УДК 004

Бурдейний О.В.
Черняк О.І.

Комп'ютерна система візуалізації троивимірних сцен з поєднанням полігонального рендерингу та маршингу променів у середовищі Unity

Вінницький національний технічний університет

Анотація

Спроєктовано комп'ютерну систему візуалізації тривимірних сцен, котра поєднує класичний полігональний рендеринг із методом маршингу променів Raymarching в межах одного кадру. Запропоновано спосіб взаємної візуалізації в межах якого процедурні об'єкти, що не мають полігональної сітки, коректно перетинаються з полігональною геометрією сцени, взаємно відкридаються тіні та освітлюються єдиним джерелом світла. Результатом дослідження є шейдер для унверсального конвеєра рендерингу Unity URP та підхід до його ін'єкції в кадр, котрі можуть використовуватись в ігрових, наукових та навчальних проєктах.

Ключові слова: полігон, піксель, модель, шейдер, HLSL, SDF, Raymarching, Unity, URP, буфер глибини, буфер кольору, графічний конвеєр, кадр, полігональна растеризація.

Abstract

A computer system for visualizing three-dimensional scenes has been designed that combines classical polygonal rendering with the raymarching method within a single frame. A method of mutual visualization has been proposed, within which procedural objects lacking a polygonal mesh correctly intersect with the scene's polygonal geometry, cast shadows on one another, and are illuminated by a single light source. The result of this research is a shader for Unity URP's universal rendering pipeline and an approach for injecting it into a frame, which can be used in gaming, scientific, and educational projects.

Keywords: polygon, pixel, model, shader, HLSL, SDF, raymarching, Unity, URP, depth buffer, color buffer, graphics pipeline, frame, polygonal rasterization.

Вступ

На даний час основним способом виведення тривимірної графіки в реальному часі залишається полігональна візуалізація з використанням растеризації [1], однак такий підхід має суттєві обмеження під час відображення об'єктів, котрі не мають чітковизначеної форми й погано описуються полігональною сіткою. Прикладом таких об'єктів зазвичай є більшість природніх структур, як от об'ємні динамічні хмари, водні поверхні, природні явища чи високодеталізовані протяжні рельєфи. Вимога збільшення деталізації подібних об'єктів призводить до різкого зростання кількості вершин та наванатження на пам'ять обчислювального пристрою [2]. Відповідно і спроба деформації таких форм в реальному часі, окрім візуальних помилок, також призводить до значного навантаження. На противагу цьому, метод маршингу променів Raymarching [3] дозволяє обчислювати подібні о'бємні динамічні об'єкти безпосередньо в пікселях зображення за математичним описом, однак в ізольованому вигляді, як окремий фон або ж повноекранне зображення. Отже, такий метод не взаємодіє з рештою сцени, оскільки відсутні коректні перетини з полігональними об'єктами, взаємне затінення та спільне освітлення. Метою даного дослідження є отримання високодеталізованих сцен з мінімальними витратами пам'яті для ігрових, наукових і навчальних застосувань за рахунок розробки методики, що дозволяє об'єднати обидва описані вище підходи в одному кадрі так, щоб результат візуально сприймався як єдина сцена. Для цього було визначено місце маршингу в порядку рендеригу кадру, спосіб узгодження глибин двох підходів та систему обміну тінями й освітленням між ними.

Основний розділ

Систему реалізовано для універсального конвеєра рендерингу URP графічного рушія Unity [4]. Основний шейдер прописано на мові HLSL [5] для подальшого виконання, як повноекранного проходу через спеціалізовану функцію рендерингу рушія Render Feature. Для такої візуалізації, на екран

виводиться один полігон, що покриває весь кадр і для кожного його пікселя обчислюється фрагментний шейдер [6], котрий випускає обчислювальний промінь маршину з камери. Ключовим фактором є момент ін'єкції проходу, він вставляється безпосередньо перед рендерингом прозорих об'єктів і має значення BeforeRenderingTransparents. До цього моменту графічний конвеєр вже виконав усі потрібні підготовчі кроки, як от побудову карти тіней від головного джерела світла, формування текстури глибини сцени, відображення непрозорих полігональних об'єктів та неба. Звідси маємо, що на момент виконання проходу, доступні кольоровий буфер з повністю освтіленою полігональною сценою та буфер глибини. В свою чергу прозорі об'єкти ще не намальовані, тому вони накладаються поверх результату raymarching методу і залишаються видимими крізь нього. За умови, коли б прохід виконувався раніше, то небо та інші непрозорі об'єкти перекрили б результат маршингу. Відповідно при пізнішому виконанні процедурні об'єкти, як от хмари чи рельєф, були б намальовані поверх прозорих об'єктів, що було б візуально некоректним відображенням (рис. 1).

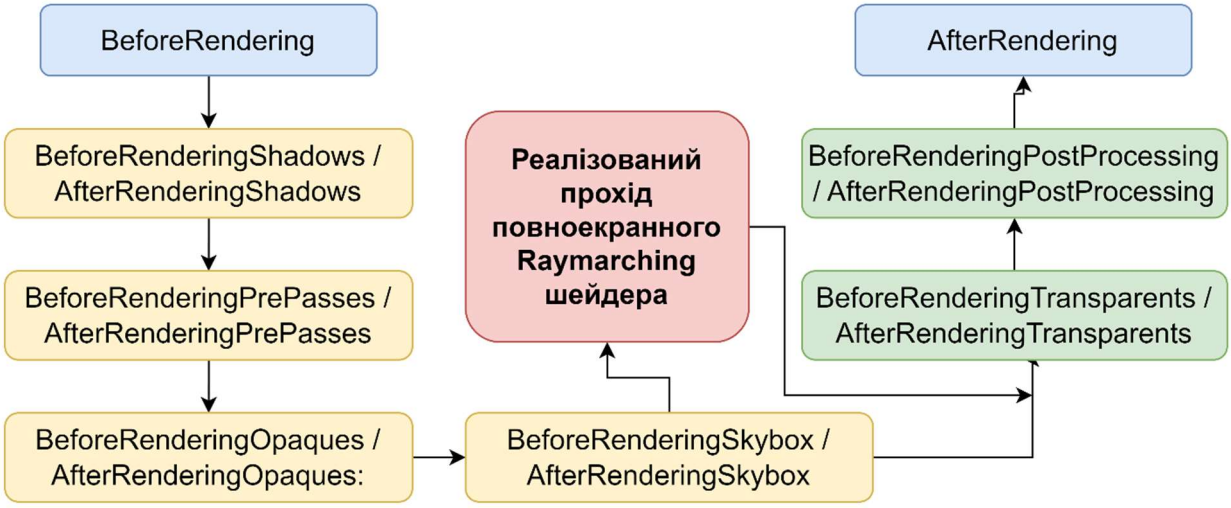


Рисунок 1 – Порядок виконання проходів кадру та місце ін'єкції проходу маршингу променів

Шейдер отримує три джерела вхідних даниз з графічного конверєру рушія, а саме: кольоровий буфер з камери через текстуру _BlitTexture, текстуру глибини сцени через функцію SampleSceneDepth та карту тіней головного джерела світла. Вихідними даними є не лише колір, а й значення глибини записане в буфер SV_Depth. Для цього прохід шейдера налаштовано зі станами Zwrite On, Ztest Always та Cull Off, що позначає вимкнення тесту глибини, оскільки сам шейдер вирішує, що буде видимо в пікселі візуалізації.Однак при цьому запис глибини ввімкнено, щоб результат маршингу променів став частиною буфера глибини для послідуючих етапів обчислень.

Важливою частиною методу є відновлення світової позиції видимої точки світлової сцени, для цього в кожному пікселі перевіряються значення глибини та обернена матриця перегляду проєкції, що доступна в графічному конвеєрі як UNITY_MATRIX_I_VP. Додатково обчислений вектор від камери огляду до обраної точки задає напрямок променя маршингу, а відстань до такої точки буде вважатись максимальною межею, за котрою для цієї точки пікселя вже існує непрозора поверхня, що буде обмежувати подальші обчислення й виводити результат. На відміну від цього, пікселі що належать умовному небу отримують значення глибини рівне заданій далекій площині, котра вважатиметься нескінченною. Наочно цей етап ілюструється в трьох зображеннях. На першому зображені (рис. 2) показано глибину полігональної сцени, що зчитана з текстури глибини й представлена у вигляді градацій сірого. Відповідно на другому рисунку (рис. 3) наведено відображення глибини, що обчислюється лише шейдером маршингу променів для рельєфу. І нарешті, третій рисунок (рис. 4) показує результати порівняння спільної глибини отримані після поєднання обох джерел. Звідси видно, що колір відображення сцени з полігональними об'єктами відмальовувався шейдером лише за умови, що значення полігональної глибини в цій точці було меншим за значення Raymarching, а отже об'єкт ввжатиметься таким, що розташований ближче.

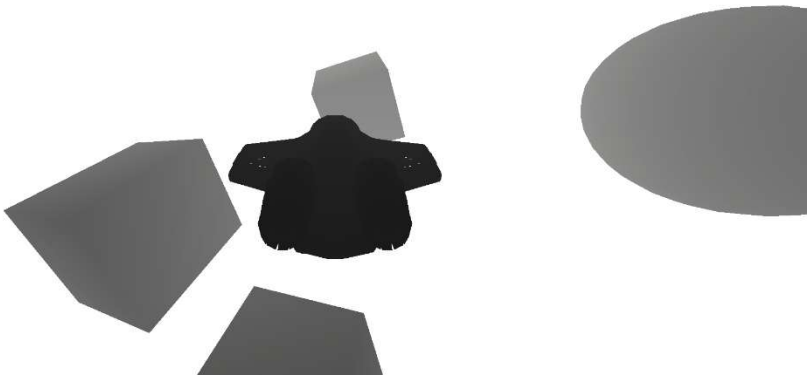


Рисунок 2 – Глибина сцени з полігоальною геометрією



Рисунок 3 – Глибина обчислена шейдером маршингу променів

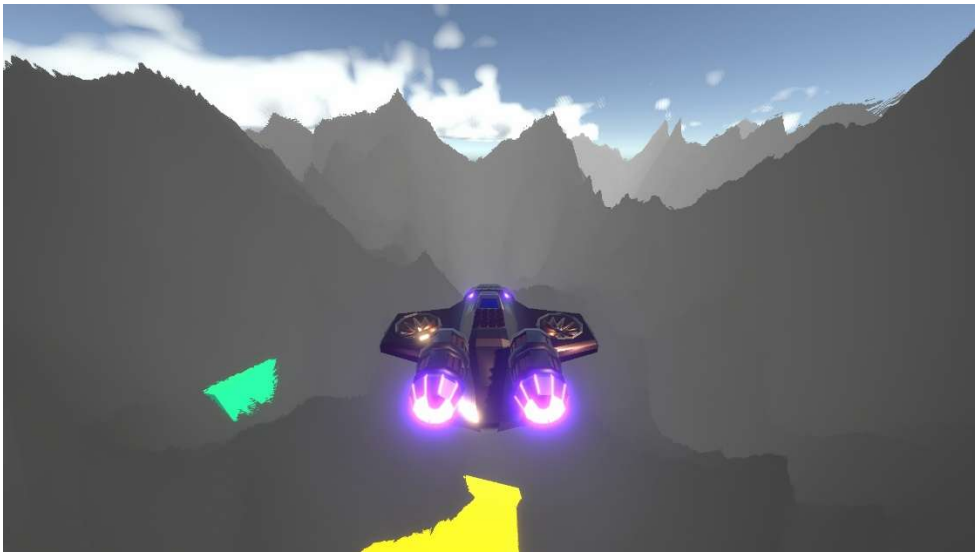


Рисунок 4 – Результати поріаняння глибин та відмальовування відповідних полігональних об’єктів поверх

Оскільки, як згадувалось вище, поєднання виконується попіксельним порівнянням відстаней через глибину, то відносне обмеження обчислення візуалізації маршингу рельєфу на певні відстанні забороняє подальшу перевірку й обчислення сцени, що дозволяє економити обчислювальні ресурси. Тобто якщо перед камерою огляду близько розмістити полігональний об’єкт, що повністю

перекриватиме екран, то обчислення процедурних об’єктів маршингу не відбуватимуться ще на етапі початкових ітерацій променя. Проте, якщо промінь знайде поверхню рельєфу на відстані, що є меншою за значення глибини полігональної сцени, то в цьому пікселі рельєф вважатиметься ближчим до камери і він замінюватиме полігональну поверхню перефарбовуючи піксель на свій колір. Завдяки цьому підходу, лінії перетину процедурного рельєфу з полігональними об’єктами формуються точно так само як і в стандартному тесті глибини, без додаткових обчислень.

Розроблена система використовує маршинг променів для двох різних призначень. Перше використання полягає у візуалізації непрозорих процедурних об’єктів таких, як рельєфи чи інші твердотільні структури. В цьому випадку промінь raymarching рухається кроками, довжина котрих визначається оцінюванням відстанні до поверхні, до перетину з поврехнею. В результаті отримується точна на поверхні з відомою відстанню, що є умовним аналогом результату з растеризації полігональних об’єктів. Тому таку точку можна порівнювати з глибиною полігональної сцени та записувати при перетині в буфер глибини.

В свою чергу другий тип використання це візуалізація напівпрозорих структур, як от об’ємної хмари. Оскільки в такому випадку перетин не є однозначним, то коли промінь проходить увесь відрізок заданого об’єму об’єкта, на кожному кроці обчислюється накопичення кольору за заданими значеннями густини. Обрахування виконуються спереду назад, тобто світло, помножене на поточну накопичену густину, додається до результату, що в результаті зменшує пропускання умовного кольору сцени чи іншої процедурної геометрії. В результаті, через обмеження променя маршингу відстанню до найближчої непрозорої поверхні будь-якого типу візуалізація позаду неї не відображатиметься, а структури перед нею будуть відповідно коректно накладатись. Кінцевий колір обчислень дорівнює накопиченому світлу та кольору поверхні помножених на залишкове пропускання такого об’єкта. Через таку особливість непрозорих об’єктів як хмари, вони не записують глибину, бо не мають єдиної чітко визначеної поверхні.

Загалом розроблена система повторює логіку класичного графічного конвеєра рушія, коли спочатку відмальовуються непрозорі об’єкти, а потім напівпрозорі. Кожен піксель екрану визначає найближчу непрозору поверхню, чи полігональну з кольорового буфера сцени чи процедурну з шейдера, а потім для цієї точки обчислюється освітлення головним джерелом світла сцени та затінення. Тільки після цих дій викнується маршинг напівпрозорих поврехонь. Джерело світла отримується через читання основної функції GetMainLight, що доступна в графічному конвеєрі за замовчуванням, тому освітлення і в шейдерних об’єктів і у полігональних на сцені — узгоджене.

У свою чергу важливим аспектом для забезпечення коректної взаємодії двох різних тривимірних візуалізацій об’єктів є репалізація взаємних тіней. Тіні від полігональних об’єктів на процедурному об’єкті прозорому та непрозорому беруться з уже існуючої карти тіней URP графічного конвеєра обраного рушія. Для кожної точки поверхні чи точки всередині об’ємної непрозорої моделі будується тіньова кордината з використанням базових функцій бібліотеки освітлення. Також обчислюється тінь процедурного об’єкта на самому собі та на полігональній геометрії, для цього використовується вже додатковий прохід маршингу променів від головного джерела світла до відповідної точки, причому різкість тіней обчислюється за співвідношенням мінімальним кроком променя до реально пройденої відстані. Для полігональних об’єктів, тінь накладається наближено: беруться значення з кольорового буферу сцени, що відповідає кольору їх поверхі з уже змішаним кольором прямого й розсіяного світла, після чого перемножується отриманий результат на заданий користувачем відтінок тіні. Тінь від напівпрозорих об’єктів, як от хмари, на будь-якій поверхні також обчислюється маршингом променів, але з урахуванням накопиченої густини через кожен шар хмари. Для цього використовується закон Бугера-Ламберта-Бера [7], про накопичення світла в густих об’єктах, за формулою 1.1. Свою чергу для самозатіннення об’єкта, доводиться використовувати додатковий короткий маршинг світла з урахуванням поверхні об’єкта.

$$T = \exp(-p*\Delta t*k), \quad (1)$$

де p — густина;
Δt — довжина кроку;
k — коефіцієнт поглинання.

Головними обмеженнями розробленої системи є те, що графічний прохід виконується для кожного пікселя екрана, а отже вартість обчислень зростає з роздільною здатністю. Щоб знизити вартість

додається обмеження дальності маршингу та похибка обчислення, зокрема замість пошуку нулевого значення поверхні через знакові функції відстанні SDF [8], на значенні меншому за 0.01 крок зупиняється з результатом поверхні. У випадку якщо загальна пройдена відстань маршингом досягає визначеної користувачем межі, то обчислення також зупиняються але з результатом пустоти, а отже вузалізується колір сцени що з велокою вірогідністю буде небесною сферою рушія. Додатково реалізовано ефект дизерингу через псевдо шум, що накладається на нормалізовані кординати екрану візуалізації, що дозволяє зменшити кількість ітерацій маршингу без видимого розділення процедурних об'єктів на шари обчислення.

Запропонована система має значні перспективи у тому, що не залежить від способу опису об'єкта, оскільки умовну функцію висот рельєфної поверхні можна змінити на знакову функцію відстані фрактала й це все одно забезпечить його перетини з полігональними об'єктами, спільні тіні та освітлення. Це дозволяє розглядати розроблену систему, як основу для фізуалізацій фрактальних, динамічних, аморфних та інших процедурних структур у складі звичайних тривимірних сцен.

Висновки

У роботі описано методику поєднання полігонального рендерингу та маршингу променів у середовищі Unity. Ін'єкція повноекранного проходу перед рендерингом прозорих об'єктів дозволяє використовувати вже готові кольовий буфер і буфер глибини, а запис значень глибини обчислених із шейдера забезпечує коректну взаємодію результату з прозорими об'єктами. Попіксельне порівняння відсатней гарантує правильні перетини об'єктів, а спільне джерело світла, карти тіней та додаткові маршинги забезпечують взаємні тіні, затінення й узгоджене освітлення полігональної та процедурної геометрії. Такий підхід дозволяє отримувати високодеталізовані сцени з мінімальними витратами пам'яті та придатний для ігрових, наукових і навчальних застосувань.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Ashbery S. Rasterization [Електронний ресурс] / Simon Ashbery // Medium. — 2019. — Режим доступу: https://si-ashbery.medium.com/rasterization-13287cee9ab4.
2. Spatial Team. The Main Benefits and Disadvantages of Polygonal Modeling [Електронний ресурс] // Spatial Blog. — 2019. — Режим доступу: https://blog.spatial.com/the-main-benefits-and-disadvantages-of-polygonal-modeling.
3. Heckel M. Painting with Math: A Gentle Study of Raymarching [Електронний ресурс] / Maxime Heckel // Maxime Heckel's Blog. — 2023. — Режим доступу: https://blog.maximeheckel.com/posts/painting-with-math-a-gentle-study-of-raymarching/.
4. Unity Documentation [Електронний ресурс]. Режим доступу: https://docs.unity.com/.
5. High-level shader language [Електронний ресурс]. Режим доступу: https://learn.microsoft.com/en-us/windows/win32/direct3dhlsl/dx-graphics-hlsl.
6. Shader Basics: Fragment Shader [Електронний ресурс] // GPU Shader Tutorial. — Режим доступу: https://shader-tutorial.dev/basics/fragment-shader/.
7. The Beer-Lambert Law [Електронний ресурс] // Edinburgh Instruments Spectral School. — 2021. — Режим доступу: https://www.edinst.com/resource/the-beer-lambert-law/.
8. Raymarching SDFs [Електронний ресурс]. Режим доступу: https://iquilezles.org/articles/raymarchingdf/.

Бурдейний Олег Володимирович — студент групи 1КІ-25м, факультет інформаційних технологій та комп'ютерної інженерії, Вінницький національний технічний університет, Вінниця, e-mail: burdeyniy.007.com@gmail.com

Черняк Олександр Іванович – кандидат технічних наук, доцент кафедри обчислювальної техніки Вінницького національного технічного університету, Вінниця, e-mail: chernyak@vntu.edu.ua

Burdeinyi Oleh Volodymyrovych. — student of group 1CE-25m, Faculty of Information Technology and Computer Engineering, Vinnytsia National Technical University, Vinnytsia, e-mail: burdeyniy.007.com@gmail.com

Cherniak Oleksandr Ivanovych — PhD, associate professor of the Department of Computer Engineering, Vinnytsia National Technical University, Vinnytsia, e-mail: chernyak@vntu.edu.ua