# Методы моделирования световых эффектов и искажений видеосигнала в виртуальных средствах наблюдения*

А. В. Мальцев

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

**Аннотация.** В работе предлагаются распределенные методы и подходы для визуализации изображений трехмерных сцен с помощью виртуальных камер, позволяющие имитировать ряд световых эффектов, помех и искажений, которые могут наблюдаться при использовании реальных средств наблюдения. Рассматриваемые решения основаны на применении возможностей современных графических карт с многоядерными GPU, что обеспечивает выполнение рендеринга в масштабе реального времени.

**Ключевые слова:** виртуальная среда, трехмерная сцена, виртуальная камера, искажение сигнала, визуализация, графическая карта, реальное время.

**DOI** 10.14357/20718632190206

## Введение

Системы виртуального окружения и имитационно-тренажерные комплексы активно используются в различных сферах человеческой деятельности. Многие из них основаны на технологиях виртуального моделирования. Это позволяет обучать операторов управлению сложными техническими средствами без необходимости их приобретения для учебных целей и исключает возможность поломки дорогостоящего оборудования в процессе тренировок. Одним из важнейших объектов синтезируемой в тренажерах трехмерной виртуальной среды является виртуальная камера. Она имитирует некоторое реальное средство наблюдения, установленное, например, на роботе, на борту космического корабля и т.п. Получаемый с помощью виртуальной камеры видеопоток позволяет оператору видеть окружающую управляемый объект обстановку, принимать решения и наблюдать результат управления. При этом эффективность обучения и правильность прививаемых навыков в значительной степени зависят от визуального сходства синтезируемого изображения виртуальной среды с изображением, которое оператор мог бы получать от реального средства наблюдения.

При использовании реальных видеокамер в сложных технических устройствах возможны ситуации, когда наблюдаются различные искажения изображения, получаемого с этих камер. Риск возникновения таких ситуаций заметно

*Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 19-07-00393

увеличивается в нештатных ситуациях. Чтобы повысить эффективность обучения операторов с помощью имитационно-тренажерных комплексов и обеспечить выработку правильных навыков управления техническими средствами, при проведении рендеринга трехмерной сцены из виртуального средства наблюдения целесообразно выполнять также моделирование присущих реальным видеосистемам искажений сигнала и световых эффектов на изображении. Световые эффекты могут возникать в оптических системах и матрицах средств наблюдения, а помехи и искажения сигнала – в электронных системах и каналах передачи информации от видеокамеры до устройства отображения.

Несмотря на обилие работ в области визуализации виртуальной среды, вопросу реалистичного моделирования в такой среде средств наблюдения, а также синтеза получаемых от них изображений с учетом возможных световых эффектов и искажений видеосигнала уделено недостаточно внимания. В основном имитируются оптические свойства линз в устройствах наблюдения [1, 2], эффекты виньетирования и хроматической аберрации [3]. Более качественное и полное моделирование виртуальных средств наблюдения реализовано в ряде коммерческих систем визуализации и графических движков таких, как Seagull [4], Unity [5] и др. Однако использование данных продуктов является платным, а применяемые в них методы и алгоритмы, как правило, составляют коммерческую тайну и не описываются в открытых источниках.

В данной работе предлагаются новые распределенные методы и алгоритмы моделирования работы виртуальных средств наблюдения в части визуализации получаемых ими изображений трехмерной виртуальной среды с имитацией ряда световых эффектов, помех и искажений видеосигнала, характерных реальным средствам наблюдения. Далее рассмотрим их подробнее.

## 1. Моделирование помех и искажений видеосигнала

Одним из распространенных типов помех при выводе аналогового видеосигнала на устройство отображения является так называемый белый шум. Он может возникать по целому ряду причин: некачественная приемная антенна, если сигнал доставляется от средства наблюдения к устройству отображения путем эфирной трансляции; неточная настройка принимаемой частоты; слабо экранированные линии передачи; дефекты контактов, кабелей и т.п. Чтобы имитировать такие шумы на видеопотоке, поступающем от виртуальной камеры, выполним распределенную постобработку на графической карте синтезируемых изображений трехмерной виртуальной среды с использованием шумовой текстуры $T_n$ (Рис. 2). Суть подхода состоит в следующем. Сразу после окончания рендеринга очередного кадра визуализируем поверх него прямоугольник $R = V_0V_1V_2V_3$ (размер $R$ соответствует размеру кадра, нумерация вершин выполняется против часовой стрелки, вершина $V_0$ совпадает с левым нижним углом кадра) с наложенной на него картой шумов $T_n$. При этом следует отключить тест глубины и установить ортографическую проекцию. Текстурные координаты $(s, t)$ для вершин $R$ рассчитаем по формулам:

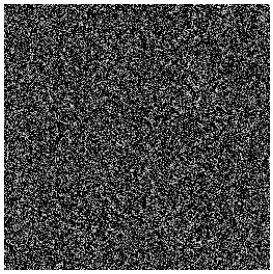

Рис. 1. Текстура шумов

$$(s_0, t_0) = (ds, dt), \quad (s_1, t_1) = (t_x + ds, dt),$$
$$(s_2, t_2) = (t_x + ds, t_y + dt),$$
$$(s_3, t_3) = (ds, t_y + dt),$$
$$t_x = W / T_{n,w}, \; t_y = H / T_{n,h},$$
$$ds = f_{rnd} - 0.5, \; dt = f_{rnd} - 0.5,$$

где $W$, $H$ – ширина и высота кадра, $T_{n,w}$,$T_{n,h}$ – ширина и высота текстуры $T_n$, $f_{rnd}$ – функция генерации псевдослучайного числа в отрезке [0.0, 1.0]. Регулировка плотности и яркости шума производится во фрагментном шейдере, использующемся при отрисовке $R$. Для этого в данный шейдер передается коэффициент $k_n \in$ [0.0, 1.0], устанавливаемый в качестве выходного значения α-канала (отвечает за прозрачность). Чтобы обеспечить корректное наложение моделируемого шума на уже имеющееся в

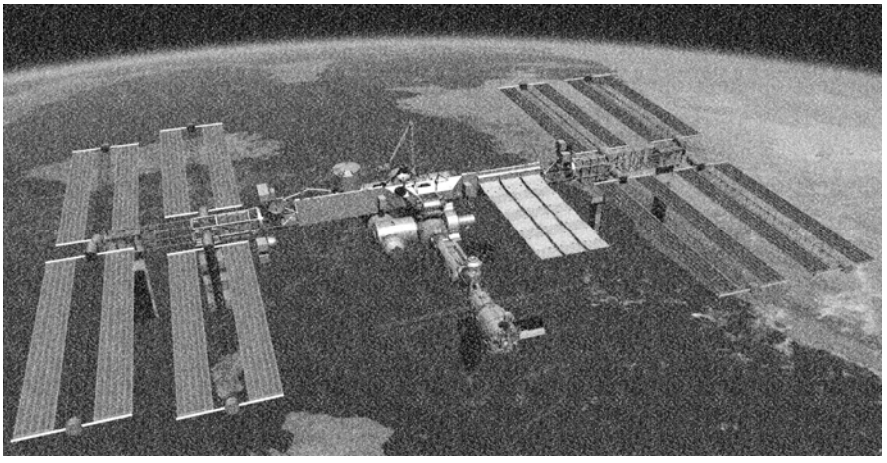

Рис. 2. Моделирование белого шума на изображении 3D сцены

буфере кадра изображение виртуальной среды, перед визуализацией прямоугольника $R$ необходимо задать режим смешивания, при котором цвет $C$ каждого пиксела буфера кадра будет рассчитываться как комбинация его текущего цвета $C_d$ и цвета $C_s$ фрагмента прямоугольника $R$, соответствующего данному пикселу, по формуле:

$$C = \alpha \cdot C_s + (1 - \alpha) \cdot C_d,$$

где $\alpha \in [0.0, 1.0]$ – значение прозрачности фрагмента $R$. Установка такого смешивания производится с помощью вызова функцию *glBlendFunc* графической библиотеки OpenGL с параметрами GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA. Рис. 2 демонстрирует пример применения описанного подхода.

Еще одним типом дефектов, наблюдающимся при выводе видеопотока на устройство отображения, является срыв синхронизации. Проблема может быть вызвана неисправностью электронных систем монитора, несовпадением частоты устройства вывода с частотой смены кадров в поступающем потоке, неправильным разрешением, поломками в кабелях передачи сигнала и т.п. При этом могут наблюдаться горизонтальные искажения и вертикальные скачки изображения. Их моделирование на изображениях, получаемых с виртуальных камер, реализуем на основе распределенной постобработки исходных кадров на видеокарте с многоядерным GPU в масштабе реального времени. Для этого выделим в памяти новую RGB-текстуру $T$ и скопируем в нее текущий буфер кадра, воспользовавшись функцией *glCopyTexSubImage2D* из графической библиотеки OpenGL. Затем создадим сетку кадра, равномерно разбив его на $m$ горизонтальных полос (Рис. 3). Примем длину сторон кадра по каждой из осей системы координат (СК), размещенной в его левом нижнем углу и совпадающей с СК текстуры $T$, за 1. Тогда длина и высота каждой полосы будут равны соответственно

$$dx = 1, \; dy = 1/m.$$

В угловых точках полос разместим вершины $V_0, \ldots, V_N$, где $N = 2m + 1$. Нумерацию выполним слева направо, начиная с левого нижнего угла кадра (Рис. 3). Координаты $(x_i, y_i)$ для $i$-ой вершины определим, как

$$x_i = i \bmod 2, \qquad y_i = [i/2] \cdot dy,$$

где $a \bmod b$ – остаток от деления числа $a$ на число $b$, квадратные скобки – целая часть числа. Для моделирования эффектов горизонтальных искажений и вертикальных скачков изображений виртуальной среды (характерных для срыва синхронизации) произведем расчет текстурных координат для полученных вершин по формулам:

$$s_i = x_i + k_h, \qquad t_i = y_i + k_v,$$

где $k_h$, $k_v$ – горизонтальное и вертикальное отклонения точки кадра от ее положения при корректном отображении видеосигнала на устройстве вывода. Значение $k_v$ одинаково для всех вершин, поскольку при вертикальной рассинхронизации искажения изображения в вертикальном направлении отсутствуют. Для регулировки силы имитируемого эффекта скачков кадра используется коэффициент $v_{\text{desync}} \in [0.0, 0.5]$. Тогда

$$k_v = (f_{rnd} - 0.5) \cdot v_{\text{desync}}.$$

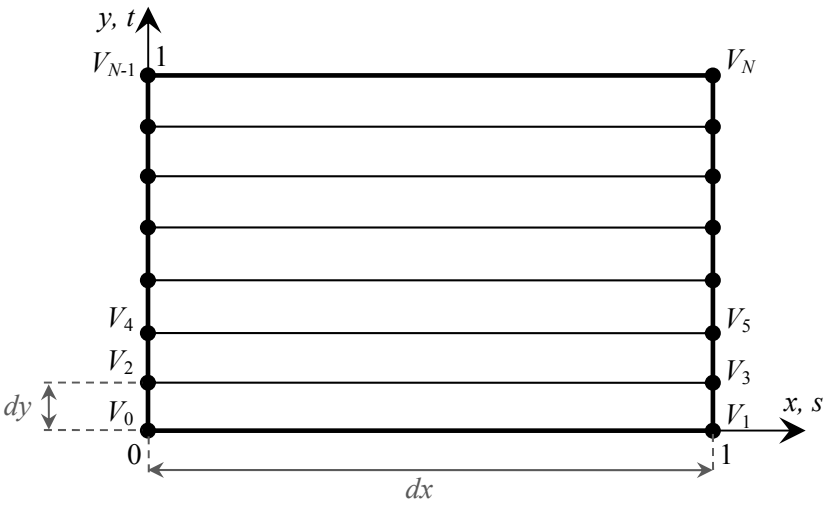


Рис. 3. Сетка кадра при имитации срыва синхронизации

Здесь функция $f_{rnd}$, как и ранее, генерирует псевдослучайное число в отрезке [0.0, 1.0]. Параметр $k_h$ определяет искажения изображения в горизонтальном направлении при моделировании горизонтальной рассинхронизации. Вычислим его значения для каждой пары вершин $V_{2k}$ и $V_{2k+1}$, где $k \in [0, m]$, используя формулу

$$k_h = (f_{rnd} - 0.5) \cdot h_{\text{desync}},$$

где $h_{\text{desync}} \in [0.0, 0.5]$ – коэффициент, влияющий на степень искажений.

Вычислив координаты и текстурные координаты вершин $V_0, \ldots, V_N$, подадим эти данные в графический конвейер видеокарты. Перед рендерингом необходимо установить ортографическую проекцию, отключить смешивание, тесты глубины и прозрачности, а также подключить текстуру $T$, хранящую текущий кадр изображения виртуальной среды. Поскольку значения рассчитанных выше текстурных координат лежат в интервале [-0.25, 1.25], требуется установить для $T$ режим наложения GL_REPEAT. Это можно сделать с помощью функции *glTexParameteri* графической библиотеки OpenGL. На основе переданных в видеокарту вершин визуализируется четырехугольный стрип (*quad strip*, цепочка связанных четырехугольников), на который накладывается текстура $T$. Примеры результатов работы описанного подхода проиллюстрированы Рис. 4 и Рис. 5.

## 2. Моделирование эффектов, возникающих в средствах наблюдения

Оптические и электронные системы самих средств наблюдения также могут порождать ряд эффектов, которые нужно моделировать при реализации виртуальных камер. К наиболее часто встречающимся эффектам относятся засветка,

Рис. 4. Моделирование срыва синхронизации с горизонтальными искажениями изображения

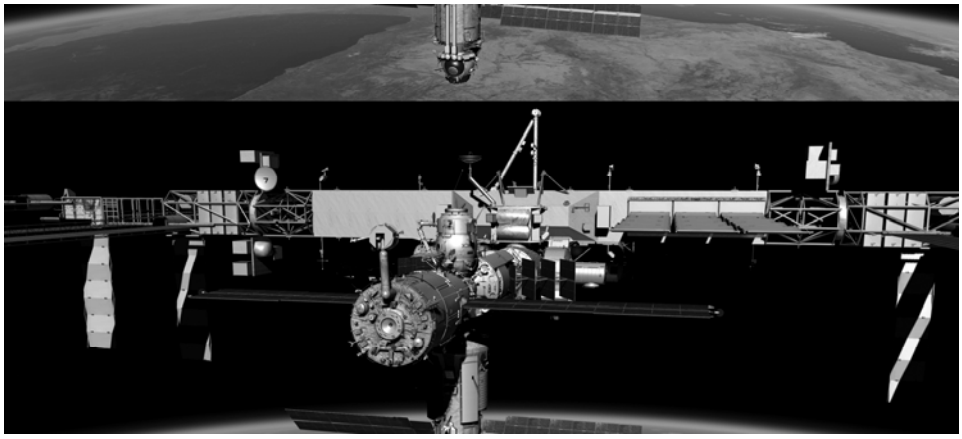

Рис. 5. Моделирование срыва вертикальной синхронизации

пересветка, и расфокусировка. Явления засветки и пересветки возникают вследствие невозможности компенсации средством наблюдения чрезмерного разброса уровней освещенности наблюдаемой визуальной обстановки, например, при попадании солнца в кадр объектива. Эффект расфокусировки наблюдается в оптических системах средств наблюдения при изменении фокусного расстояния объектива. Чтобы имитировать описанные явления в виртуальной среде, целесообразно использовать возможности распределенных вычислений на современных видеокартах, поддерживающих шейдерную обработку.

Для моделирования засветки и пересветки скорректируем изображение, получаемое путем визуализации трехмерной сцены из виртуальной камеры. Для этого вначале создадим текстуру $T$ и скопируем в нее текущий буфер кадра (функция *glCopyTexSubImage2D*), который далее будет полностью перезаписан. Затем установим ортографическую проекцию, отключим смешивание, тесты глубины и прозрачности, а также активируем использование собственного шейдера обработки изображения. После этого визуализируем прямоугольник (размер совпадает с размером кадра) с наложением на него текстуры $T$. Чтение данных из текстуры, их корректировка и окрашивание пикселов производятся во фрагментом шейдере. При этом значение $C$ каждого из цветовых каналов RGB каждого пиксела прямоугольника вычисляется по формуле:

$$C = \left(1+f_1\right)\left(f_2\left\{a_c\right\}+1\right)C_T + a_o,$$
$$f_1 = \left(a_c \geq 2\right),\ f_2 = \left(a_c > 1\right),$$

где $C_T$ – цвет аналогичного канала тексела текстуры $T$, соответствующего рассматриваемому пикселу, $a_c \in [1.0, 3.0]$ – регулируемый коэффициент пересветки, $a_o \in [0.0, 1.0]$ – регулируемый коэффициент засветки, фигурные скобки обозначают взятие дробной части числа. Коэффициенты $a_c$ и $a_o$ передаются в шейдер в качестве его входных параметров. На Рис. 6 и Рис. 7 показаны примеры моделирования засветки и пересветки изображений трехмерной виртуальной сцены с использованием описанного подхода.

Эффект расфокусировки будем моделировать на основе технологии размытия изображения. Суть процесса размытия заключается в замене цвета каждого пиксела кадра на усредненную сумму цветов самого этого пиксела и соседних с ним, лежащих в пределах заданного радиуса. Использование равномерного распределения при расчете весовых коэффициентов каждого из пикселов не обеспечивает приемлемого результата – отсутствует плавность в размытии изображения. Поэтому для решения данной задачи будем использовать фильтр Гаусса, при котором упомянутые весовые коэффициенты рассчитываются на основе закона нормального распределения. Этот подход обладает двумя важными свойствами. Во-первых, размытие Гаусса обладает линейной сепарабельностью (*linearly separable*), то есть может быть применено к двумерному изображению как два независимых одномерных преобразования. Иными словами, алгоритм делится на две части: сначала производится размытие вдоль горизонтальной оси $X$ изображения, затем – вдоль вертикальной оси $Y$.

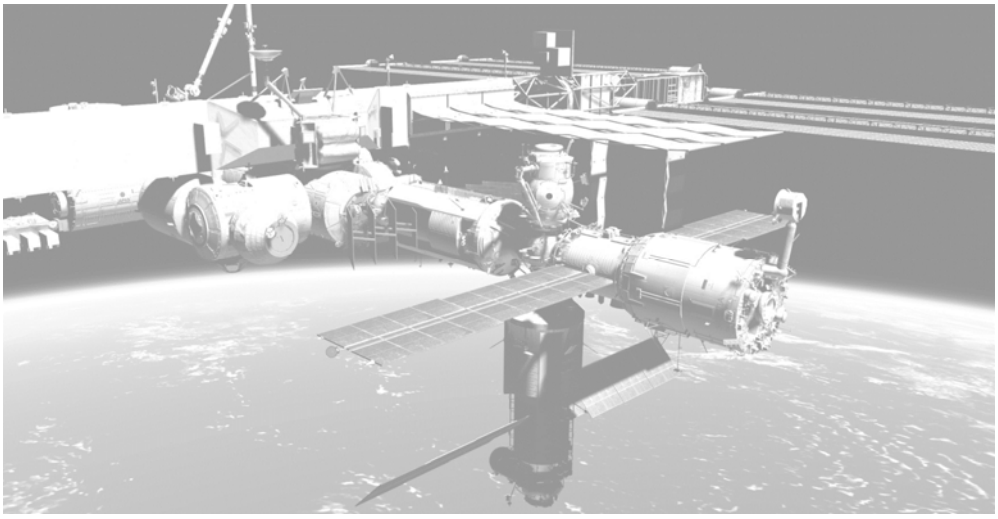

Рис. 6. Моделирование засветки изображения 3D сцены

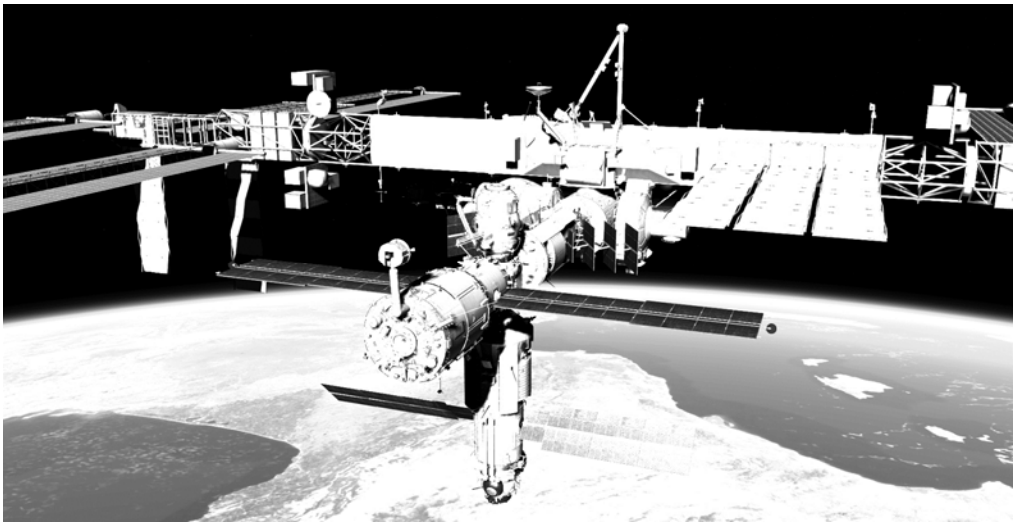

Рис. 7. Моделирование пересветки изображения 3D сцены

Во-вторых, пикселы, находящиеся на расстоянии более 3σ, где σ – среднеквадратичное отклонение в функции Гаусса, не будут иметь существенного влияния на общую сумму цветов, ввиду малого значения весового коэффициента. Поэтому ими можно пренебречь.

Расчет нормализованных сумм цветов для пикселов кадра будем производить параллельно на графическом процессоре во фрагментном шейдере. Для этого, как и в других описанных выше эффектах, будем визуализировать в буфер кадра прямоугольник с наложением текстуры $T$, в которую предварительно скопируем текущий буфер кадра. Однако в данном случае требуется два прохода рендеринга. В первом проходе выполним размытие в горизонтальном направлении. При этом цвет $C$ каждого пиксела выводимого в буфер кадра прямоугольника вычислим по формуле:

$$C = \frac{1}{\sigma^2\sqrt{2\pi}} \cdot \left( C_T(x,y) + \sum_{i=1}^{n} \begin{pmatrix} C_T(x - i \cdot dx, y) + \\ + C_T(x + i \cdot dx, y) \end{pmatrix} \cdot e^{-i^2/2\sigma^2} \right),$$

$$dx = 1/W; \quad n = [3\sigma],$$

где σ – среднеквадратичное отклонение; $C_T(s, t)$ – цвет пиксела текстуры с координатами $(s, t)$; $x, y \in [0.0, 1.0]$ – нормализованные оконные координаты рассматриваемого пиксела; $W$ –

Рис. 8. Моделирование расфокусировки виртуальной камеры

ширина кадра в пикселах; квадратные скобки – целая часть числа. Изображение, полученное в результате произведенного рендеринга, будет записано в буфере кадра. Чтобы выполнить второй этап (вертикальное размытие) этот буфер необходимо снова скопировать в текстуру $T$ и визуализировать прямоугольник с наложением текстуры, аналогично первому этапу, но с расчетом цвета каждого пиксела во фрагментном шейдере по формуле:

$$C = \frac{1}{\sigma^2\sqrt{2\pi}} \cdot \left( C_T(x,y) + \sum_{i=1}^{n} \begin{pmatrix} C_T(x, y - i \cdot dy) + \\ + C_T(x, y + i \cdot dy) \end{pmatrix} \cdot e^{-i^2/2\sigma^2} \right),$$

$$dy = 1/H; \quad n = [3\sigma],$$

где $H$ – высота кадра в пикселах. Для задания величины расфокусировки введем коэффициент $k_f \in (0.0, 1.0]$. Если $k_f = 1.0$, то изображение сфокусировано, то есть выполнение размытия не требуется. В противном случае параметр σ вычислим, как

$$\sigma = \min\left(\frac{1}{2k_f}, 50\right).$$

Пример моделирования расфокусировки виртуальной камеры на базе описанного подхода, представлен на Рис. 8.

## Заключение

На основе описанных в статье методов и подходов были разработаны программные модули для систем визуализации трехмерных виртуальных сцен. Они позволяют обеспечить для виртуальных средств наблюдения имитацию ряда важных световых эффектов, помех и искажений видеосигнала, свойственных тем, которые возникают при использовании реальных устройств. Создание программных компонентов выполнялось с применением возможностей распределенных вычислений на современных видеокартах, графической библиотеки OpenGL и шейдерного языка GLSL. Апробация полученных модулей в составе системы визуализации «GLView», разработанной в ФГУ ФНЦ НИИСИ РАН, показала адекватность предлагаемых решений и возможность их применения при создании систем виртуального окружения и имитационно-тренажерных комплексов.

## Литература


1. Brian A. Barsky, Daniel R. Horn, Stanley A. Klein, Jeffrey A. Pang, Meng Yu. Camera models and optical systems used in computer graphics: part I, object-based techniques // In Proceedings of the 2003 international conference on Computational science and its applications. 2003. pp. 246-255.
2. Brian A. Barsky, Daniel R. Horn, Stanley A. Klein, Jeffrey A. Pang, Meng Yu. Camera models and optical systems used in computer graphics: part II, object-based techniques // In Proceedings of the 2003 international conference on Computational science and its applications. 2003. pp. 256-265.
3. Kučiš M., Zemčík P. Simulation of Camera Features // In Proceedings of the 16th Central European Seminar on Computer Graphics. 2012. pp. 117-123.
4. Система визуализации Seagull // URL: http://www.transas.ru/products/Seagull (дата обращения: 05.04.2019)
5. Unity // URL: https://unity3d.com/ru (дата обращения: 05.04.2019)

**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: 75. Область научных интересов: компьютерная графика, системы виртуальной реальности, информационные технологии. E-mail: avmaltcev@mail.ru

## Methods for simulation of light effects and video signal distortions in virtual surveillance devices

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 distributed methods and approaches for visualizing images of three-dimensional scenes by means of virtual cameras, which allow to simulate a number of lighting effects, noises and distortions that can be observed when real surveillance devices are used. Considered solutions are based on applying the capabilities of modern graphics cards with multi-core GPUs, which ensures an implementation of rendering in real-time.

**Keywords:** virtual environment, three-dimensional scene, virtual camera, signal distortion, visualization, graphics card, real time.

**DOI** 10.14357/20718632190206

## References

1. Brian A. Barsky, Daniel R. Horn, Stanley A. Klein, Jeffrey A. Pang, Meng Yu. Camera models and optical systems used in computer graphics: part I, object-based techniques // In Proceedings of the 2003 international conference on Computational science and its applications. 2003. pp. 246-255.
2. Brian A. Barsky, Daniel R. Horn, Stanley A. Klein, Jeffrey A. Pang, Meng Yu. Camera models and optical systems used in computer graphics: part II, object-based techniques // In Proceedings of the 2003 international conference on Computational science and its applications. 2003. pp. 256-265.
3. Kučiš M., Zemčík P. Simulation of Camera Features // In Proceedings of the 16th Central European Seminar on Computer Graphics. 2012. pp. 117-123.
4. Sistema vizualizatsii Seagull [Visualization system Seagull]. Available at: http://www.transas.ru/products/Seagull (accessed April 05, 2019)
5. Unity. Available at: https://unity3d.com/ru (accessed April 05, 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