УДК 004.925
DOI: 10.15827/0236-235X.116.069-072
Дата подачи статьи: 15.09.16
2016. Т. 29. № 4. С. 69–72

ОТОБРАЖЕНИЕ ТРЕХМЕРНЫХ ОБЪЕКТОВ С ИСПОЛЬЗОВАНИЕМ КЛАСТЕРНОЙ ВИЗУАЛИЗАЦИИ

А.М. Гиацинтов, научный сотрудник, algts@inbox.ru;
К.А. Мамросенко, к.т.н., руководитель, kirillam@ya.ru
(Центр визуализации и спутниковых информационных технологий НИИСИ РАН, Нахимовский просп., 36, корп. 1, г. Москва, 117218, Россия)

В статье рассмотрены методы освещения трехмерных объектов, позволяющие существенно повысить реалистичность виртуальных трехмерных сцен, приведены их достоинства и недостатки, а также предложены подходы к организации данных об источниках света трехмерной сцены в памяти видеокарты.

Зачастую для корректного освещения трехмерной сцены требуется значительное количество источников света. При традиционном (forward rendering) подходе расчет освещения производится следующим образом: для каждой вершины в трехмерной сцене и каждого пикселя конечного изображения, выводимого на экран, последовательно производится расчет влияния каждого источника света в сцене, то есть для каждого пикселя выполняются расчет освещения и геометрическая обработка. Соответственно, при большом количестве источников света производительность подсистемы визуализации может быть существенно снижена.

Одним из решений данной проблемы является применение методов так называемой отложенной визуализации, в частности, отложенного освещения. Основная идея заключается в отделении этапа геометрической обработки от этапа освещения трехмерной сцены. Прорисовка изображения осуществляется в несколько этапов. Вся геометрия сцены прорисовывается один раз, при этом сохраняется информация о цвете, нормалях и глубине прорисовки для каждого пикселя в промежуточный G-Buffer, который используется на последующих этапах прорисовки. Однако прорисовка трехмерных сцен при помощи отложенной визуализации также имеет ряд ограничений.

В статье описаны несколько методов устранения некоторых недостатков отложенной визуализации с сохранением ее преимуществ. Это так называемые тайловая и кластерная визуализации. Приводятся данные измерений производительности системы визуализации, использующей различные методы освещения.

Ключевые слова: *тренажерно-обучающая система, подсистема визуализации, тренажер, рендеринг, отложенная визуализация, освещение, кластерная визуализация, тайловая визуализация.*

Тренажерно-обучающая система оператора сложной технической системы (далее ТОС) – техническое средство для исследований и подготовки операторов сложных технических систем, отвечающее требованиям методик подготовки, обеспечивающее получение знаний, навыков и умений, реализующее модель таких систем и осуществляющее контроль над действиями обучаемого. ТОС могут применяться для формирования индивидуальных профессиональных навыков и умений, а также для отработки групповых операций [1, 2].

Подсистема визуализации ТОС обеспечивает отображение результатов моделирования внешней среды и объекта управления с помощью устройств отображения информации. Воспроизведение визуальной картины должно быть с достаточно подробным содержанием, позволяющим операторам ТОС успешно выполнять поставленные задачи [3]. Важную роль в достижении реалистичности отображаемой сцены играет освещение.

Для корректного освещения трехмерной сцены, как правило, требуется значительное количество источников света. При традиционном (forward rendering) подходе расчет освещения производится следующим образом: для каждой вершины в трехмерной сцене и каждого пикселя конечного изображения, выводимого на экран, последовательно производится расчет влияния каждого источника света в сцене, то есть этапы расчета освещения и геометрической обработки выполняются для каждого пикселя. Кроме того, расчеты производятся даже для скрытых и перекрывающихся поверхностей, а также для тех пикселей, которые могут не попасть в итоговое изображение. Соответственно, при большом количестве источников света производительность подсистемы визуализации может быть существенно снижена.

Одним из решений данной проблемы является применение методов так называемой отложенной визуализации, например, отложенного освещения, основная идея которого в отделении этапа геометрической обработки от этапа освещения трехмерной сцены. Прорисовка изображения осуществляется в четыре этапа [4]: геометрическая обработка, освещение, постобработка и заключительный этап. Вся геометрия сцены прорисовывается один раз, при этом информация о цвете, нормалях и глубине прорисовки для каждого пикселя сохраняется в промежуточный G-Buffer, который используется на последующих этапах прорисовки [5]. Далее с использованием сохраненной в G-Buffer информации производится расчет освещения. Затем итоговое изображение копируется в кадровый буфер.

Однако прорисовка трехмерных сцен при помощи отложенной визуализации имеет множество ограничений. Во-первых, существуют значительные проблемы с отображением полупрозрачных объектов из-за того, что в G-Buffer сохраняется информация только о ближайшем к виртуальной камере трехмерном объекте. Во-вторых, из-за слож-

ности реализации ограничено применение аппаратного сглаживания изображения (Multisample Anti-aliasing – MSAA). В-третьих, отложенная визуализация характеризуется значительным расходом видеопамяти, который увеличивается при повышении разрешения итогового изображения, а также значительным объемом данных, передаваемых по шине данных при прорисовке каждого кадра.

В настоящее время существует несколько методов устранения некоторых недостатков отложенной визуализации с сохранением ее преимуществ. Одним из них является тайловая (плиточная) визуализация, которая может применяться как с традиционным подходом к освещению трехмерной сцены, так и с отложенным [6–8]. Отложенная визуализация с помощью тайлов позволяет уменьшить объем передаваемых данных по шине, но увеличивает общий объем вычислений, связанных с освещением. С учетом мощностей современных графических адаптеров применение отложенной визуализации с помощью тайлов обеспечивает значительное увеличение производительности системы визуализации по сравнению с «чистой» отложенной визуализацией.

Принцип тайловой визуализации состоит в группировании пикселей итогового изображения, которое будет отображено на экране, в прямоугольные участки (тайлы), а также в использовании значения глубины каждого пикселя для определения дополнительных зон видимости – усеченных пирамид (view frustum). Соответственно, тайлы, в которых находятся пиксели со схожими значениями глубины, например, в данных пикселях отображается один и тот же трехмерный объект, определяют достаточно небольшой объем трехмерного пространства. В то же время для тайлов, пиксели которых существенно отличаются значениями глубины, дальняя граница ограничивающего объема тайла должна совпадать с максимальным значением глубины его пикселей. Это снижает эффективность отсечения обрабатываемых источников света и приводит к прямой зависимости производительности от выводимого изображения. Данный аспект особенно нежелателен в приложениях, работающих в режиме реального времени (не менее 25 кадров в секунду), так как в этом случае практически невозможно гарантировать стабильность частоты смены кадров системы визуализации.

Еще одним методом является так называемая кластерная визуализация, которая решает проблемы тайловой визуализации. Каждый кластер имеет фиксированную глубину, что снижает зависимость производительности от отображаемого изображения. Применение кластерного рендеринга позволяет использовать более миллиона источников света в виртуальной трехмерной сцене в реальном масштабе времени [9].

По аналогии с тайловой визуализацией кластерная визуализация может быть задействована как с традиционным (forward) расчетом освещения, так и с отложенным (deferred). Основные этапы кластерной визуализации:

– прорисовка трехмерной сцены в G-Buffer (в случае использования кластерной отложенной визуализации);

– создание массива кластеров на основе информации о размерах трехмерной сцены или усеченной пирамиды видимости;

– поиск уникальных кластеров, задействованных в данном кадре прорисовки;

– сохранение информации об источниках света в найденных уникальных кластерах.

Одной из основных задач кластерной визуализации является создание массива кластеров, то есть разбиение трехмерного пространства на определенное количество сегментов. Необходимо, чтобы кластеры были небольшого размера для охвата как можно меньшего количества источников света, в то же время они должны быть достаточного размера для сохранения требуемого уровня производительности при обработке кластеров. Одним из методов создания массива кластеров является разбиение виртуального трехмерного пространства на равные промежутки – создание сетки. Этот метод позволяет быстро определять уникальный ключ каждого кластера, а также создавать кластеры одинакового размера. Однако данный подход требует ручной настройки для каждой отдельной трехмерной сцены, к тому же уникальный ключ каждого кластера может занимать значительное число битов в зависимости от размера сцены. Кроме того, из-за проецирования сетки кластеров на экран дальние кластеры становятся маленькими на итоговом изображении, вплоть до размера одного пикселя, что снижает общую производительность.

Еще один метод – создание массива кластеров на основе информации о видимой части трехмерной сцены. Аналогично тайловой визуализации итоговое изображение разбивается на определенное количество прямоугольников, а затем производится разбиение видимого пространства вглубь трехмерной сцены. Наиболее простым методом разбиения пространства является разделение глубины видимого пространства трехмерной сцены на равные сегменты в нормализованных координатах устройства (на отрезке значений [0, 1]). Однако распределение значений в нормализованных координатах устройства нелинейно, что приводит к неравномерным размерам кластеров: у ближней границы прорисовки виртуальной камеры кластеры становятся очень тонкими, в то время как у дальней границы очень длинными.

Чтобы кластеры наиболее соответствовали форме куба, применяется экспоненциальное разделение пространства по следующей формуле: $near_k = near(1+\frac{2\tan\theta}{S_y})^k$, где $near$ – ближняя граница зоны видимости виртуальной камеры; S_y – вы-

сота ячеек кластера по оси *y*; 2θ – угол обзора виртуальной камеры; *k* – итерация разделения видимого виртуального пространства.

Количество делений видимого пространства должно быть ограничено во избежание существенных потерь производительности при обработке. Зачастую видимое пространство разделяют на 16 частей вглубь, а размер тайлов по высоте и ширине составляет от 32 до 64 пикселей.

После разделения пространства на кластеры необходимо определить список используемых кластеров. Одним из способов является передача информации обо всех кластерах в шейдерные программы. Преимущества такого подхода в простоте реализации и фиксированных размерах передаваемых данных, недостаток – в значительном расходе памяти на хранение информации об источниках света в каждом кластере. Например, для разрешения FullHD (1 920×1 080) и с учетом возможности сохранения информации о 256 источниках света в каждом кластере объем занимаемой информации составит приблизительно 4 Мб. С учетом двойной буферизации (в случае заполнения списка на центральном процессоре) объем составит 8 Мб.

Другим способом является определение наибольшего возможного количества одновременно используемых кластеров. Практически ни в одной трехмерной сцене источники света не охватывают все кластеры. Следует определить количество одновременно используемых кластеров в трехмерной сцене с максимальным количеством источников света и, например, использовать для записи информации о кластерах буфер данных, размер которого в два раза превышает тот, который ранее определен по числу кластеров.

На следующем этапе определяется количество источников света в каждом кластере. Основная задача – выявление пересечений ограничивающего контура источника света с кластерами. Для каждого типа источника света (наиболее часто применяются направленные источники света в виде конуса, и всенаправленные источники света в виде сферы) используется свой вид определения пересечений. Для всенаправленного источника света часто применяется ограничивающая сфера, что затрудняет определение пересечений с кластерами: большинство алгоритмов отсечения работают тогда, когда сфера значительно меньше проверяемого ограничивающего объема, в то время как в случае с кластерами ограничивающая сфера источника света зачастую в разы больше, чем кластер. Один из способов решения проблемы – использование ограничивающих кубов (Axis Aligned Bounding Box – AABB) для всенаправленных источников света и их сравнение с ограничивающим пространством каждого кластера.

Для направленных источников света определение пересечений при помощи ограничивающего куба дает значительное количество ложных срабатываний, когда множество кластеров, которые на самом деле не пересекаются обрабатываемым источником света, все равно задействованы в дальнейших расчетах освещения. Одним из решений данной проблемы является создание усеченной пирамиды (frustum) для обрабатываемого источника света, посредством которой будут определяться пересечения с кластерами [10].

Информацию об используемых в обрабатываемом кадре системы визуализации кластерах и источниках света следует передать в память видеокарты для использования в шейдерных подпрограммах. Одним из способов является хранение информации о кластерах и источниках света в различных текстурах. В шейдерной подпрограмме определение конкретного кластера производится при помощи координат текущего обрабатываемого пикселя в итоговом изображении и значения глубины обрабатываемого пикселя. Информация об используемых кластерах может храниться в 3D-текстуре, в каждом текселе которой сохраняются информация о количестве направленных и всенаправленных источников в данном кластере, а также ссылка на данные об источниках света. Если же информация о различных источниках света занимает различное количество байтов, может быть создана промежуточная текстура с индексами элементов массива с источниками света. Тогда в трехмерной текстуре кластеров будет храниться информация об используемом индексе в промежуточной текстуре. В зависимости от количества одновременно используемых источников света объем передаваемых на видеокарту данных для каждого кадра подсистемы визуализации может составлять 50–100 Кб.

По производительности визуализация с использованием кластерного разбиения пространства и освещения в среднем в два раза превосходит визуализацию с использованием отложенного освещения: на тестовой сцене с использованием 1 024 источников света частота смены кадров системы визуализации с отложенной визуализацией составляла 96 кадров в секунду, для кластерной визуализации – 216 кадров в секунду. Кластерная визуализация с традиционным подходом к освещению (clustered forward) приблизительно на 20 % производительнее тайловой визуализации с традиционным подходом и примерно на 25 % медленнее тайловой визуализации с отложенным расчетом освещения. Измерения производились на оборудовании Intel Core i7-950, Nvidia Geforce 470, на тестовой сцене Crytek Sponza.

В настоящее время визуализация с использованием кластерного подхода к освещению начинает активно использоваться в мультимедийных проектах. Кластерная визуализация не ограничена пропускной способностью шины данных, что характерно для отложенной визуализации. Кроме того, в отличие от тайловой визуализации у кластерной

нет прямой зависимости между производительностью и отображаемым изображением. Возможность использования как традиционного расчета освещения, так и отложенного вместе с кластерным разбиением пространства позволяет использовать данный метод визуализации для различных проблемно-ориентированных отраслей, где требуется реалистичная визуализация трехмерных объектов в реальном масштабе времени.

Работа выполняется в рамках проекта № 2.9 программы фундаментальных исследований ОНИТ РАН.

Литература

1. Гиацинтов А.М., Мамросенко К.А., Решетников В.Н. Инструментальные средства предтренажерной и тренажерной подготовки операторов сложных технических систем // Программные продукты, системы и алгоритмы. 2014. № 1. URL: http://swsys-ru/simulator-training-operators.html (дата обращения: 14.09.2016).

2. Решетников В.Н. Космические телекоммуникации (начала). М.: Изд-во МАТИ, 2013. 184 с.

3. Мамросенко К.А., Решетников В.Н. Моделирование подстилающей поверхности в имитационных системах // Программные продукты и системы. 2015. № 4. С. 70–74.

4. Alamia M. Tutorial – simple OpenGL deferred rendering. Coding Labs . URL: http://www.codinglabs.net/tutorial_simple_def_rendering.aspx (дата обращения: 08.07.2016).

5. Боресков А. Steps3d – Tutorials – Deferred Shading. URL: http://steps3d.narod.ru/tutorials/ds-tutorial.html (дата обращения: 08.07.2016).

6. Olsson O., Assarsson U. Tiled shading. J. Graph. GPU Game Tools, 2011, vol. 15, no. 4, pp. 235–251.

7. Trebilco D. Light-indexed deferred rendering. Shader X7, 2009, pp. 243–256.

8. Harada T. Forward+: Bringing deferred lighting to the next level. URL: https://takahiroharada.files.wordpress.com/2015/04/forward_plus.pdf (дата обращения: 08.07.2016).

9. Olsson O., Billeter M., Assarsson U. Clustered deferred and forward shading. Hight Perform. Graph. 2012. С. 1–10.

10. Persson E. Practical clustered shading. URL: www.humus.name/Articles/PracticalClusteredShading.pdf (дата обращения: 08.07.2016).

Software & Systems
DOI: 10.15827/0236-235X.116.069-072

Received 15.09.16
2016, vol. 29, no. 4, pp. 69–72

3D OBJECTS RENDERING USING CLUSTERED SHADING

A.M. Giatsintov[1]*, Researcher, algts@inbox.ru*
K.A. Mamrosenko[1]*, Ph.D. (Engineering), Head of the Research Center, kirillam@ya.ru*

[1] *Center of Visualization and Satellite Information Technologies SRISA, Nakhimovsky Ave. 36/1, Moscow, 117218, Russian Federation*

Abstract. The article describes methods of lighting and shading of 3D objects that significantly enhance the realism of three-dimensional virtual scenes, their advantages and disadvantages. It also presents approaches for organization of light sources data in graphics card memory.

Usually a large number of light sources are required in order to shade the 3D scene correctly. With forward rendering lighting is computed in the following way: influence of each light source on the resulting image is computed for each vertex in 3D scene and each pixel of the framebuffer. Therefore, visualization subsystem performance may be significantly reduced when there are a lot of light sources in a scene.

In order to solve this problem we can use the methods of deferred rendering. The main idea of deferred rendering is to separate the geometry processing phase from lighting phase. An image is rendered in several phases. Scene geometry is rendered only once and information about color, normal and depth of each pixel is stored in temporary G-Buffer that is used in the following phases of rendering. But deferred rendering also has a number of significant downsides.

This article provides information about some methods that resolve most of drawbacks of deferred rendering while retaining its advantages. They include tiled visualization and clustered visualization. The paper provides performance measurements of visualization subsystem with various rendering methods.

Keywords: training simulation system, visualization subsystem, trainer, rendering, deferred shading, lighting, clustered rendering, tile rendering.

Acknowledgements. *This work is done within the ONIT fundamental research program project no. 2.9.*

References

1. Giatsintov A.M., Mamrosenko K.A., Reshetnikov V.N. A toolkit for pre-training and training operators of complec technical systems. *Programmnye produkty, sistemy i algoritmy* [Software, Systems and Algorithms]. 2014, no. 1. Available at: http://swsys-ru/simulator-training-operators.html (accessed September 14, 2016) (in Russ).

2. Reshetnikov V.N. *Kosmicheskie telekommunikatsii (nachala)* [Space telecommunications (beginning)]. Moscow, ITTs MATI Publ., 2013, 2nd ed., 184 p. (in Russ).

3. Mamrosenko K.A., Reshetnikov V.N. Terrain modelling in simulation training systems. *Programmnye produkty i sistemy* [Software & Systems]. 2015, no. 4, pp. 70–74 (in Russ).

4. Boreskov A. *Steps3d – Tutorials – Deferred Shading*. Available at: http://steps3d.narod.ru/tutorials/ds-tutorial.html (accessed July 8, 2016) (in Russ).

5. Alamia M. *Coding Labs. Simple OpenGL deferred rendering tutorial*. Available at: http://www.codinglabs.net/tutorial_simple_def_rendering.aspx (accessed July 8, 2016).

6. Olsson O., Assarsson U. Tiled Shading *J. Graph. GPU Game Tools*. 2011, vol. 15, no. 4, pp. 235–251.

7. Trebilco D. Light-indexed deferred rendering. *Shader X7*. 2009, pp. 243–256.

8. Harada T. *Forward+: Bringing deferred lighting to the next level.* Available at: https://takahiroharada.files.wordpress.com/2015/04/forward_plus.pdf (accessed July 8, 2016).

9. Olsson O., Billeter M., Assarsson U. Clustered Deferred and Forward Shading. *High Perform. Graph*. 2012, pp. 1–10.

10. Persson E. *Practical clustered shading*. Available at: www.humus.name/Articles/PracticalClusteredShading.pdf (accessed July 8, 2016).