УДК 004.925

**А. М. Гиацинтов,** науч. сотр., e-mail: algts@inbox.ru,
**К. А. Мамросенко,** канд. техн. наук, зав. отделом, e-mail: kirillam@ya.ru
Центр визуализации и спутниковых информационных технологий,
Научно-исследовательский институт системных исследований РАН

# Методы отображения трехмерных объектов при применении отложенной визуализации

*Описан метод отложенной визуализации, применяемый на современных аппаратных платформах. Определены этапы формирования изображения, необходимые при отложенной визуализации. Рассмотрен метод устранения артефактов итогового изображения (сглаживания), основанный на поиске граней и реализуемый в качестве этапа постобработки. Представлен разработанный метод отображения полупрозрачных объектов при применении отложенной визуализации.*

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

## Введение

Тренажерно-обучающая система (далее ТОС) оператора сложной технической системы — техническое средство для 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, а также используемое для исследований. ТОС могут применять для формирования индивидуальных профессиональных навыков и умений, а также для отработки групповых операций [1].

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

Зачастую для освещения трехмерной сцены требуется несколько источников света. При традиционном подходе (*forward rendering*) этапы прорисовки геометрических объектов и их освещения совмещены. Упрощенно прорисовка объектов осуществляется следующим образом: информация о геометрических объектах (вершинах, нормалях, текстурных координатах) передается сначала в вершинный шейдер, затем, при необходимости, в геометрический шейдер. Далее происходит растеризация геометрических объектов. Полученные при растеризации данные обрабатываются пиксельными шейдерами и выводятся в кадровый буфер. Схема прорисовки геометрии при традиционном подходе представлена на рис. 1. Расчет освещения определяет влияние каждого источника света в сцене на геометрические объекты и проводится для каждой вершины в трехмерной сцене и каждого пикселя конечного изображения, выводимого на экран. Для динамических источников света и динамической геометрии сцены требуются расчеты пересечения источников света и геометрических объектов, например, с помощью ограничивающих сфер, кубов [2]. Кроме того, расчеты будут проведены даже для скрытых и перекрывающихся поверхностей, а также для тех пикселей, которые могут не попасть в итоговое изображение. Соответственно, при большом числе источников света производительность подсистемы визуализации может быть существенно снижена.

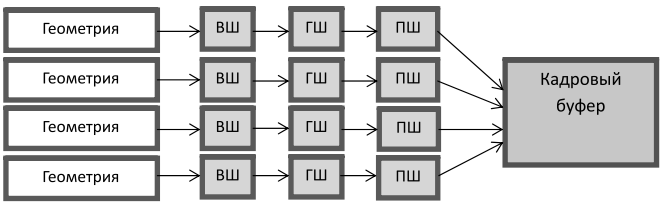


**Рис. 1. Процесс прорисовки изображения с помощью традиционного метода визуализации; ВШ — вершинный шейдер, ГШ — геометрический шейдер, ПШ — пиксельный шейдер**

## Метод отложенной визуализации

Одним из решений данной проблемы является применение методов так называемой отложенной визуализации, в частности, отложенного освещения. Первый метод был разработан Майклом Дирингом и его коллегами в 1988 г. В своей работе [3] авторы предложили систему, в которой процессоры обработки треугольников растеризируют геометрию, а затем шейдерные процессоры применяют затенение по Фонгу к обрабатываемым геометрическим объектам с использованием многочисленных источников света. Существенный вклад в разработку технологии отложенной визуализации внесла ра-

бота Сайто и Такахаши в 1990 г. [4]. В своей статье авторы предлагают способ визуализации 3D-изображений, основанный на распознавании геометрических фигур и их последовательностей. Для оптимизации процесса геометрические свойства поверхностей сохраняются в геометрических буферах (G-Buffers). Применение геометрических буферов для хранения промежуточных результатов позволяет отделить этап распознавания фигур от этапов геометрической обработки (проецирования, удаления невидимых граней, затенения, применения текстур), и использовать его для постобработки.

Рассмотренные методы отложенной визуализации требовали применения узкоспециализированных аппаратных модулей. В настоящее время методы разрабатываются с учетом применения вычислительных мощностей центральных процессоров и графических адаптеров.

Рассмотрим подробнее базовый метод отложенной визуализации, применяемый на современных аппаратных платформах. Его основной идеей является отделение этапа геометрической обработки, во многом схожего с этапом прорисовки геометрии в традиционном подходе, от этапа освещения трехмерной сцены.

Прорисовка изображения осуществляется в несколько этапов. Все геометрические объекты сцены прорисовываются один раз, при этом информация о цвете, нормалях и глубине прорисовки для каждого пикселя сохраняется в промежуточный G-Buffer. Далее, с использованием сохраненной информации рассчитывается освещение. Затем итоговое изображение копируется в кадровый буфер (рис. 2). Всего этапов четыре: этап геометрической обработки, этап освещения, этап постобработки и заключительный этап [5].

Каждый этап использует функционал программируемого графического конвейера, применяя вершинные и пиксельные шейдеры. Все расчеты, проводимые на каждом этапе, сохраняются в памяти видеокарты в виде текстур [6].

Этап геометрической обработки является единственным этапом, который работает с данными трехмерных объектов. Входной информацией для данного этапа является трехмерная сетка обрабатываемого объекта, а выходом — изображение в G-Buffer. Для заполнения G-Buffer информацией буфер необходимо сделать активным. Первой обрабатывается информация о вершинах, текстурных координатах и материалах — она записывается в текстуру глубины G-Buffer в памяти видеокарты. Далее геометрические данные и информация о материалах обрабатываются пиксельными шейдерами. Результат их работы загружается в остальные текстуры буфера — текстуры нормалей, цвета и отражений.

В текстуру глубины записывается информация глубины для каждого пикселя итогового изображения. Значение глубины находится в отрезке 0—1, где 0 — ближняя граница прорисовки виртуальной камеры, 1 — дальняя граница прорисовки. Если один трехмерный объект перекрывает другой, то в текстуру глубины записывается значение глубины, соответствующее ближайшему объекту. Во время этапа освещения возможно восстановить информацию о позиции объекта, зная значение глубины и направление виртуальной камеры.

Далее проводится заполнение текстур нормалей, цвета и отражений. Каждый объект трехмерной сцены прорисовывается с учетом заданного материала, в котором указываются используемые вершинные и пиксельные шейдеры, определяющие алгоритм обработки данного объекта.

После заполнения всех текстур G-Buffer (нормалей, цвета, отражения и глубины) начинается этап освещения. Сначала делается активной текстура цвета, для которой рассчитывается влияние рассеянного света. Затем для каждого источника света в трехмерной сцене определяется ограничивающий прямоугольник, в рамках которого учитывается влияние источника света. Прямоугольник рассчитывается в экранных координатах, его размер равен разрешению экрана. Благодаря этому достигается полное отделение этапа освещения от этапа геометрической обработки, что приводит к значительному повышению производительности при большом числе (более 8) источников света.

При использовании метода отложенной визуализации нет необходимости знать, какие геометрические объекты будут освещены каким источником света, при этом возможна обработка всех источников света, которые освещают видимые наблюдателем объекты. Для уменьшения числа обрабатываемых источников света могут быть применены различные методы оптимизации, например, могут быть исключены из обработки источники света, которые закрыты другими объектами.

Как и при визуализации традиционным способом, существует несколько вариантов расчета

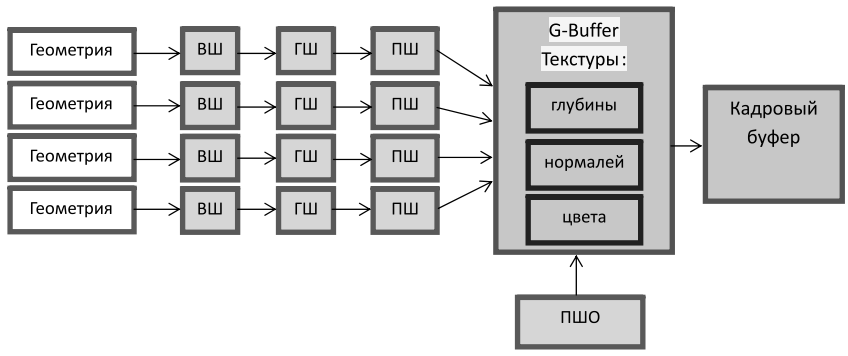


**Рис. 2. Процесс прорисовки изображения с помощью отложенного метода визуализации; ВШ — вершинный шейдер, ГШ — геометрический шейдер, ПШ — пиксельный шейдер без процедур расчета освещения, ПШО — пиксельный шейдер, рассчитывающий освещение пикселя**

влияния источников света на трехмерную сцену, например, последовательная обработка каждого источника света (более простой вариант в реализации, но более затратный по производительности) или же обработка сразу множества источников света за один проход. Рассмотрим процесс последовательной обработки источников света. Описанные ниже действия также могут быть использованы при создании более сложных шейдеров, реализующих обработку нескольких источников света за один проход.

Первым шагом обработки является активация созданных ранее текстур глубины, цвета, нормалей и отражений и передача параметров об источнике света (позиция, цвет, радиус) в пиксельный шейдер. Далее необходимо рассчитать влияние рассеянного света на пиксели итогового изображения (*ambient pass*). Те пиксели, на которые не оказывают влияния источники света, будут прорисованы только с учетом рассеянного освещения, для остальных пикселей рассчитывается рассеянное освещение с учетом влияния источников света. Достигается это путем прорисовки полноэкранного прямоугольника. По сути, данный этап осуществляет "заливку" прорисовываемых трехмерных объектов выбранным цветом. Если у объектов текстуры присутствуют, то происходит смешивание цветов.

При последовательной обработке источников света для каждого источника будет прорисован полноэкранный прямоугольник, определяющий его влияние на трехмерные объекты. Возможно ввести ограничение размера прорисовываемого прямоугольника. Так как при прорисовке полноэкранного прямоугольника каждый пиксель итогового изображения будет обработан шейдером освещения, то ограничение его размера, т.е. отбрасывание всех пикселей, на которые обрабатываемый источник света не оказывает влияния, позволит существенно сократить вычислительную нагрузку.

Например, имеется трехмерная сцена с тремя источниками света. Если размер итогового изображения составляет 512×512 (262 144) пикселей, то без оптимизации для обработки трех источников света шейдер обработки будет вызван 786 432 раза. Применение оптимизации позволяет сократить число обрабатываемых пикселей для каждого источника света приблизительно на 50 % в зависимости от расположения источников света и их радиуса.

Расчет ограничивающего прямоугольника выполняется на центральном процессоре перед прорисовкой каждого источника света. Одним из возможных методов расчета прямоугольника является проецирование восьми вершин ограничивающего куба источника света (AABB — Axis-Aligned Bounding Box) на экранную плоскость и нахождение ограничивающего 2D-прямоугольника из этих точек.

Следующим шагом расчета освещения является расчет влияния источников света (*illumination pass*). Для этого при обработке каждого источника света прорисовывается прямоугольник с размером, совпадающим с размером итогового изображения. Обработка начинается с поиска позиции текущего пикселя в видовой системе координат. Значения ближней и дальней границ прорисовки, вектор направления виртуальной камеры и сохраненное нормализованное значение глубины для конкретного пикселя передаются в шейдерную программу, написанную на языке GLSL (OpenGL Shading Language) для расчета положения пикселя. Чтобы вектор направления виртуальной камеры был доступен в шейдере для каждого пикселя изображения, его необходимо передавать как текстурные координаты для каждой из четырех вершин прорисовываемого прямоугольника. Вектор будет линейно интерполирован при обработке пикселей прорисовываемого прямоугольника. Для компенсации ошибок линейной интерполяции на этапе растеризации в шейдере необходимо проводить нормализацию вектора.

Первым этапом расчета положения пикселя является определение компоненты z из нормализованного значения глубины, сохраненного в текстуре глубины. Для этого в шейдере выполняются следующие расчеты:

$$\text{float pos.z} = \text{far\_plane}*\text{near\_plane}/(d*(\text{far\_plane} - \text{near\_plane}) - \text{far\_plane}),$$

где pos.z — значение компоненты z пикселя в системе координат виртуальной камеры; near_plane — значение ближней границы прорисовки; far_plane — значение дальней границы прорисовки; d — значение глубины для текущего пикселя.

После расчета координаты z выполняется расчет X и Y координат позиции пикселя:

$$\text{pos.xy} = \text{view.xy} / \text{view.z} * \text{pos.z},$$

где view.xy, view.z — компоненты вектора направления виртуальной камеры, pos.xy, pos.z — позиции пикселя в координатах виртуальной камеры.

Теперь позиция пикселя получена и после активации текстур цвета, отражений и нормалей возможно применение освещения по Фонгу к пикселю. Цвет пикселя также учитывает освещение от других источников света в трехмерной сцене путем аддитивного смешивания.

После окончания этапа освещения имеется готовое изображение, которое можно отобразить на экране. Но при необходимости также можно применить к изображению такие эффекты постобработки, как сглаживание (*antialiasing*) и засветка (*bloom*).

Так как одним из недостатков отложенной визуализации является отсутствие поддержки аппаратного сглаживания изображения (MSAA — Multisample Anti-Aliasing) при прорисовке в основной кадровый буфер, то сглаживание реализуется в качестве этапа постобработки (фильтра). Одним из вариантов реализации сглаживания как этапа постобработки является поиск граней в изображении. Для этого используется шейдер, который будет при-

менять размытие к пикселям итогового изображения, если соответствующие значения нормали и/или глубины скачкообразно изменяются. Фильтр сглаживания складывает квадраты расстояний от текущего пикселя до соседних пикселей для расчета коэффициента размытия пикселей. Осуществляется обработка восьми соседних пикселей вокруг целевого пикселя для поиска скачкообразных изменений по всем направлениям.

Формула расчета коэффициента размытия пикселя:

$$K(u, v) = \sum_{i=1}^{8} (p_{(u + \text{сдвиг}[i].\, x)(v + \text{сдвиг}[i]).\, y)} - p_{uv})^2, \quad (1)$$

где $p_{uv}$ — целевой пиксель с координатами $(u, v)$, сдвиг $[i]$ — таблица сдвигов для изменения позиций соседних пикселей вокруг целевого пикселя.

После определения коэффициента размытия пикселя (с помощью текстур нормалей и/или глубины) рассчитывается окончательный цвет пикселя. Так, в тех участках изображения, где грани не были обнаружены, сдвиг будет равен нулю, т.е. размытие проводиться не будет. В случае обнаружения граней к цвету пикселя добавляются цвета соседних пикселей в соответствии с рассчитанным коэффициентом размытия.

### Отображение полупрозрачных объектов при применении отложенной визуализации

Рассмотренный метод отложенной визуализации не предусматривает наличия в сцене полупрозрачных объектов. Это обусловлено тем, что при прорисовке геометрии применяется так называемый тест видимости для исключения объектов, которые перекрываются другими объектами, для повышения скорости визуализации. Однако из-за аппаратных ограничений графических адаптеров для каждого пикселя за один проход теста видимости может быть сохранена только информация о ближайшем объекте, что усложняет процесс визуализации — для правильной прорисовки полупрозрачных объектов требуется информация о видимости каждого объекта, перекрываемого полупрозрачным объектом. Из существующих способов расчета видимости пикселей наиболее часто применяются зависимый и независимый от порядка прорисовки способы. Для реализации второго способа требуется, чтобы все прорисовываемые объекты были отсортированы от дальнего к ближнему, что позволяет получить результат без искажений [7].

Одним из методов прорисовки полупрозрачных объектов при отложенной визуализации является метод создания слоев прозрачности (*depth peeling*) [8]. Он позволяет отображать полупрозрачные объекты независимо от порядка прорисовки, последовательно обрабатывая слои полупрозрачных пикселей с помощью теста глубины [9].

По сравнению с другими методами прорисовки полупрозрачных объектов, например, отложенной визуализации с разделением изображения на секции (*tiled deferred rendering*) *depth peeling* не требует применения вычислительных шейдеров (*compute shader*) для своей работы. Соответственно, метод отложенной визуализации с использованием *depth peeling* может быть реализован на графических картах уровня DirectX 10 (OpenGL 3.3). По сравнению с методом отложенной визуализации с индексированием источников света (Light Indexed Deferred Rendering) *depth peeling* не требует предварительной сортировки источников света и не имеет проблем с пересекающимися источниками света. Однако метод *depth peeling* может требовать большего количества видеопамяти для хранения промежуточных результатов визуализации.

Разработанный метод прорисовки полупрозрачных поверхностей основывается на описанном выше методе отложенной визуализации.

Во время инициализации программируемого конвейера вместе с основным G-Buffer создается так называемый буфер прозрачности, который идентичен G-Buffer. Так же, как и в базовом алгоритме отложенной визуализации, расчет итогового изображения разделен на несколько этапов: прорисовка геометрических объектов, освещение, прорисовка прозрачных объектов, вывод на экран.

На этапе геометрической обработки в G-Buffer помещаются все данные о геометрии трехмерной сцены, за исключением данных о полупрозрачных объектах. Далее на этапе освещения выполняются переключение на буфер прозрачности, а также очистка его текстур глубины и цвета. Текстура глубины из G-Buffer копируется в текстуру глубины буфера прозрачности шейдером копирования данных. Для этого требуется прорисовка полноэкранного прямоугольника поверх буфера отрисовки. Затем происходит отсоединение текстуры глубины и присоединение текстур цвета, нормалей, отражений из G-Buffer. Присоединенные текстуры прорисовываются в соответствующие текстуры буфера прозрачности шейдером обработки за счет прорисовки полноэкранного прямоугольника. После этого осуществляются освещение прорисованных объектов и отсоединение всех буферов.

Следующим этапом является прорисовка и освещение полупрозрачных объектов.

Последним этапом является перемещение информации из буфера прозрачности в кадровый буфер. Проводится переключение на кадровый буфер и его очистка. Далее цветовая текстура буфера прозрачности копируется в кадровый буфер с помощью шейдера копирования. После этого изображение в кадровом буфере считается завершенным и может быть отображено на экране.

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

## Заключение

Использование методов отложенной визуализации позволяет сократить объем вычислений при наличии более восьми источников света по сравнению с традиционными методами визуализации за счет исключения из обработки всех пикселей, которые не попадут в итоговое изображение. Одну из основных проблем отложенной визуализации — отсутствие возможности обработки полупрозрачных объектов, решает разработанный авторами метод. Преимуществами метода являются применимость вне зависимости от порядка прорисовки объектов и относительная простота реализации. Недостатком является значительный объем используемой видеопамяти, который увеличивается при повышении разрешения итогового изображения. Несмотря на указанный недостаток разработанный метод применим для визуализации высокореалистичных трехмерных сцен в тренажерно-обучающих системах.

**Список литературы**


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-web.ru/simulator-training-operators.html.
2. **Решетников В. Н.** Тригонометрические воспоминания. Серия Космические телекоммуникации. Санкт-Петербург: ООО "РИП СПБ", 2015. 138 с.
3. **Deering M., Schediwy B., Duffy C**. The triangle processor and normal vector shader: a VLSI system for high performance graphics // SIGGRAPH'88 Proceedings of the 15th annual conference on Computer graphics and interactive techniques., 1988. P. 21—30.
4. **Saito T., Takahashi T.** Comprehensible Rendering of 3-D Shapes // Computer Graphics. 1990. N. 4. P. 197—206.
5. **Alamia M.** Coding Labs. Simple OpenGL Deferred Rendering Tutorial [Электронный ресурс]. URL: http://www.codinglabs.net/tutorial_simple_def_rendering.aspx (дата обращения: 08.07.2015).
6. **Fonseca F**. Deferred Shading Tutorial [Электронный ресурс]. URL: http://gamedevs.org/uploads/deferred-shading-tutorial.pdf.
7. **Боресков А**. steps3D — Tutorials — Deferred Shading. [Электронный ресурс]. URL: http://steps3d.narod.ru/tutorials/ds-tutorial.html (дата обращения: 08.07.2015).
8. **Persson E**. Deep deferred shading [Электронный ресурс]. URL: http://www.humus.name/index.php?page = 3D&ID = 75 (дата обращения: 08.07.2015).
9. **Everitt C**. Interactive Order-Independent Transparency [Электронный ресурс]. URL: http://www.eng.utah.edu/~cs5610/handouts/order_independent_transparency.pdf (дата обращения: 18.02.2016).



**A. M. Giatsintov,** Researcher, e-mail: algts@inbox.ru,
**K. A. Mamrosenko,** Head of Department, e-mail: kirillam@ya.ru,
The center of visualization and satellite information technologies, SRISA RAS


# Methods of 3D Objects Visualization with Deferred Rendering


*Article depicts a method of deferred rendering utilized on modern hardware platforms. The use of deferred rendering methods can significantly reduce the amount of required calculations when there are more than eight lights in 3d scene. In contrast with traditional forward rendering methods, deferred rendering also discards pixels that would not complement to the final image. The phases of image drawing, required with the use of deferred rendering, are determined. A method of antialiasing, based on edge detection and implemented as a post-process effect, is described. Article presents a developed method of visualizing semi-transparent 3d objects with deferred rendering.*



***Keywords:*** *training simulation system, visualization subsystem, rendering, trainer, lighting, transparency, deferred rendering*


**References**


1. **Giatsintov A. M., Mamrosenko K. A., Reshetnikov V. N.** Instrumental'nyye sredstva predtrenazhernoy i trenazhernoy podgotovki operatorov slozhnykh tekhnicheskikh sistem, *Programmnye produkty, sistemy i algoritmy*, 2014, no. 1 [Web resource]. URL: http://swsys-web.ru/simulator-training-operators.html.
2. **Reshetnikov V. N.** *Trigonometricheskie vospominaniya. Kosmicheskie telekommunikatsii.* SPb: RIP SPB, 2015. 138 p.
3. **Deering M., Schediwy B., Duffy C**. The triangle processor and normal vector shader: a VLSI system for high performance graphics, *Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, New York, USA, 1988, pp. 21—30.
4. **Saito T., Takahashi T.** Comprehensible Rendering of 3-D Shapes, *Computer Graphics*, 1990, no. 4, pp. 197—206.
5. **Alamia M.** *Coding Labs. Simple OpenGL Deferred Rendering Tutorial [Web resource].* URL: http://www.codinglabs.net/tutorial_simple_def_rendering.aspx
6. **Fonseca F.** *Deferred Shading Tutorial.* [Web resource]. URL: http://gamedevs.org/uploads/deferred-shading-tutorial.pdf.
7. **Boreskov A.** *Deferred Shading.* [Web resource]. URL: http://steps3d.narod.ru/tutorials/ds-tutorial.html
8. **Persson E.** *Deep deferred shading.* [Web resource]. URL: http://www.humus.name/index.php?page = 3D&ID = 75
9. **Everitt C**. *Interactive Order-Independent Transparency* [Web resource]. URL: http://www.eng.utah.edu/~cs5610/handouts/order_independent_transparency.pdf