуючи швидкодію алгоритму.

Використовуючи піраміду видимості та читаючи дані з різних MIP-буферів, алгоритм може обробляти декілька пікселів за раз, що пришвидшує процедуру маршування. З самого початку мар-

шування використовується MIP-буфер з індексом 0 (тобто, найдетальніший буфер). Якщо у поточній ітерації циклу перетину не відбулося, MIP-індекс збільшується на одиницю, і тоді у наступній ітерації алгоритм сканує наступні ділянки перетину пікселів з променем, використовуючи менш детальну інформацію. Це допомагає швидко минати великі площі, де перетинів немає. Якщо ж відбувається перетин, то MIP-індекс зменшується на одиницю, а наступна ітерація повторює обробку ділянки у якій відбувся перетин. Цикл маршування завершується, якщо відбувається перетин з ділянкою, що складається з одного пікселя (тобто, якщо індекс MIP-буфера відповідної вибірки дорівнював нулю), або ж у разі коли маршування виходить за межі головного виду.

Стадія корекції результату — фінальна стадія алгоритму, де на основі результатів циклу маршування визначається результат роботи алгоритму. Якщо цикл маршування вийшов за межі об'єму головного виду — запит трасування оголошується неуспішним, оскільки алгоритм не має геометричної інформації необхідної для трасування променя по всій його довжині. Якщо ж цикл маршування виявив перетин променя з глибиною деякого пікселя, алгоритм повторює тест перетину променя з цим пікселем, інтерпретуючи піксель як площину, яка не є перпендикулярною до напрямку зору з використанням значень похідних глибини по осях екрану з відповідного буфера. Якщо повторний тест виявляється успішним, алгоритм вважає знайдений піксель коректним результатом та записує його координати у вихідний буфер; інакше, ідентичний тест застосовується до попереднього пікселя, який покривається променем. Якщо тест з попереднім пікселем є успішним, цей піксель вважається коректним результатом; інакше, трасування вважається неуспішним. Після цього робота алгоритму завершується. Такий підхід до корекції результатів дозволяє отримати доволі точні результати трасування та значною мірою нівелює похибки класичної інтерпретації глибини пікселів, що використовується у циклі маршування, при цьому він є досить ефективним з погляду швидкодії.

Приклад візуалізації результату роботи алгоритму показано на рисунку. Як видно, алгоритм здатний обробити значну частку запитів трасування; також, частка неточних результатів є відносно невеликою.



а *б*

Результат роботи алгоритму на сцені Sponza: *а* — рендеринг головного виду; *б* — дзеркальні відображення, обчислені методом SSRM (зображені у відтінках сірого). Ділянки синього кольору вказують на неуспішність трасування у відповідних пікселях, червоного — на результати, що виявилися неточними порівняно з результатами апаратного трасування

Тестування

З метою визначення швидкодії розробленої реалізації методу маршування променів у екранному просторі та оцінки актуальності побудови гібридної системи трасування променів на її основі здійснено експериментальне тестування. Для цього розроблено демонстраційний додаток, який здійснює рендеринг певного середовища (сцени). При цьому, для кожного пікселя підсумкового зображення генерується промінь, що є дзеркальним відображенням напрямку зору спостерігача від відповідної видимої поверхні. Такі запити трасування імітують загальновживаний метод обчислення дзеркальних відображень, що є важливим компонентом фотореалістичного рендерингу та одним з основних застосувань трасування променів у сучасних бібліотеках рендерингу у реальному часі. Додаток здійснює трасування згенерованих променів методом SSRM та з використанням апаратного прискорення трасування променів, що дає змогу порівняти результати та ефективність цих методів.

Тестування здійснювалося з використанням графічного процесора Nvidia RTX 2080 Super. Використано три різні сцени, які містять різну кількість геометрії: Sphere (приблизно 1000 трикутни-

ків), Sponza (приблизно 2^{18} трикутників) та Bistro (приблизно 2^{20} трикутників). Використання різних сцен необхідне для повної оцінки ефективності апаратного трасування променів, швидкість якого залежить від складності сцени, натомість, швидкість методу SSRM є майже незмінною на різних сценах через використання буфера глибини як джерела геометричної інформації. Результати вимірювань середнього часу виконання реалізації SSRM та апаратного трасування променів подано у табл. 1 та 2 відповідно. Як видно з таблиць, час виконання апаратного трасування значно переважає час трасування за використання методу SSRM.

Таблиця 1

Середній час виконання реалізації методу SSRM

Роздільна здатність екрану	1280×720	1920×1080	2560×1440	3840×2160
Час виконання, мс	0,212	0,503	0,937	2,205

Таблиця 2

Час виконання трасування променів з використанням апаратного прискорення

Роздільна здатність екрану	1280×720	1920×1080	2560×1440	3840×2160
Sphere	0,91	1,76	2,78	5,49
Sponza	2,76	5,59	10,24	23,79
Bistro	7,65	9,63	25,11	52,81

Для того щоб оцінити ефективність гібридної системи трасування променів з використанням розробленої реалізації SSRM визначено частку успішно виконаних запитів трасування цим методом. Середнє її значення становило $\alpha = 0{,}31$; конкретне значення дуже відрізняється залежно від сцени та положення спостерігача. Зважаючи на отримане середнє значення, час трасування з використанням гібридної системи можна оцінити як $T \approx T^{SSRM} + (1 - \alpha)T^{HW}$, де T^{SSRM} та T^{HW} — час трасування з використанням методу SSRM та апаратного прискорення відповідно. У табл. 3 відображено очікуване підвищення ефективності трасування внаслідок використання гібридної системи, визначене як відношення часу трасування з використанням апаратного прискорення до оцінки часу трасування з використанням гібридної системи. Як видно з табл. 3, розробка та використання гібридної системи трасування променів є актуальним у разі необхідності рендерингу складних середовищ з великою кількістю геометрії.

Таблиця 3

Оцінка підвищення ефективності трасування променів внаслідок використання гібридної системи

Роздільна здатність екрану	1280×720	1920×1080	2560×1440	3840×2160
Sphere	1,083	1,025	0,96	0,901
Sponza	1,304	1,282	1,273	1,271
Bistro	1,393	1,347	1,372	1,363

Висновок

У роботі розроблено та представлено реалізацію методу маршування променів у екранному просторі (SSRM), яка призначена для точного трасування променів за наявності достатньої геометричної інформації. Завдяки використанню оптимізованих підходів, реалізація є доволі ефективною. Експериментально показано, що вона є швидшою за трасування з використанням апаратного прискорення, навіть для рендерингу нескладних середовищ. Евристичні оцінки, здійснені на основі проведеного тестування, свідчать, що гібридна система трасування променів з використанням методу SSRM та апаратного прискорення трасування може бути ефективнішою за просте апаратне трасування на типових сценах. Подальші дослідження мають бути направлені на вдосконалення точності трасування методом SSRM та збільшення частки успішно виконаних запитів трасування.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

[1] M. Pharr, W. Jakob, and G. Humphreys, *Physically Based Rendering: From Theory to Implementation*, 3rd ed., San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2016.

[2] J. Pineda, “A parallel algorithm for polygon rasterization’,” *SIGGRAPH Comput. Graph.*, vol. 22, no. 4, pp. 17-20, 1988.

[3] T. Whitted, “An improved illumination model for shaded display’,” *Commun. ACM*, vol. 23, no. 6, pp. 343-349, 1980.

[4] M. McGuire, and M. Mara, “Efficient GPU Screen-Space Ray Tracing’,” *Journal of Computer Graphics Techniques*

(JCGT), vol. 3, no. 4, pp. 73-85, 2014.

[5] Y. Uludag, “Hi-Z Screen-Space Cone-Traced Reflections’,” in *GPU Pro 5: Advanced Rendering Techniques (1st ed.)*, A K Peters/CRC Press, 2014, ch. 2.4, pp. 149-192.

[6] T. Theoharis, G. Papaioannou, and A. Karabassi, “The Magic of the Z-Buffer,” *a Survey*, 01. 2001.

[7] T. Willberger, C. Musterle, and S. Bergmann, “Deferred Hybrid Path Tracing: High-Quality and Real-Time Rendering with DXR and Other APIs’,” in E. Haines, T. Akenine-Möller. *Ray Tracing Gems*. Apress, Berkeley, CA, USA, 2019, pp. 475-492.

[8] AMD FidelityFX Single Pass Downsampler (SPD). AMD GPUOpen. [Online]. Available: https://gpuopen.com/fidelityfx-spd/ .

Рекомендована кафедрою програмного забезпечення ВНТУ

Стаття надійшла до редакції 4.04.2025

Пікульський Ростислав Михайлович — аспірант кафедри математичної інформатики, e-mail: rostyslav.pikulsky@gmail.com .

Київський національний університет імені Тараса Шевченка, Київ

R. M. Pikulsky[1]

Ray Tracing Algorithm Based on Screen Space Ray Marching

[1]Taras Shevchenko National University of Kyiv

Ray tracing is a very important operation in modern computer graphics which enables photorealistic rendering with advanced level of accuracy in simulating light transport and interaction with geometric objects. By tracing the path of individual light rays and determining how they bounce through the environment, rendering engines produce highly detailed and precise reflections, shadows, refractions and global illumination, which significantly enhances visual fidelity. This technology is an integral part of computer simulation and modeling software packages, animated films and cinematic effects production software and also computer games and other interactive graphics applications where the demand for realistic visuals continues to grow. Recent advancements in graphics hardware capabilities including hardware ray tracing support introduction increased ray tracing performance significantly. However, ray tracing remains a significant bottleneck in a majority of algorithms used in modern rendering engines. Due to that, ray tracing optimization is an important research direction nowadays.

This work is aimed at development of the screen space ray marching implementation capable of correct determination of the valid ray tracing query result when environment geometric information stored in the depth buffer is sufficient to guarantee it. The implementation of this method is presented that is both efficient and provides good precision of the results. Experimental testing is performed to validate the results and measure the performance of the implementation. Based on these observations, we prove the benefit of using our implementation in the hybrid ray tracing system, which runs screen space ray marching method to avoid executing costly hardware ray tracing for all the rays to be traced.

Keywords: computer graphics, interactive rendering, ray tracing, screen space ray marching.

Pikulsky Rostyslav M. — Post-Graduate Student of the Chair of Mathematical Informatics, e-mail: rostyslav.pikulsky@gmail.com