АЛГОРИТМ ИНСТАНЦИРОВАНИЯ ОБЪЕКТОВ С РАЗНЫМИ МАТЕРИАЛАМИ

Барилко М.А., Алейник И.С.

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Красковский П.Н. – ст. преподаватель

Исследуется метод инстанцирования для оптимизации отрисовки множества объектов и его ограничение в игровом движке Unreal Engine на использование одного материала. Представление концепции для решение данного ограничения, применяя массивы текстур и передавая индекс текстуры как атрибут экземпляра.

Инстанцирование (instancing или «создание экземпляров») – это метод оптимизации вызова отрисовки, который визуализирует несколько копий объекта с тем же материалом в одном вызове отрисовки. Каждая копия объекта называется экземпляром(instance). Это полезно для рисования вещей, которые появляются несколько раз в сцене, например, деревьев или кустов, для того чтобы не делать отдельный вызов для каждой копии [1]. Основная цель инстанцирования – оптимизация производительности, особенно в сценах с большим количеством одинаковых или похожих объектов. Основная проблема которую он решает – это то, что при каждом вызове отрисовки от ЦПУ требуется выполнения определенной работы: подготовка состояния графического процессора, указание буферов данных, отправка команды на отрисовку, а когда на сцене нужно отобразить тысячи или миллионы похожих объектов (трава, деревья, солдаты в толпе, камни), то вся система начинает упираться в возможности центрального процессора и пропускную способность соединения с графическим ускорителем.

Основной принцип, по которому происходит инстанцирование: данные базового объекта загружаются в память видеокарты один раз; создается дополнительный буфер, который содержит уникальные данные для каждого отдельного экземпляра (например, его мировая матрица преобразования (позиция, поворот, масштаб), цвет, идентификатор текстуры и т.д.). ЦПУ отправляет одну специальную команду отрисовки, указывая базовую геометрию и количество экземпляров, которые нужно нарисовать. Видеокарта для каждого экземпляра (от 0 до N – 1, где N – количество экземпляров) выполняет вершинный шейдер. В шейдере доступны как атрибуты базовой геометрии, так и уникальные атрибуты текущего экземпляра (благодаря специальной настройке шага чтения из буфера атрибутов экземпляров). Вершинный шейдер вычисляет финальную позицию вершины в пространстве, используя и базовые, и уникальные для экземпляра данные. На рисунке 1 представлена разница без использования инстанцирования и с ним. Можно заметить, что при использовании инстанцирования уменьшается использование памяти, уменьшается время отрисовки и количество вызовов отрисовки.

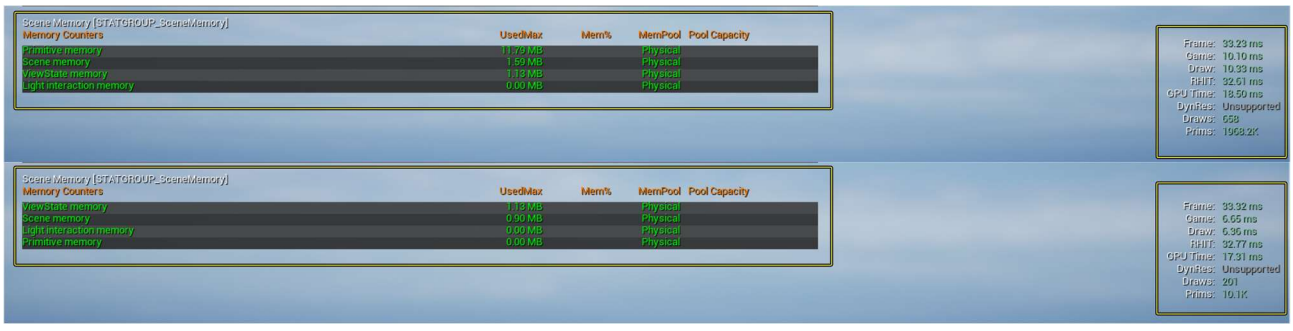

Рисунок 1 – Сравнение работы без использования инстанцирования и с использованием [2]

Этот общий принцип инстанцирования находит свою реализацию в различных графических API и игровых движках, однако конкретные реализации могут иметь свои особенности и ограничения. Например, рассматривая стандартные инструменты инстанцирования в популярном движке Unreal Engine, такие, как компоненты Instanced Static Mesh (ISMC) и Hierarchical Instanced Static Mesh (HISM), необходимо учитывать важное ограничение: все экземпляры внутри одного компонента обязаны использовать один и тот же ресурс модели и, как следствие, одни и те же ресурсы материалов. Прямая возможность назначить разные ресурсы материалов отдельным экземплярам в рамках одного компонента и одного вызова отрисовки не предусмотрена. Это ограничение напрямую связано с целью оптимизации, так как смена материалов является ресурсоемкой операцией для графического

процессора [2]. Однако данное ограничение становится существенным препятствием при создании сложных и визуально насыщенных сцен, где требуется высокий уровень детализации и разнообразия даже для массовых объектов. Например, при визуализации городских кварталов с различными вариациями зданий, лесных массивов с отличающимися по состоянию деревьями (сухие, молодые, покрытые мхом) или толп персонажей с разной одеждой, стандартный подход требует либо компромиссов в визуальном качестве, либо отказа от преимуществ инстанцирования для этих объектов, что приводит к падению производительности.

В рамках разработки программного средства ßaron для разработки игровых приложений на базе графического API Vulkan была поставлена задача преодолеть это ограничение. Был предложен подход, который, сохраняя фундаментальные преимущества инстанцирования, позволяет добиться визуального разнообразия, эквивалентного использованию разных материалов для отдельных экземпляров. Ключевая идея решения заключается в использовании массивов текстур.

В контексте Vulkan API, массивы текстур представляют собой специализированный тип ресурса изображений (VkImage), позволяющий объединить множество двумерных текстурных слоев в единую логическую сущность. Данная структура данных характеризуется строгим требования составляющих ее слоев: все они должны обладать идентичными размерами (ширина и высота), одинаковым форматом пиксельных данных (VkFormat) и идентичным количеством уровней детализации (MIP-уровней). Создание такого ресурса осуществляется посредством задания параметра arrayLayers в структуре VkImageCreateInfo значением, превышающим единицу, при сохранении imageType равным VK_IMAGE_TYPE_2D [3]. Остаётся только добавить в буфер с уникальными данными для каждого экземпляра переменную, хранящую в себе идентификатор необходимого материала. На рисунке 2 представлен результат реализации подхода.

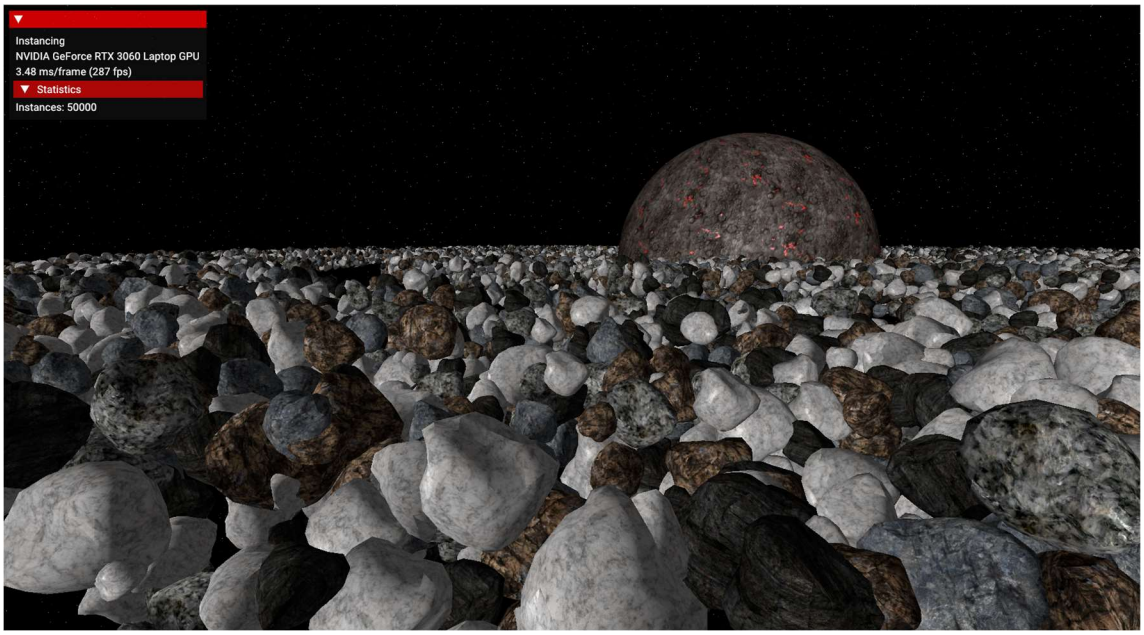

Рисунок 2 – Конечный результат реализации подхода [4]

Таким образом, реализация инстанцирования в программном средстве ßaron, использующая массивы текстур (Texture Arrays) в рамках Vulkan API, эффективно решает проблему ограничения стандартных подходов, требующих использования одного материала для всех экземпляров в рамках одного вызова отрисовки. Это позволяет достичь необходимого уровня визуального разнообразия в сценах с большим количеством однотипных объектов, не жертвуя при этом ключевыми преимуществами инстанцирования – существенным снижением нагрузки на ЦПУ и оптимизацией процесса отрисовки. Данный подход демонстрирует гибкость Vulkan API и открывает возможности для создания более разнообразных виртуальных миров при сохранении высокой производительности.

Список использованных источников:

1. Introduction to GPU instancing. [Электронный ресурс]. – Режим доступа: https://docs.unity3d.com/Manual/GPUInstancing.html – Дата доступа: 03.04.2024.

2. Instanced Static Mesh Component [Электронный ресурс]. – Режим доступа: https://dev.epicgames.com/documentation/en-us/unreal-engine/instanced-static-mesh-component-in-unreal-engine – Дата доступа: 03.04.2024.

3. Vulkan Specification. Images [Электронный ресурс]. – Режим доступа: https://registry.khronos.org/vulkan/specs/latest/html/vkspec.html#VkImage – Дата доступа: 04.04.2024.

4. Vulkan Samples [Электронный ресурс] – Режим доступа: https://github.com/KhronosGroup/Vulkan-Samples – Дата доступа: 07.04.2024.