Методы распределенной имитации непрямого затенения виртуальной среды на GPU в реальном времени*

А. В. Мальцев

Федеральное государственное учреждение "Федеральный научный центр Научно-исследовательский институт системных исследований Российской академии наук", г. Москва, Россия

Аннотация. В работе предлагаются методы и алгоритмы для распределенной имитации непрямого затенения объектов трехмерных виртуальных сцен, освещаемых рассеянным светом, на базе технологии ambient occlusion. Применение высокопроизводительных параллельных вычислений на современных графических процессорах обеспечивает выполнение визуализации виртуальной среды с моделированием таких затенений в масштабе реального времени.

Ключевые слова: виртуальный объект, визуализация, рассеянное освещение, затенение, графический процессор, шейдер, реальное время.

DOI 10.14357/20718632190302

Введение

На сегодняшний день тенденции развития многих научных и технических областей тесно связаны с внедрением технологий трехмерного компьютерного моделирования и виртуальной реальности. Реальные предметы и явления заменяются синтезируемыми с помощью вычислительных устройств виртуальными моделями. Последние имеют более обширные возможности по реализации и требуют меньших финансовых затрат на создание и обслуживание. Такой подход, в том числе, активно используется в имитационно-тренажерных комплексах и системах виртуального окружения. Основными требованиями, предъявляемыми к моделируемой виртуальной среде в сфере тренажерных систем, являются реалистичность визуализируемых изображений и поддержка рендеринга в масштабе реального времени.

Одним из ключевых факторов для достижения реализма, помимо детализации моделей и их корректной динамики, является вычисление освещенности трехмерной сцены. Проблема в данном случае заключается в соблюдении баланса между точностью расчета и временем, необходимым на его выполнение. Так, высокореалистичная визуализация методами глобального освещения (например, с помощью технологии фотонных карт [1, 2], Radiosity [3, 4], многоуровневой трассировки лучей и т.д.) в настоящее время не позволяет обеспечить приемлемое в тренажерных комплексах время синтеза кадра даже с применением параллельных вычислений на современных многоядерных графических процессорах (GPU). Поэтому актуальным путем развития технологий и методов реалистичного рендеринга является приближенная имитация глобального освещения, позволяющая достичь упомянутого выше баланса точности и времени.

*Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 18-07-00950

Одной из важных составляющих иллюминации трехмерной среды является непрямое освещение и затенение поверхностей, возникающее в результате рассеяния световых лучей при столкновении с объектами среды. Наглядным примером может служить открытое пространство в пасмурную погоду, когда прямые солнечные лучи отсутствуют. Освещение объектов при этом достаточно равномерное, контрастных теней не наблюдается. Однако некоторые затенения на поверхностях все же присутствуют, что позволяет человеку различать их форму и объем. Если в виртуальной среде не применяется глобальная модель расчета, то, как правило, рассеянный свет реализуется с помощью фонового освещения (*ambient light*), интенсивность которого добавляется к вычисленной интенсивности освещения каждой точки каждого объекта. Добавление происходит без учета геометрии самого объекта и сцены в целом, принимается во внимание лишь материал поверхностей. Поэтому при отсутствии прямого освещения источниками света изображения виртуальных объектов и их частей могут сливаться, а ощущение их объема теряется.

В данной работе предлагаются оригинальные распределенные методы и алгоритмы имитации рассеянной освещенности и соответствующего затенения виртуальных объектов в реальном масштабе времени, основанные на технологии *screen space ambient occlusion* [5], а также использовании преимуществ параллельных вычислений на современных многоядерных GPU с поддержкой шейдеров последнего поколения. Особенностями предлагаемых решений является учет текстур нормалей и прозрачности при формировании карты затенения рассеянной освещенности виртуальной сцены, использование полусферического ядра выборки и применение карты затенения в процессе прямого, а не отложенного расчета освещения. Далее рассмотрим эти решения подробнее.

1. Формирование данных о сцене в геометрическом буфере

Предлагаемый подход для имитации затенения виртуальных объектов в условиях рассеянного освещения включает в себя четыре этапа: формирование специального набора данных о видимой наблюдателем области сцены, синтез на его основе карты затенения данной области, фильтрация полученной карты с целью устранения шумов и артефактов, а также непосредственно ее применение при рендеринге изображения сцены.

Для записи информации о видимой части сцены будем использовать так называемый геометрический буфер (G-буфер), который широко применяется при визуализации с отложенным освещением и затенением (*deferred shading*, [6]). G-буфер представляет собой совокупность нескольких целей рендеринга в виде перезаписываемых текстур одинакового размера, как правило, совпадающего с размером текущей области вывода изображения на экране. Каждая из них может иметь собственный формат хранения данных и свое количество цветовых каналов. Для решения нашей задачи достаточно использовать G-буфер, состоящий из двух прямоугольных трехканальных текстур G_{pos} и G_{norm} с внутренним форматом *GL_RGB16F* (каждый канал представлен 16-битным вещественным числом). G_{pos} будет содержать координаты видимых наблюдателем точек объектов виртуальной среды, G_{norm} – координаты нормалей к поверхностям объектов в данных точках. Для эффективного использования текстур G_{pos} и G_{norm} на следующем этапе сохраняемые в них координаты точек и нормалей должны быть посчитаны в одной системе координат (СК). В данном случае удобно использовать видовую СК VCS. Обе текстуры ассоциируются с одним и тем же буфером FBO (*frame buffer object*, [7]), который устанавливается в качестве текущего внекадрового буфера вывода.

Далее выполняется распределенный рендеринг на GPU полигональных объектов виртуальной сцены из положения текущего наблюдателя с помощью специальных вершинного и фрагментного шейдеров. При этом параметры виртуальной камеры соответствуют заданным для этого наблюдателя. Вершинный шейдер рассчитывает параллельно для всех вершин полигонов каждого объекта их координаты V_{ndcs} и V_{vcs} в нормализованном объеме видимости и видовом пространстве, а также координаты соответствующих им нормалей $N_{\text{v,vcs}}$ в СК VCS по формулам:

$$V_{\text{vcs}} = M_{\text{mv}} \cdot V_{\text{ocs}};$$
$$V_{\text{ndcs}} = M_{\text{pr}} \cdot V_{\text{vcs}},$$
$$N_{\text{v,vcs}} = \left(M_{\text{mv}}^{-1}\right)^{T} \cdot N_{\text{v,ocs}},$$

где V_{ocs} и $N_{\text{v,ocs}}$ – координаты вершины и нормали в локальной СК OCS объекта, M_{mv} и M_{pr} – модельно-видовая и проекционная матрицы. Полученные данные передаются на выход вершинного шейдера, после чего интерполируются графическим конвейером для каждого треугольного полигона. В результате интерполяции на вход фрагментного шейдера поступают координаты положений P_{vcs} обрабатываемых фрагментов полигонов и соответствующих этим фрагментам нормалей N_{vcs}, представленные в СК VCS. Если материал объекта не содержит карту прозрачности и/или карту нормалей, то P_{vcs} и N_{vcs} направляются на выход шейдера и посредством подключенного буфера FBO записываются соответственно в текстуры G_{pos} и G_{norm}. Координаты перезаписываемых текселов эквивалентны координатам фрагмента в области вывода изображения сцены. Благодаря выполнению теста глубины фрагментов в текстурах сохраняются данные для точек объектов наиболее близких к наблюдателю.

Карта прозрачности используется для определения степени полупрозрачности участков объекта и представляет собой двумерную текстуру в оттенках серого, в которой белые области (значение 1.0) указывают на полную непрозрачность, а черные (0.0) – на абсолютную прозрачность. Зачастую такие текстуры применяются дизайнерами для формирования сквозных вырезов в поверхностях объектов или отсечения какой-либо их части без непосредственного изменения геометрии. Например, при наложении текстуры листа дерева на прямоугольник, состоящий из двух треугольных полигонов, добавление соответствующей карты прозрачности позволяет «вырезать» из него сложную форму этого листа. Число же полигонов, используемых для геометрической реализации такого же объекта, составляло бы на порядок больше, что в масштабе всей сцены с множеством листьев отрицательно сказалось бы на скорости визуализации. При формировании данных в G-буфере необходимо учитывать вырезы и отсечения такого типа. Поэтому при наличии карты прозрачности фрагментный шейдер считывает из нее и анализирует значение прозрачности для обрабатываемого фрагмента. Если оно меньше некоторого малого числа ε (т.е. имеет место полная прозрачность), обработка фрагмента прекращается с помощью вызова команды *discard*.

Карта нормалей – это текстура, в каждом пикселе которой в цветовом формате закодирован вектор нормали, соответствующий точке поверхности, к которой применяется материал. Данный тип текстуры используется для моделирования микрорельефа поверхности (например, по технологии *bump*). Если материал объекта содержит карту нормалей, фрагментный шейдер считывает из нее координаты нормали N_{ts} в точке объекта, соответствующей рассматриваемому фрагменту. Как правило, эти координаты представлены в касательном базисе TBN и требуют перевода в СК VCS по формуле:

$$N_{\text{vcs}} = \left(M_{\text{mv}}^{-1}\right)^{T} \cdot M_{\text{ts}}^{-1} \cdot N_{\text{ts}},$$

где M_{ts} – матрица перехода от локальной СК OCS объекта к касательному базису TBN, M_{mv} – модельно-видовая матрица. Полученный вектор N_{vcs} подается на выход шейдера вместо интерполированного входящего вектора нормали и далее записывается в текстуру G_{norm}.

На Рис. 1 представлен пример виртуальной сцены, освещенной рассеянным светом и визуализируемой без использования технологии ambient occlusion. Сцена содержит трехмерную текстурированную модель гусеничного робота, размещенную на плоской поверхности. На Рис. 2 и Рис. 3 продемонстрированы текстуры G_{pos} и G_{norm} для этой сцены, полученные с помощью описанного выше подхода. Отметим, что изображение на Рис. 2 имеет явную цветовую сегментацию на четыре квадранта, связанную с характером записанных в текстуре G_{pos} данных – координат точек объектов в СК VCS. Как упоминалось ранее, синтез G_{pos} производится путем рендеринга сцены из положения наблюдателя, т.е. из центра правосторонней СК VCS, проецируемого в центр изображения. Взгляд виртуальной камеры направлен вдоль оси -Z, которая перпендикулярна текстуре. Отсюда имеем следующее распределение значений координат для квадрантов: I – $x > 0$, $y > 0$, $z < 0$; II – $x < 0$, $y > 0$, $z < 0$; III – $x < 0$,

$y < 0$, $z < 0$; IV – $x > 0$, $y < 0$, $z < 0$. При отображении текстуры G_{pos} на экране сохраненные в ней значения координат точек трактуются как интенсивности цветовых каналов R, G и B, при этом отрицательные значения приравниваются к 0. Поэтому точки первого квадранта будут иметь желтый цвет, второго – зеленый, третьего – черный, а четвертого – красный.

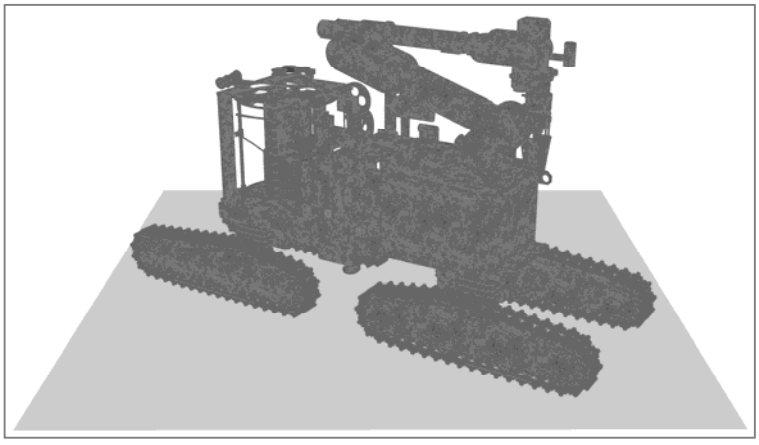

Рис. 1. Виртуальная сцена с рассеянным освещением

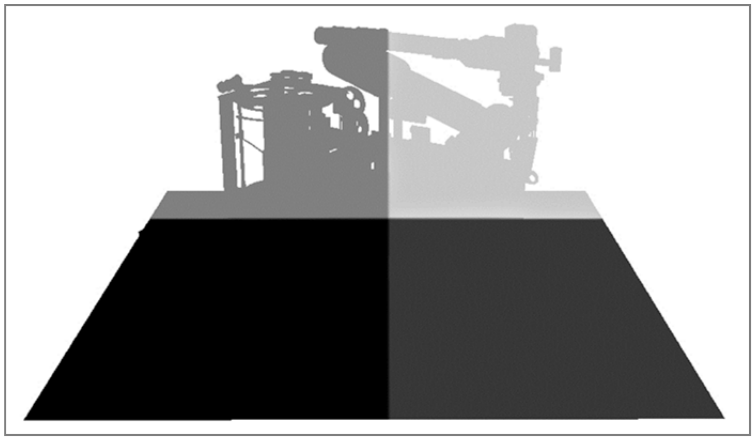

Рис. 2. Текстура координат фрагментов видимых поверхностей сцены

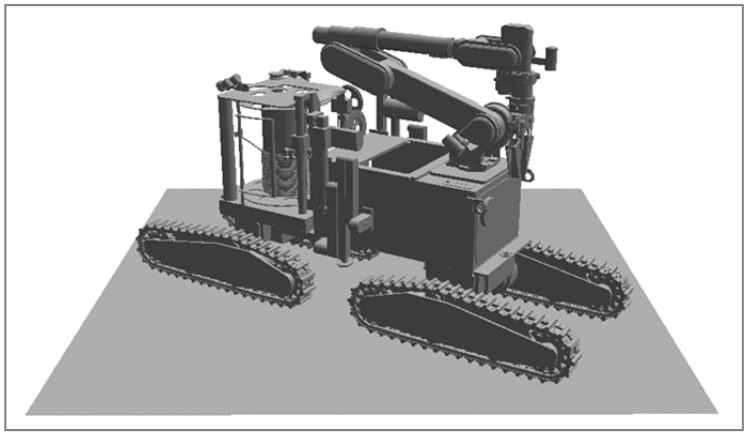

Рис. 3. Текстура нормалей к видимым поверхностям сцены

2. Синтез карты затенения рассеянной освещенности виртуальной среды

Суть построения карты затенения рассеянной освещенности состоит в вычислении для всех видимых наблюдателю точек P поверхностей объектов виртуального пространства, сохраненных в пикселах текстуры G_{pos}, коэффициентов затенения и записи этих коэффициентов в аналогичные пикселы выходной текстуры T_{ao} (размеры текстур G_{pos} и T_{ao} должны быть одинаковы). Для этого вокруг каждой точки P необходимо выделить некоторую область и определить отношение ее части, проникающей внутрь объекта, к общему объему данной области. Точное решение данной задачи является весьма трудоемким и не подходит для реализации в масштабе реального времени, поэтому используется приближенный метод на основе выборки, включающей m точек области. Каждая точка A_t, где $t \in [0, m\text{-}1]$, выбирается таким образом, чтобы она располагалась внутри полусферы некоторого радиуса R с центром в P и ориентированной вдоль нормали N к поверхности в точке P (Рис. 4, слева). Тогда величина коэффициента затенения определяется отношением количества точек выборки, попавших внутрь объекта, к общему числу точек m. Заметим, при использовании сферической области (ядра выборки) половина точек выборки заведомо оказалась бы внутри объекта, что привело бы к ложному затенению поверхности (Рис. 4, справа).

Для реализации выборок необходимо заранее сформировать m случайных векторов S_t смещения каждой точки P из G_{pos} в пределах ее полусферы, где $t \in [0, m\text{-}1]$. Как было отмечено в п.1, координаты P и соответствующей ей нормали N, записанные в текстурах G_{pos} и G_{norm}, представлены в видовой СК VCS. Следовательно, смещение точек тоже надо производить в этой СК. Однако информация в данных текстурах для динамичных сцен и сцен с возможностью изменения положения и ориентации наблюдателя обновляется от кадра к кадру. Поэтому нормаль N, а значит и связанная с ней полусфера, могут иметь бесконечное число ориентаций в пространстве VCS. Выполнение генерации векторов S_t в СК VCS привело бы к необходимости проведения этой операции на каждом кадре с перебором всех пикселов текстуры G_{norm}. Для решения проблемы целесообразно один раз на этапе загрузки программы синтезировать и сохранить векторы S_t в касательном базисе, где нормаль всегда имеет одни и те же координаты (0,0,1), а непосредственно при расчетах коэффициента затенения переводить их в СК VCS. Тогда:

$$S_{t,x} = 2 \cdot f_{\text{rnd}} - 1, \quad S_{t,y} = 2 \cdot f_{\text{rnd}} - 1, \quad S_{t,z} = f_{\text{rnd}},$$

где f_{rnd} – функция генерации псевдослучайного числа в отрезке [0.0, 1.0].

Построение карты затенения выполним параллельно на многоядерном GPU с помощью шейдеров (фрагментного или вычислительного). Каждый поток рассчитывает коэффициент затенения $k_{i,j}$ в точке $P_{i,j}$, для которой координаты положения P_{vcs} и нормали N_{vcs} записаны в пикселах (i, j) текстур G_{pos} и G_{norm} соответственно, и сохраняет его в пикселе (i, j) результирующей текстуры T_{ao}. Векторы смещения $S_{t,\text{vcs}}$ в СК VCS вычислим по формуле

$$S_{t,\text{vcs}} = M_{\text{tbn}} \cdot S_t,$$

где

$$M_{\text{tbn}} = \begin{pmatrix} T_{\text{vcs},x} & B_{\text{vcs},x} & N_{\text{vcs},x} \\ T_{\text{vcs},y} & B_{\text{vcs},y} & N_{\text{vcs},y} \\ T_{\text{vcs},z} & B_{\text{vcs},z} & N_{\text{vcs},z} \end{pmatrix} -$$

матрица преобразования из касательного пространства в видовое; T_{vcs}, B_{vcs} – тангенциальный и бинормальный векторы, представленные

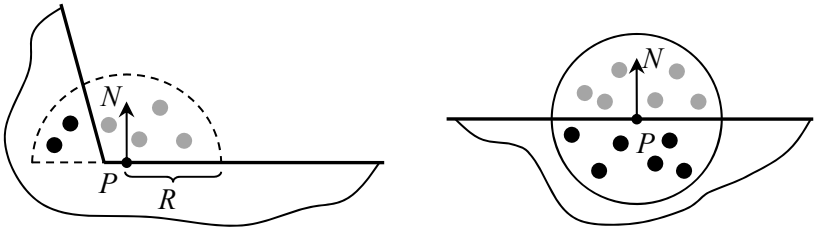

Рис. 4. Полусферическое и сферическое ядро выборки

в СК VCS. Тангенциальный вектор вычислим с использованием процесса Грама-Шмидта, а бинормальный – с помощью векторного произведения N_{vcs} и T_{vcs}

$$T'_{\text{vcs}} = V_r - N_{\text{vcs}}\left(V_r \bullet N_{\text{vcs}}\right),\ T_{\text{vcs}} = \frac{T'_{\text{vcs}}}{|T'_{\text{vcs}}|},$$

$$B_{\text{vcs}} = N_{\text{vcs}} \times T_{\text{vcs}},$$

где V_r – случайный вектор, обеспечивающий поворот полусферы ядра выборки с m векторами $S_{t,\text{vcs}}$ вокруг нормали N_{vcs}, координаты которого равны

$$V_{r,x} = 2 \cdot f_{\text{rnd}} - 1, \quad V_{r,y} = 2 \cdot f_{\text{rnd}} - 1, \quad V_{r,z} = 0.$$

Далее вычислим m точек выборки по формуле

$$A_{t,\text{vcs}} = P_{\text{vcs}} + R \cdot S_{t,\text{vcs}},$$

где R – радиус полусферы ядра выборки (индивидуален для каждой трехмерной сцены), $t \in [0, m\text{-}1]$. После этого выполним расчет их координат $A_{t,\text{scr}}$ в экранном пространстве, применив проекционную матрицу M_{pr} (использовавшуюся в п.1), перспективное деление и преобразование к диапазону [0, 1]:

$$A_{t,\text{ndcs}} = M_{\text{pr}} \cdot A_{t,\text{vcs}},$$

$$A_{t,\text{scr}} = \begin{pmatrix} 0.5 & 0 & 0 & 0.5 \\ 0 & 0.5 & 0 & 0.5 \\ 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \frac{A_{t,\text{ndcs}}}{A_{t,\text{ndcs},w}}.$$

Используя координаты x и y точек $A_{t,\text{scr}}$ в качестве текстурных, считаем из текстуры G_{pos} сохраненные в ней точки $P_{t,\text{vcs}}$. Поскольку каждая пара точек A_t и P_t проецируется в один и тоже пиксел кадра, то обе точки такой пары лежат на одном луче, выходящем из положения виртуального наблюдателя (камеры). Если расстояние от начала луча до точки выборки A_t больше, чем до точки P_t поверхности объекта виртуальной среды, будем считать, что данная точка выборки находится внутри объекта. В противном случае – вне объекта. В СК VCS указанные расстояния можно сравнить с помощью z-координат. Поскольку взгляд наблюдателя направлен в отрицательном направлении оси Z, то расстояние будет больше для той точки, у которой значение z-координаты меньше. Тогда коэффициент затенения $k_{i,j}$ в рассматриваемой точке $P_{i,j}$ найдем по формулам

$$k_{i,j} = \frac{1}{m}\sum_{t=0}^{m-1} F\left(A_{t,ycs,z}, P_{t,ycs,z}\right),$$

$$F\left(z_1, z_2\right) = \begin{cases} 0, & \text{если } z_2 < z_1 + \delta \\ 1, & \text{если } z_2 \geq z_1 + \delta \end{cases},$$

где δ – параметр управления затенением рассеянной освещенности, который позволяет устранять возможный эффект муара на результирующем изображении виртуального пространства. Значение δ подбирается индивидуально для каждой трехмерной сцены. Для получения приемлемого качества картинки при имитации непрямого затенения виртуальной среды количество выборок m следует устанавливать ≥ 32.

Рис. 5 демонстрирует получаемую с помощью описанного подхода карту T_{ao} затенения рассеянной освещенности для вышеупомянутой виртуальной сцены с роботом.

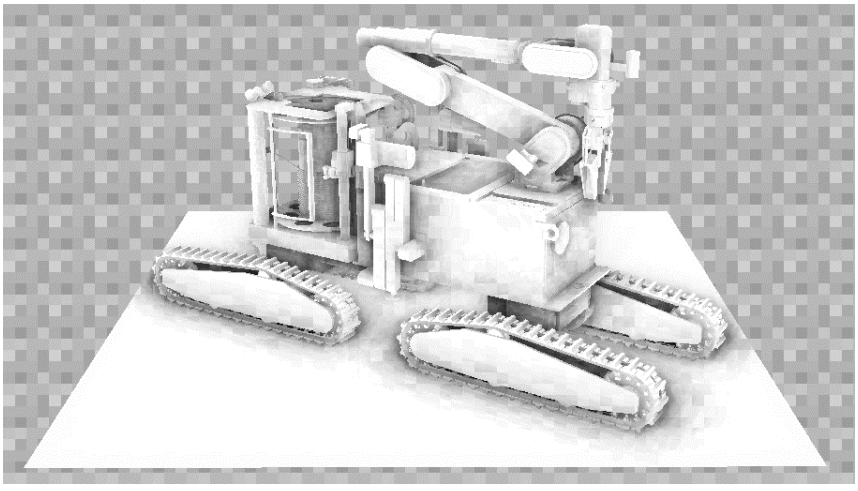

Рис. 5. Карта затенения рассеянной освещенности виртуальной сцены

3. Фильтрация карты затенения

Изменение ориентации набора векторов смещения $S_{t,\text{vcs}}$ путем поворота ядра выборки на некоторый случайный угол (п.2) обеспечивает возможность снизить количество самих требуемых выборок для получения приемлемого результата по скорости визуализации. Однако следствием такого подхода является наличие на подготовленной текстуре затенения достаточно сильных шумов, которые будут в дальнейшем видны и на визуализируемых объектах. Чтобы этого избежать, сгладим полученную текстуру T_{ao}, применив к ней фильтр размытия. Фильтрация производится в отдельном проходе с использованием распределенных вычислений на многоядерном графическом процессоре. Каждый поток на GPU обрабатывает свой пиксел (i, j) текстуры. При этом считывается значение коэффициента затенения $k_{i,j}$ в самом рассматриваемом пикселе, а также в окружающих его соседних пикселах, образующих область N x N, где $N \geq 3$. Результирующее значение коэффициента затенения $k_{i,j}^{res}$ вычисляется по формуле

$$k_{i,j}^{res} = \frac{1}{N^2} \sum_{m,n=-r}^{r} k_{i+m,j+n}, \; r = \left[\frac{N}{2} \right],$$

где квадратные скобки обозначают целую часть числа. При реализации фильтрации с помощью шейдеров полученное значение записывается фрагментным шейдером в пиксел (i, j) выходной текстуры T_{ao}^{f} посредством буфера FBO.

На Рис. 6 показана разница одинаковых участков карты затенения рассеянной освещенности виртуальной сцены с гусеничным роботом до и после применения фильтра размытия. Рис. 7 демонстрирует полную отфильтрованную карту для этой сцены.

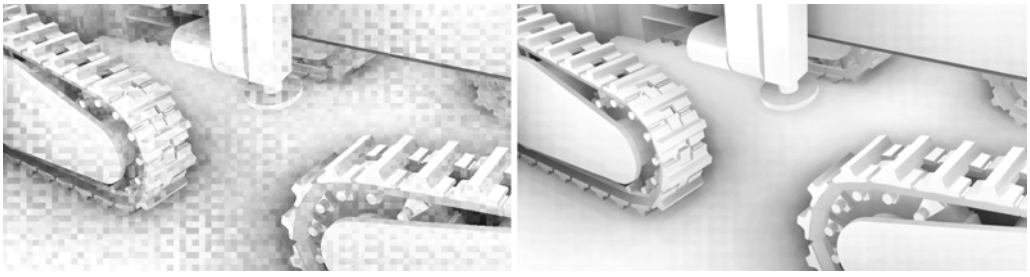

Рис. 6. Фрагмент карты затенения рассеянной освещенности до и после фильтрации

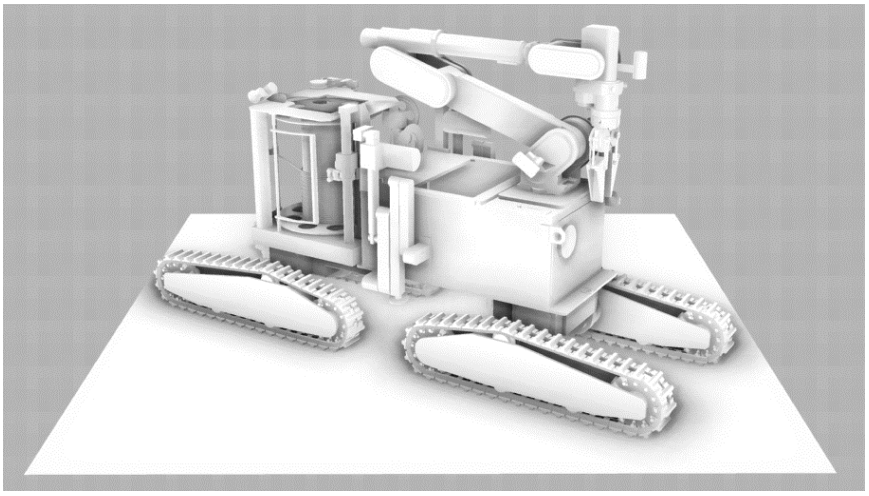

Рис. 7. Отфильтрованная карта затенения

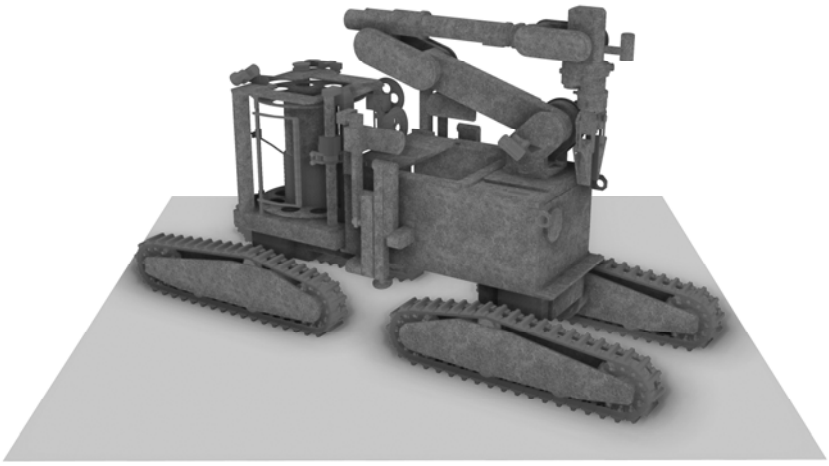

Рис. 8. Визуализация сцены с применением затенения рассеянной освещенности

4. Применение построенной карты затенения при визуализации

Чтобы выполнить имитацию непрямого затенения объектов при визуализации трехмерной виртуальной сцены, применим отфильтрованную карту затенения T_{ao}^{f}, которая была получена на предыдущем этапе, во фрагментном шейдере расчета освещенности. На вход данного шейдера автоматически поступают оконные координаты (x, y, z, $1/w$) обрабатываемого пиксела. Они доступны через стандартную переменную *gl_FragCoord* языка GLSL. Используем x и y в качестве текстурных координат для выборки значения k_{ao} затенения рассеянной освещенности в рассматриваемом пикселе из текстуры T_{ao}^{f}. Полученный коэффициент умножим на значение интенсивности фоновой освещенности I_{amb}.

На Рис. 8 показан результат применения описанных методов и алгоритмов при визуализации ранее упомянутой виртуальной сцены с моделью гусеничного робота (Рис. 1), в которой присутствует только рассеянное освещение. Сравнение полученных изображений (Рис. 1 и Рис. 8) демонстрирует повышение реалистичности и улучшение визуального восприятия объема объектов виртуальной среды в условиях фонового освещения.

Заключение

На базе описанных в статье методов и алгоритмов были созданы программные модули для систем визуализации трехмерных виртуальных сцен в масштабе реального времени. Они позволяют имитировать в трехмерной виртуальной среде непрямое освещение объектов, возникающее в результате наличия рассеянного света, и соответствующее их затенение. Разработка программных компонентов проводилась с использованием возможностей распределенных вычислений на современных многоядерных GPU, графической библиотеки OpenGL и шейдерного языка GLSL. Созданные модули прошли успешную апробацию в составе разработанной в ФГУ ФНЦ НИИСИ РАН системы визуализации «GLView» и продемонстрировали возможность применения предлагаемых решений в имитационно-тренажерных комплексах и системах виртуального окружения.

Литература

1. Jensen H.W. Global illumination using photon maps // In Proceedings of the 7th Eurographics Workshop on Rendering. 1996. pp. 21-30.
2. McGuire M., Luebke D. Hardware-accelerated global illumination by image space photon mapping // ACM SIGGRAPH/EuroGraphics High Performance Graphics. 2009. pp. 77-89.
3. Keller A. Instant radiosity // Proc. SIGGRAPH. 1997. 31 (3). pp. 49-56.

4. Sanjurjo J.R., Amor M., Boo M., Doallo R., Casares J. Optimizing Monte Carlo radiosity on graphics hardware // The Journal of Supercomputing. 2011. vol. 58, issue 2. pp. 177-185.
5. Mittring M. Finding Next Gen – CryEngine 2. SIGGRAPH Advanced Real-Time Rendering in 3D Graphics and Games course. 2007. URL: http://developer.amd.com/wordpress/media/2012/10/Chapter8-Mittring-Finding_NextGen_CryEngine2.pdf (дата обращения: 27.06.2019)
6. Deferred shading // URL: https://en.wikipedia.org/wiki/Deferred_shading (дата обращения: 27.06.2019)
7. Framebuffer object // URL: https://en.wikipedia.org/wiki/Framebuffer_object (дата обращения: 27.06.2019)

Мальцев Андрей Валерьевич. Федеральное государственное учреждение «Федеральный научный центр Научно-исследовательский институт системных исследований Российской академии наук» (ФГУ ФНЦ НИИСИ РАН), г. Москва, Россия. Ведущий научный сотрудник. Кандидат физико-математических наук. Количество печатных работ: 80. Область научных интересов: компьютерная графика, системы виртуальной реальности, информационные технологии.
E-mail: avmaltcev@mail.ru

Methods for Real-Time Distributed Imitation of Indirect Shading in Virtual Environment on GPU

A. V. Maltsev

Federal State Institution “Scientific Research Institute for System Analysis of the Russian Academy of Sciences” (SRISA RAS), Moscow, Russia

Abstract. The paper presents methods and algorithms for distributed imitation of indirect shading of three-dimensional virtual scenes’ objects illuminated by diffused light, based on ambient occlusion technology. Using high performance parallel computing on modern graphics processors provides real-time visualization of virtual environment with such shading simulation.
Keywords: virtual object, visualization, ambient lighting, shading, graphics processor, shaders, real time.

DOI 10.14357/20718632190302

References

1. Jensen H.W. Global illumination using photon maps // In Proceedings of the 7th Eurographics Workshop on Rendering. 1996. pp. 21-30.
2. McGuire M., Luebke D. Hardware-accelerated global illumination by image space photon mapping // ACM SIGGRAPH/EuroGraphics High Performance Graphics. 2009. pp. 77-89.
3. Keller A. Instant radiosity // Proc. SIGGRAPH. 31 (3). 1997. pp. 49-56.
4. Sanjurjo J.R., Amor M., Boo M., Doallo R., Casares J. Optimizing Monte Carlo radiosity on graphics hardware // The Journal of Supercomputing. 2011. vol. 58, issue 2. pp. 177-185.
5. Mittring M. Finding Next Gen – CryEngine 2. SIGGRAPH Advanced Real-Time Rendering in 3D Graphics and Games course. 2007. Available at: http://developer.amd.com/wordpress/media/2012/10/Chapter8-Mittring-Finding_NextGen_CryEngine2.pdf (accessed June 27, 2019)
6. Deferred shading. Available at: https://en.wikipedia.org/wiki/Deferred_shading (accessed June 27, 2019)
7. Framebuffer object. Available at: https://en.wikipedia.org/wiki/Framebuffer_object (accessed June 27, 2019)

Maltsev A. V. PhD. Federal State Institution “Scientific Research Institute for System Analysis of the Russian Academy of Sciences”, 36/1 Nakhimovskiy Av., Moscow, 117218, Russia, e-mail: avmaltcev@mail.ru