УДК 004.925
DOI: 10.25686/978-5-8158-2474-4-2025-183-191

Разработка и анализ метода отражений в экранном пространстве для систем с дискретным и интегрированным GPU

Т. Р. Галеев, М. К. Богданов
Университет ИТМО, Санкт-Петербург, Россия

Аннотация. В работе рассмотрены возможности распределения вычислительной нагрузки при рендеринге отражений в экранном пространстве (Screen Space Reflections, SSR) между дискретным и интегрированным GPU. Проведен обзор основных подходов к мульти-GPU рендерингу (Alternate Frame Rendering, Split Frame Rendering, Hybrid Multi-GPU Rendering) и методов построения отражений (геометрические техники, environment-mapping, SSR, трассировка лучей). На основании анализа выбран гибридный подход, при котором дискретная видеокарта отвечает за основной рендеринг сцены, а интегрированная – за вычисление отражений методом SSR. Описана архитектура графического приложения, реализующего предложенный подход, и проведены рекомендации по оптимизации параметров алгоритма для минимизации задержек и повышения производительности.

***Ключевые слова*:** мульти-GPU, интегрированный GPU, дискретный GPU, экранные отражения, Screen Space Reflections, cross-adapter ресурсы, синхронизация fence, гибридный рендеринг

Development and analysis of a screen space reflection method for systems with discrete and integrated GPUs

T. R. Galeev, M. K. Bogdanov
ITMO University, St. Petersburg, Russia

Abstract. This work investigates the distribution of computational load for Screen Space Reflections (SSR) between a discrete and an integrated GPU. It provides a survey of multi-GPU rendering techniques (Alternate Frame Rendering, Split Frame Rendering, Hybrid Multi-GPU Rendering) and reflection methods (geometry-based techniques, environment mapping, SSR, real-time ray tracing). Based on the analysis, a hybrid approach is adopted where the discrete GPU performs the main scene rendering, and the integrated GPU computes SSR reflections. The architecture of a graphical application implementing the proposed approach is described, and recommendations for optimizing the algorithm parameters to minimize delays and improve performance are made.

***Keywords*:** multi-GPU, integrated GPU, discrete GPU, screen space reflections, SSR, cross-adapter resources, fence synchronization, hybrid rendering

Введение

Современным видеоиграм предъявляют высокие требования к качеству визуализации. Одним из главных критериев хорошей 3D-графики является наличие отражений окружающих объектов на поверхностях, которое добавляет визуализируемой 3D-сцене реалистичности. Среди различных методов реализации отражений особое место занимает метод отражений в экранном пространстве (Screen Space Reflections, SSR), который позволяет генерировать отражения в реальном времени, основываясь на информации, доступной в буфере изображения, что делает его одним из самых быстрых методов построения отражений. Однако рендеринг сцены с отражениями, даже при помощи SSR, может существенно повлиять на производительность в условиях использования одного графического процессора.

Современные пользовательские ЭВМ часто имеют как дискретные GPU, обеспечивающие высокую вычислительную мощность, так и интегрированные GPU, встроенные в центральный процессор или материнскую плату и обеспечивающие базовый уровень графики. Эти два типа GPU значительно различаются по архитектуре, производительности и объему памяти. В частности, интегрированные GPU могут испытывать трудности с обработкой сложных эффектов, в то время как дискретные GPU предлагают значительно больше возможностей для их реализации. По этой причине вся вычислительная нагрузка во время 3D-рендеринга приходится на дискретную видеокарту, однако в условиях использования устаревшего оборудования добиться качественных изображений и одновременно высокой производительности системы крайне проблематично, в то время как интегрированный GPU практически не используется. Это создает проблему распределения вычислительной нагрузки между дискретным и интегрированным GPU для обеспечения высокой производительности рендера в реальном времени.

Развитие современных графических API, таких как DirectX и Vulkan [1], предоставило новые возможности для одновременного управления несколькими GPU [2], что открывает перспективы повышения производительности графической визуализации в контексте игровых приложений путём синхронной работы интегрированной и дискретной видеокарт.

Постановка задачи

Целью данной работы является исследование перспективы разделения вычислительной нагрузки рендеринга 3D-сцены между дискретной и интегрированной видеокартами для повышения производительности визуализации в реальном времени, а также разработка графического приложения, использующего дискретную видеокарту для основного рендеринга 3D-сцены и интегрированную видеокарту для рендеринга отражений с помощью метода SSR.

Для достижения данной цели были поставлены следующие задачи:

1) исследование подходов Multi-GPU рендеринга для синхронной работы дискретных и интегрированных видеокарт;

2) исследование методов построения отражений;

3) программная реализация функционала графического приложения для визуализации 3D-сцены с использованием дискретных и интегрированных видеокарт для исследования производительности выбранного подхода;

4) сравнительный анализ полученных результатов и оценка применения выбранного подхода в игровых приложениях.

Теория

1. Методы Multi-GPU рендеринга

Современные алгоритмы взаимодействия между несколькими графическими процессорами позволяют распределять вычислительную нагрузку между несколькими GPU для увеличения производительности графической визуализации [2]. Рассмотрим три ключевых подхода: Alternate Frame Rendering, Split Frame Rendering и Hybrid Multi-GPU Rendering.

Alternate Frame Rendering (AFR) – это метод рендеринга для систем с несколькими графическими процессорами, при котором каждый GPU обрабатывает полный кадр, но выполняет это попеременно [2]. Например, один GPU рендерит нечётные кадры, а другой – чётные [3]. В то время как один GPU завершает рендеринг кадра, другой начинает работу над следующим кадром, что обеспечивает высокий уровень параллелизма (рис. 1) [4].

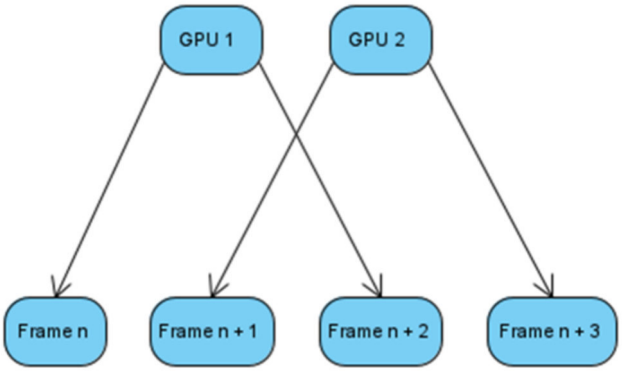

Рисунок 1. Работа алгоритма AFR

Split Frame Rendering (SFR) – это метод рендеринга для систем с несколькими графическими процессорами, при котором один кадр делится на несколько частей, каждая из которых обрабатывается отдельным GPU. В отличие от AFR, SFR распределяет рендеринг внутри одного кадра, а не между кадрами [5]. Ключевая идея SFR заключается в том, чтобы разделить экран на области (например, горизонтальные или вертикальные полосы) и распределить их между GPU [4]. Каждый GPU рендерит свою область параллельно, а затем результаты объединяются для вывода на экран (рис. 2).

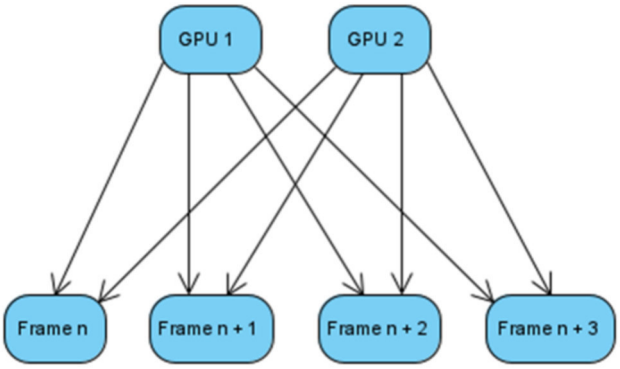

Рисунок 2. Работа алгоритма SFR

Основная идея Hybrid Multi-GPU Rendering заключается в том, чтобы использовать несколько графических процессоров, где каждый GPU выполняет задачи, наиболее подходящие для его архитектуры. Примером может быть сценарий, где один GPU выполняет рендеринг основной геометрии сцены, а другой GPU обрабатывает специфические задачи, такие как расчёт освещения, тени или отражения. Такой подход особенно актуален в системах с различными по мощности GPU, например, дискретными и интегрированными.

Пример данного алгоритма, представленный на рисунке 3, показывает, как можно выиграть время на обработку каждого кадра, разделив задачи рендеринга на две видеокарты, различающиеся по мощности. Основная часть всех графических задач выполняется на основной видеокарте. Она введет расчет различных текстур, таких как Карта нормалей и Карта локального затенения, которые необходимы для построения сцены. В это время дополнительное устройство занимается просчетом карты теней и копирует полученные данные в память ведущей видеокарты для последующего финального построения изображения сцены [6].

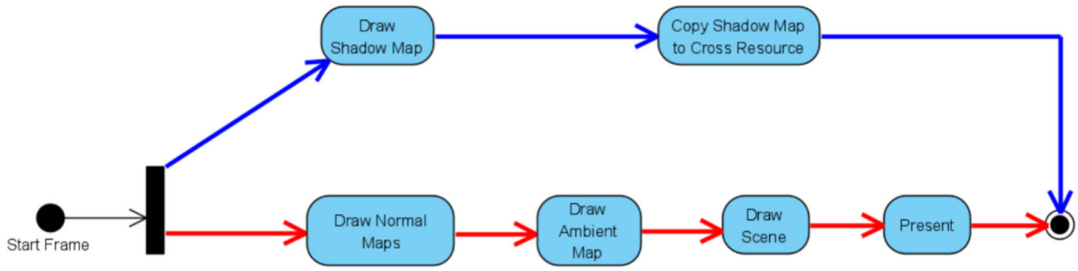

Рисунок 3. Алгоритм Shared Shadow Map

Проанализировав каждый подход Multi-GPU взаимодействия, можно сделать вывод о том, что AFR и SFR не подходят для реализации графического приложения в данной работе, так как не учитывают разницу в мощности между дискретным и интегрированным GPU. В AFR производительность будет ограничена слабой мощностью интегрированной видеокарты. А в SFR интегрированной видеокарте необходимо также пройти по всем ресурсам, что и основной видеокарте, чтобы сформировать свой кусок изображения, что значительно снизит производительность. Hybrid Multi-GPU Rendering является наиболее подходящим методом для реализации, так как учитывает малую мощность интегрированной GPU.

2. Техники построения отражений

Основные подходы делятся на три категории: геометрические, основанные на изображениях, и трассировка лучей.

Геометрические методы (Planar Surfaces, Curved Surfaces) используют трансформацию самой геометрии объектов, чтобы создать отражения [7]. Это значит, что создаются виртуальные объекты, которые размещаются так, чтобы имитировать отражение относительно отражающей поверхности. Подходы для плоских и криволинейных поверхностей различаются.

Для создания отражений на плоской поверхности (Planar Surfaces) требуется только одно линейное преобразование, поскольку нормаль к плоскости отражения остаётся неизменной. Для каждого объекта сцены создаётся виртуальная копия, которая отражается относительно плоскости (рис. 4). Это выполняется с помощью матрицы трансформации [7, 8].

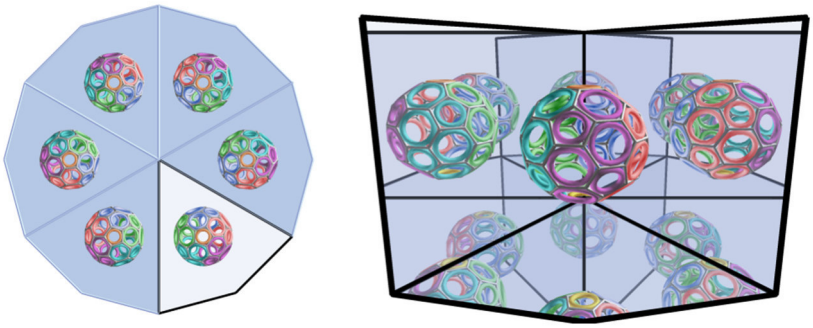

Рисунок 4. Пример построения отражений с помощью Planar Surfaces

Для создания отражений на кривой поверхности (Curved Surfaces) сложность возрастает, поскольку отражения зависят от точки обзора. Для расчета отражения каждого вершинного элемента применяются дополнительные алгоритмы. Для каждой вершины объекта находят пересечение луча зрения с поверхностью отражателя (рис. 5) [7, 9].

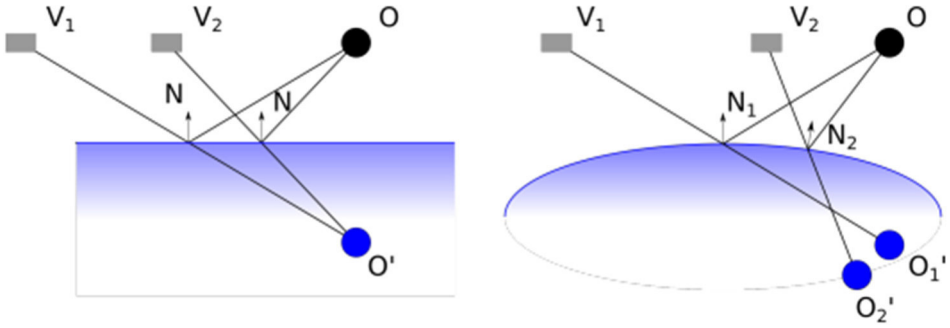

Рисунок 5. Пример построения отражений с помощью Curved Surfaces

Методы, основанные на изображениях (Cube Mapping, Sphere Mapping, SSR), используют текстуры (или уже отрендеренные изображения сцены) для симуляции отражений. Эти техники более просты и эффективны для отображения отражений на криволинейных поверхностях, где точное геометрическое моделирование слишком сложное.

Отражение окружающей среды предполагает, что сцена проецируется на текстуру, которая затем применяется к отражающим объектам. Отражение создаётся с использованием заранее подготовленной текстуры [7]. При построении отражений с помощью кубической карты (Cube Mapping) сцена визуализируется из центра отражающего объекта в шесть направлений (по осям X, Y, Z) для создания шести текстур, формирующих куб. Кубические текстуры используются для определения цвета отражения на основе направления отражённого луча. Этот метод популярен благодаря простоте реализации, а также высокой производительности (рис. 6) [7, 10]. При построении отражений с помощью сферической карты (Sphere Mapping) отражения кодируются в одной текстуре, проецируемой на сферу. Текстура затем преобразуется в плоскую 2D-карту. Преимущество данного метода в том, что требуется только одна текстура, а не шесть, как в кубическом методе. Однако значительная потеря точности на краях текстуры, а также неравномерность выборки пикселей (линейная интерполяция на сфере создаёт артефакты) [7].

Screen-Space Reflections (SSR) – это метод, который работает в пространстве экрана и использует уже отрендеренную сцену для создания отражений [11]. Сначала сцена рендерится в G-Buffer – структуру данных, содержащую глубину каждого пикселя, нормали поверхностей и положение объектов. Для каждого пикселя экрана рассчитывается отражённый луч на основе его нормали. Луч

отслеживается в пространстве экрана, где проверяется его пересечение с данными глубины в G-Buffer. Как только находится пересечение, цвет соответствующего пикселя используется для отображения отражения, как показано на рисунке 7 [12, 13].

Методы трассировки лучей (Real-Time Ray Tracing) являются наиболее точным способом создания отражений, так как они моделируют физическое поведение света (рис. 8). В процессе лучи отслеживаются от наблюдателя через отражающую поверхность к объектам, которые они пересекают [7].

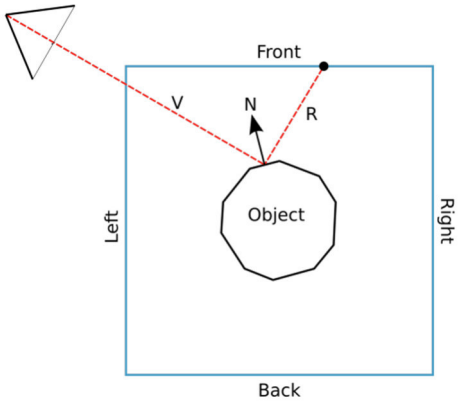

Рисунок 6. Пример построения отражений с помощью Cube Mapping

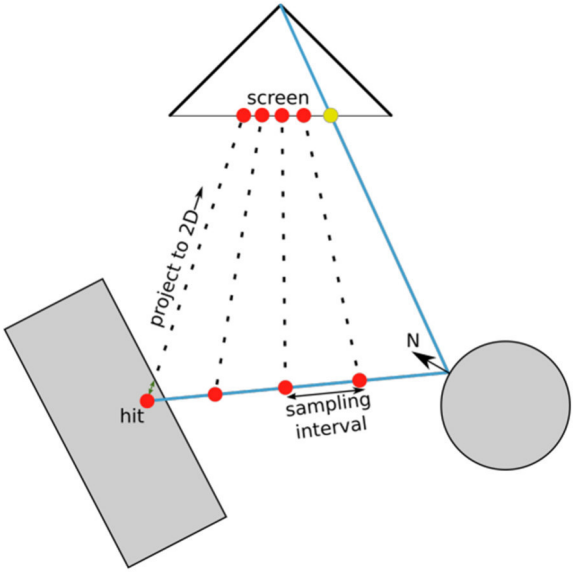

Рисунок 7. Алгоритм построения отражений с помощью SSR

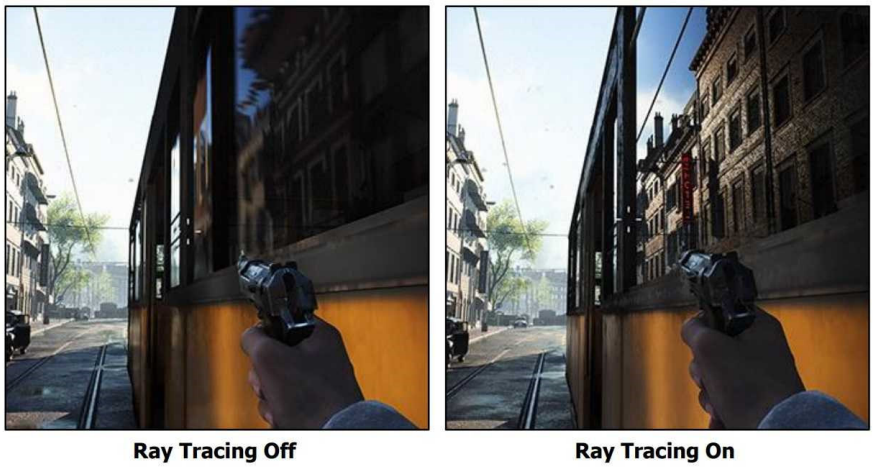

Рисунок 8. Пример работы Ray Tracing

Для каждого пикселя экрана рассчитывается луч, который начинает своё движение от точки обзора и пересекает сцену. Луч проверяется на пересечения с геометрическими объектами в сцене, чтобы определить ближайшую точку контакта. Если поверхность в этой точке зеркальная, рассчитывается направление отражённого луча, который отслеживается дальше. Процесс повторяется для заданного числа отражений, чтобы избежать бесконечных вычислений (рис. 9). Одно из преимуществ трассировки лучей – способность моделировать многократные отражения, т.е. многократное отражение лучей между поверхностями. Это автоматически учитывается при каждом отскоке луча.

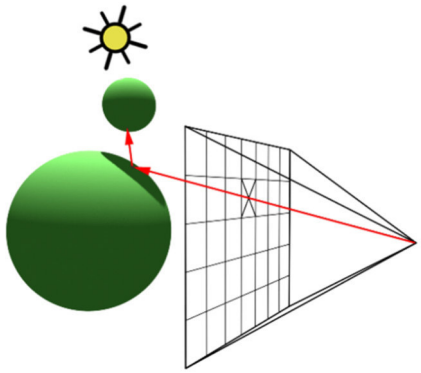

Рисунок 9. Пример построения отражений с помощью Ray Tracing

Проанализировав различные алгоритмы построения отражений с помощью дополнительной видеокарты, можно сделать вывод о том, что наиболее подходящим методом для реализации графического приложения с использованием гибридного Multi-GPU подхода является Screen Space Reflections, так как обеспечивает отражения динамических объектов в реальном времени, а также высокую производительность отрисовки.

Planar Surfaces не подходит, так как, несмотря на высокую производительность, работает только для плоских отражающих поверхностей. Curved Surfaces и Ray Tracing не подходят, так как имеют высокую вычислительную сложность, что не подходит для интегрированной GPU с маленькой вычислительной мощностью. Техники Environment Mapping, такие как Cube Mapping и Sphere Mapping, не подходят, так как отражения строятся для статических объектов.

Предложенный метод

На основе выводов анализа алгоритмов Multi-GPU рендеринга в данной работе предлагается использование гибридного подхода, учитывающего малую мощность интегрированной GPU. Для начала необходимо разобрать текущий пайплайн графического приложения, представленный на рисунке 10.

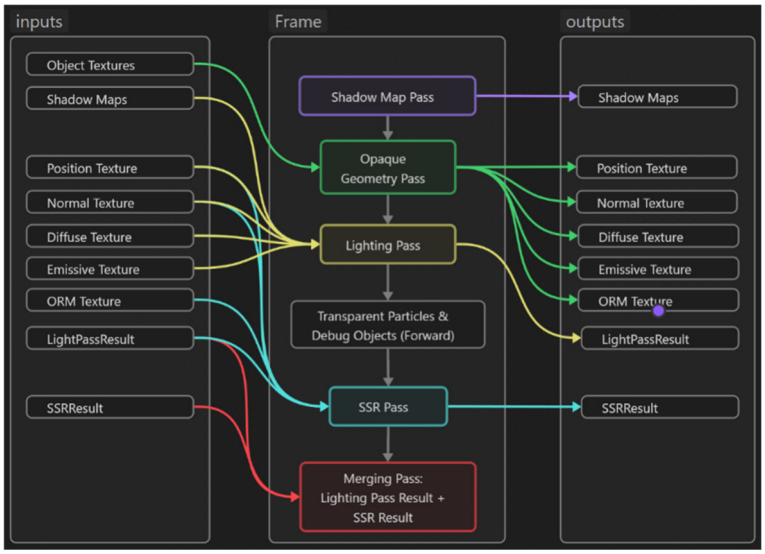

Рисунок 10. Рендер граф Single-GPU графического приложения

На данном графе можно увидеть, что этап вычисления отражений SSR Pass находится в конце графического пайплайна, так как требует G-Buffer текстуры позиций, нормалей и текстуру ORM (Occlusion, Roughness, Metalness), а также результат Lighting Pass в виде текстуры. Затем в Merging Pass две текстуры SSR Pass Result и Lighting Pass Result смешиваются воедино.

Для вычисления отражений на дополнительной видеокарте нужно подготовить и передать на неё необходимые текстуры GBuffer, а также текстуру Light Pass Result. Результат работы SSR Pass должен отправляться на основную видеокарту для дальнейшего смешивания в Merging Pass. Таким образом, рендер граф системы будет выглядеть, как на рисунке 11.

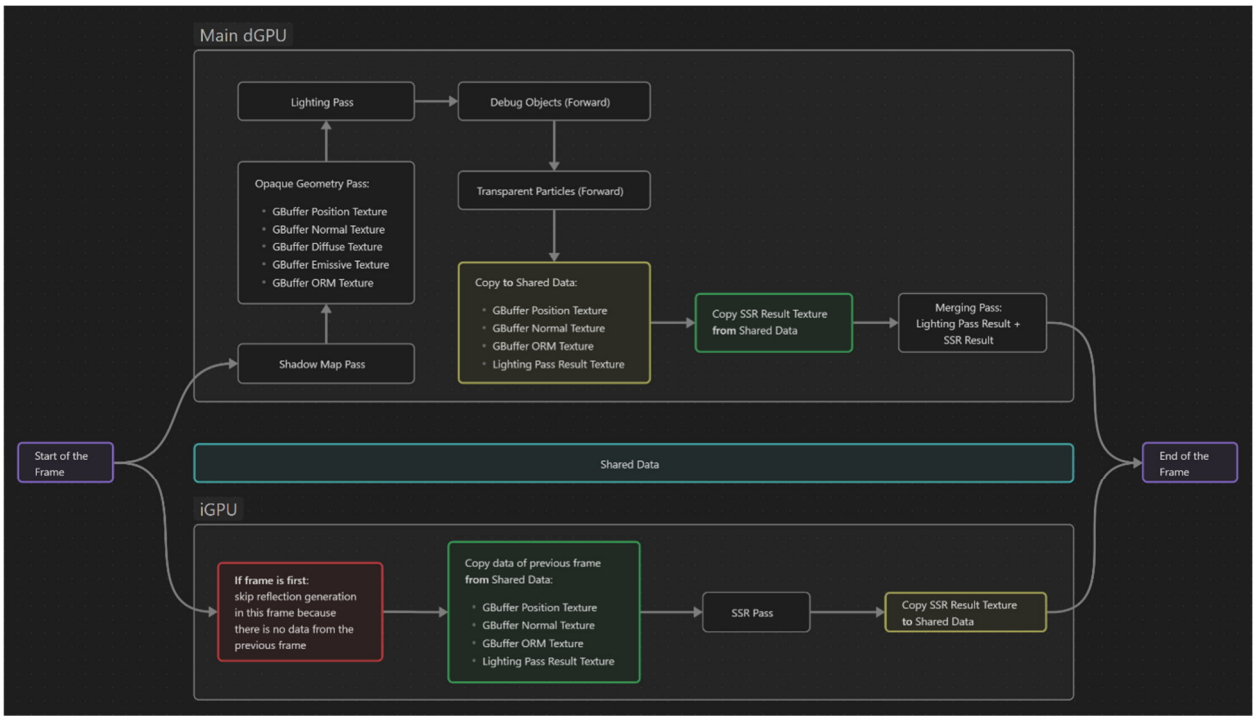

Рисунок 11. Рендер граф Multi-GPU графического приложения

Shared Data является «буферной зоной» для передачи данных между двумя видеокартами. Зеленым и желтым цветами выделены соответственно этапы загрузки ресурсов в Shared Data и выгрузки из неё. Генерация отражений в первом кадре пропускается, так как основная видеокарта ещё не рассчитала данные для G-Buffer и Lighting Pass Result, необходимые для SSR Pass. Когда результаты Geometry Pass и Lighting Pass из предыдущего кадра будут получены интегрированной видеокартой, она выполнит SSR Pass и скопирует результат в Shared Data. После этого в следующем кадре основная видеокарта скопирует SSR Pass Result, посчитанный в предыдущем кадре, из Shared Data.

Результаты экспериментов

На основе полученного Multi-GPU пайплайна была проведена предварительная реализация графического приложения с использованием всех вышеперечисленных этапов (рис. 12).

Также был проведен предварительный эксперимент сравнения производительности подходов Single-GPU и Hybrid Multi-GPU в условиях сильной загруженности основной видеокарты. Результаты эксперимента представлены на рисунке 13.

Рисунок 12. Предварительная реализация графического приложения с отражениями

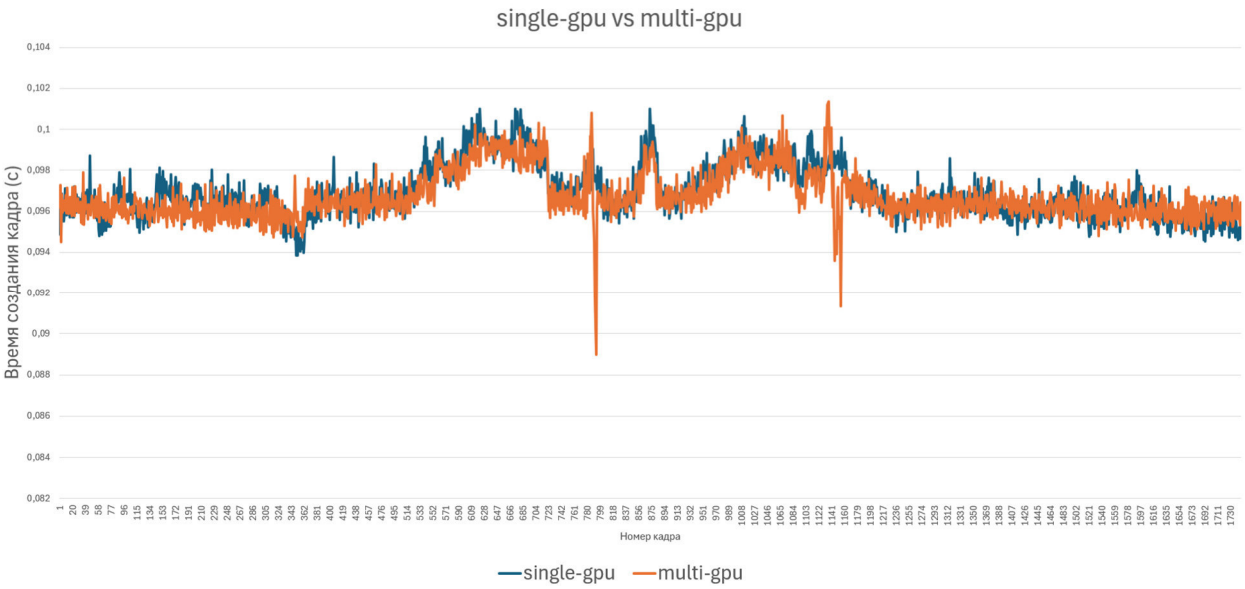

Рисунок 13. Сравнение производительности подходов Single-GPU (RTX 4070 Super) и Hybrid Multi-GPU (AMD iGPU + RTX 4070 Super)

Обсуждение результатов

Предварительный эксперимент сравнения производительности подходов показал, что подход Hybrid Multi-GPU более производительный, чем Single-GPU. Однако такие результаты были получены в условиях сильной нагрузки на основную видеокарту. Даже при том, что при Hybrid Multi-GPU основная видеокарта не тратит время на вычисление отражений, она всё равно тратит какое-то время на копирование ресурсов в общую память и из неё. Если нагрузка на основную видеокарту слабая, а время на рендер отражений на интегрированной видеокарте и на копирование ресурсов между графическими устройствами превосходит время рендера на основной видеокарте, то целесообразнее использовать Single-GPU рендеринг на основной видеокарте.

Предполагается, что дальнейшее улучшение и оптимизация гибридного подхода multi-GPU взаимодействия минимизирует простой на дискретной и интегрированной видеокартах, а также

повысит производительность системы в условиях равного времени работы обеих видеокарт. Однако при таком подходе отражения 3D-сцены будут всегда отставать на два кадра от самой сцены, что некритично при большом количестве кадров в секунду. Также возможным решением для минимизации простоев является динамическая настройка параметров для SSR, таких как RayStepLength и MaxDistance, от которых зависит время выполнения работы алгоритма.

Выводы

В ходе данной работы проведен анализ подходов Multi-GPU рендеринга, в результате чего для реализации графического приложения был выбран алгоритм Hybrid Multi-GPU Rendering. Также осуществлен анализ методов построения отражений, в результате чего для реализации отражений был выбран алгоритм SSR.

Результаты проведенного исследования и предварительная реализация показали, что гибридный Multi-GPU подход с переносом вычислений SSR на интегрированный GPU позволяет снизить нагрузку на основную видеокарту и добиться прироста производительности в условиях её высокой загрузки. Результаты экспериментов подтвердили, что при достаточной нагрузке на дискретный GPU гибридная схема превосходит классический Single-GPU рендеринг, тогда как при низкой нагрузке рендеринг на дополнительной видеокарте может снизить производительность. Дальнейшие работы будут направлены на динамическую адаптацию параметров SSR и оптимизацию графического приложения для уменьшения задержек и выравнивания нагрузки между GPU.

Список литературы

1. Szczerbiński A. Multithreaded game engine architecture. Wrocław, Poland: Wrocław University of Science and Technology, 2021. P. 16.

2. Sjöholm J. Explicit Multi GPU Programming with DirectX 12 // Game Developers Conference. 2016.

3. Aguaviva R., Baker D. Explicit DirectX 12 Multi GPU rendering // Game Developer Conference. 2017.

4. Roca J., Grossman M. Scaling of 3D game engine workloads on modern multi-GPU systems // Proceedings of the Conference on High Performance Graphics 2009. New York, USA: ACM, 2009. P. 37–46.

5. Кожемяко А. История и перспективы multi-GPU на рынке игровой графики. 2023. URL: https://www.ixbt.com/3dv/multi-gpu-pc-retrospect.html (дата обращения 29.12.2024).

6. Богданов М. Разработка и анализ алгоритмов мульти-GPU рендеринга с использованием дискретного и интегрированного GPU. Санкт-Петербург: Университет ИТМО, 2021.

7. Clemenz C., Weydemann L. Reflection Techniques in Real-Time Computer Graphics // KoG. 2021. Vol. 25, no. 25. P. 87–95.

8. McReynolds T., Blythe D. Advanced Graphics Programming Using OpenGL. Elsevier, 2005. P. 407–411.

9. Ofek E., Rappoport A. Interactive reflections on curved objects // Proceedings of the 25th annual conference on Computer graphics and interactive techniques. New York, USA: ACM Press, 1998. P. 333–342.

10. Cube Mapping. 2018. URL: https://research.ncl.ac.uk/game/mastersdegree/graphicsforgames/cubemapping/Tutorial%2013%20-%20Cube%20Mapping.pdf (дата обращения 29.12.2024).

11. Beug A. Screen Space Reflection Techniques. Regina, Canada: University of Regina, 2020.

12. Roberts J. Screen Space Reflections (SSR). 2023.

13. De Macedo D. V., Serpa Y. R., Rodrigues M. A. F. Fast and Realistic Reflections Using Screen Space and GPU Ray Tracing – A Case Study on Rigid and Deformable Body Simulations // Computers in Entertainment (CIE). New York, USA: ACM, 2018. Vol. 16, no. 4. P. 1–18.

14. Гвоздкова У., Мойсейчук А. Алгоритм рейтрейсинга в современных играх // Информационные технологии и управление: материалы 59-й научной конференции аспирантов, магистрантов и студентов (2023). Минск: БГУИР, 2023. 117 с.

15. Redmon J. RAY TRACING – Middlebury. Middlebury College, USA, 2011.