Моделирование атмосферных осадков в трехмерных сценах с использованием CUDA[1]

А.В. Мальцев

Аннотация. В работе предлагаются методы реализации атмосферных осадков в трехмерных сценах на основе разработанной технологии распределенного моделирования и визуализации систем частиц с применением современных многоядерных графических процессоров. Рассмотрена реализация в реальном времени падающего снега с помощью текстурированных частиц в виде «спрайтов», а также дождя, формируемого из частиц-капель, имеющих форму полупрозрачных тетраэдров.

Ключевые слова: визуализация в реальном времени, параллельные вычисления, графический процессор, моделирование, система частиц, атмосферные осадки.

Введение

Одним из крупных направлений в трехмерном компьютерном моделировании является создание и визуализация виртуальных сцен, представляющих собой открытые пространства вне помещений. Такие сцены широко используются как в игровой индустрии, так и в области имитационно-тренажерных комплексов. Данные комплексы позволяют осуществлять подготовку специалистов для работы с различными техническими средствами [1-3]. В процессе обучения на тренажерах у человека вырабатываются навыки управления техникой (автомобили, самолеты, корабли и т.д.), что позволяет минимизировать затраты на испорченное дорогостоящее оборудование. Для формирования правильных навыков необходимо, чтобы окружающая оператора виртуальная среда, синтезируемая на компьютере, как можно больше соответствовала реальности. При обеспечении реалистичности открытых трехмерных сцен (вне помещений), важным фактором является моделирование в них таких природных явлений, как атмосферные осадки.

Решение данной задачи обычно основано на применении систем частиц [4, 5]. Известные методы и алгоритмы реализации таких систем, как правило, рассчитаны на использование центрального процессора, позволяющего выполнять расчет параметров частиц либо в последовательном режиме, либо с некоторой степенью распараллеливания. Но даже в самых современных процессорах такое распараллеливание ограничивается всего 6-8 ядрами. Однако количество элементов в системах частиц, используемых для реализации атмосферных осадков (дождь, снег и т.п.), порой может достигать порядка сотен тысяч и даже миллионов. Поэтому сохранить режим реального времени визуализации (частота смены кадров не менее 25 раз в секунду), необходимый для обеспечения плавной и реалистичной динамики виртуальных объектов, становится практически невозможно.

Для моделирования атмосферных осадков в трехмерных виртуальных сценах в данной работе предлагается технология распределенной обработки и визуализации систем частиц в режиме реального времени, основанная на сов-

[1] Работа выполняется при поддержке РФФИ, грант № 13-07-00674.

местном использовании шейдерной обработки и архитектуры параллельных вычислений CUDA [6] на современных многоядерных графических процессорах (GPU). Данная технология поддерживает использование систем частиц с количеством элементов порядка 10^6, сохраняя при этом возможность рендеринга в режиме реального времени, что особо важно для таких областей, как имитационно-тренажерные комплексы и системы виртуальной реальности.

1. Параллельная обработка и визуализация систем частиц на GPU

Предлагаемая технология реализации системы частиц включает два этапа, выполняющихся при синтезе каждого кадра изображения виртуальной сцены.

На *первом этапе*, использующем возможности программно-аппаратной архитектуры CUDA, производится обработка данных системы частиц с нахождением ее состояния для требуемого момента времени, а именно: генерация новых и уничтожение старых частиц, расчет их физических параметров (положения, скорости, времени жизни и т.д.). При этом каждая частица хранится в памяти и обрабатывается как некоторая точка P сцены, являющаяся геометрическим центром частицы. Совокупность координат точки P и параметров, соответствующей ей частицы, можно представить в виде некоторой структуры S. Таким образом, перед выполнением первого этапа мы имеем систему частиц в виде массива M из n элементов типа S, где n – максимально возможное количество частиц в системе. Параметры элементов соответствуют времени T, в которое производилась предыдущая обработка. В результате данного этапа вычисляется массив M', отражающий состояние системы частиц на момент времени $T' = T + \Delta T$, где ΔT – разница между текущим временем и T.

Исходя из данных, содержащихся в полученном массиве M', на *втором этапе* осуществляется синтез необходимой геометрии частиц (с помощью геометрического шейдера), а также их визуализация в буфер кадра, которая включает расчет освещенности от источников света, применение материалов и наложение различных текстур (фрагментный шейдер).

Далее рассмотрим каждый из упомянутых этапов подробно.

1.1. Обработка массива частиц с использованием CUDA

На Рис. 1 представлена предлагаемая схема этапа обработки массива частиц на графическом процессоре видеоадаптера, поддерживающего архитектуру параллельных вычислений CUDA. Данная архитектура подразумевает одновременное выполнение на GPU множеством параллельных потоков одной программы, называемой ядром, но с различными параметрами. За каждый из потоков отвечает один из

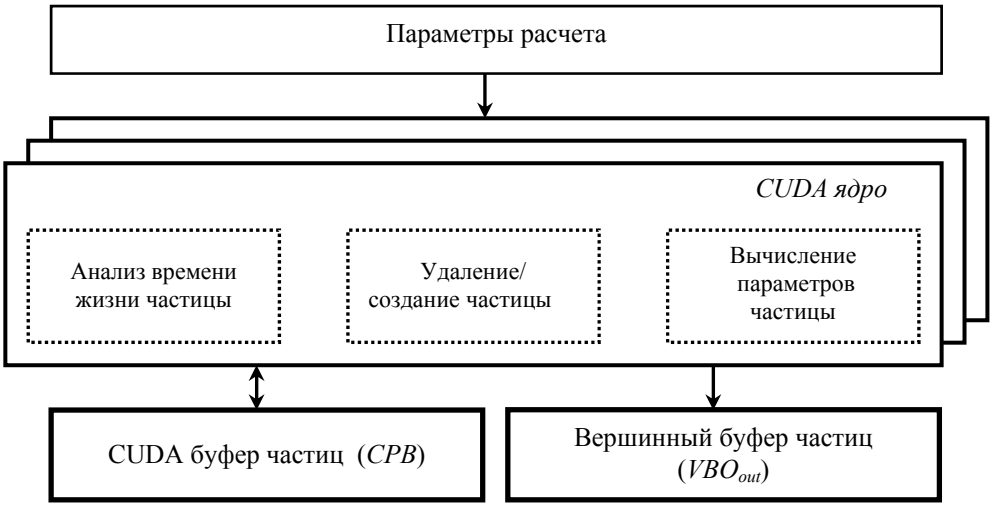


Рис. 1. Обработка массива частиц на GPU с поддержкой CUDA

составляющих GPU процессоров. Максимальное количество потоков, одновременно находящихся в обработке, зависит от характеристик конкретного GPU, и в настоящее время достигает 3072 для чипов NVIDIA GM200-400-A1.

Для реализации представленной схемы выделим в видеопамяти область данных, доступную всем CUDA потокам GPU (Рис. 1, *CPB*), и разместим в ней рассмотренный ранее массив M частиц. Каждый поток $i \in [0, n-1]$ получает на вход данные на момент времени T об обрабатываемой им частице $M[i]$ из буфера *CPB*, а также некоторый набор параметров расчета (положение эмиттера системы частиц, максимальное время жизни и начальная скорость частиц, действующие на систему силы и т.д.). Выполнив операции, предусмотренные ядром (удаление частицы, если время ее жизни больше максимального, и создание новой, вычисление актуальных параметров частицы; подробно эти операции будут рассмотрены в п. 1.2), поток возвращает информацию о частице на момент $T' = T + \Delta T$, перезаписывая ячейку $M[i]$ в *CPB* рассчитанными данными $M'[i]$.

Поскольку дальнейшая визуализация частиц будет производиться с помощью шейдеров в контексте OpenGL, то для обеспечения доступа из шейдеров к данным о частицах, полученным потоками и находящимся в контексте CUDA, в рассматриваемую схему (Рис. 1) также введен вершинный буфер VBO_{out} [7]. Каждый поток i записывает в него ту часть из полученных данных $M'[i]$, которая требуется непосредственно для визуализации частицы (в самом простом случае достаточно координат центра P частицы). Доступность VBO_{out} в обоих упомянутых контекстах обеспечивается механизмом *CUDA OpenGL interoperability*, позволяющим избежать копирования данных из одной области видеопамяти в другую с использованием центрального процессора и шины PCI Express. Для получения доступа из ядра к вершинному буферу VBO_{out}, находящемуся в контексте OpenGL, необходимо перед выполнением этого ядра на GPU вызвать функции *cudaGraphicsGLRegisterBuffer*, *cudaGraphicsMapResources* и *cudaGraphics ResourceGetMappedPointer*, отвечающие соответственно за регистрацию VBO_{out} в CUDA, его отображение в память CUDA и получение указателя на эту память. После выполнения первого этапа необходимо вызвать функцию *cudaGraphicsUnmapResources* освобождающую доступ к буферу для дальнейшего использования его в шейдерах.

1.2. Ядро для обработки частицы

В данной работе мы будем использовать следующий набор параметров для частицы из массива M: геометрический центр – точка P с координатами (P_x, P_y, P_z); время жизни t; скорость $\boldsymbol{v}(v_x, v_y, v_z)$; углы $\boldsymbol{r_0}(r_{0,x}, r_{0,y}, r_{0,z})$ поворота (в момент генерации) частицы относительно осей X, Y, Z системы координат WCS', полученной путем переноса мировой системы координат (WCS) в точку P; изменения $\boldsymbol{r}$ (r_x, r_y, r_z) углов поворота в текущий момент времени (относительно $\boldsymbol{r_0}$). Для удобства вычислений будем работать с координатами точки и векторов, представленными в системе WCS.

Рассмотрим действия, выполняемые ядром для расчета состояния некоторой частицы G из i-ой ячейки ($i \in [0, n-1]$) массива M в момент времени $T' = T + \Delta T$. Пусть определены следующие параметры расчета: длина W и ширина H эмиттера E системы частиц (в нашей задаче мы считаем, что эмиттер имеет форму прямоугольника, и его локальная система координат OCS размещена в центре этого прямоугольника так, что оси X и Y системы параллельны сторонам); начальная скорость частицы $\boldsymbol{v_0}(v_{0,x}, v_{0,y}, v_{0,z})$ в момент генерации; коэффициент $k_v \geq 0$ случайного изменения вектора и величины начальной скорости; действующее на систему ускорение $\boldsymbol{a}(a_x, a_y, a_z)$; $k_{rot} \in [0, \pi]$ – угол максимального поворота частицы от ее начального положения $\boldsymbol{r_0}$; $k_{vel} \geq 0$ – значение скорости поворота; t_{max} – максимальное время жизни частицы. Все векторы также заданы в мировой СК WCS.

Вначале вычислим текущее время жизни t' частицы G, сложив t и ΔT. Если $t' \geq t_{max}$, то необходимо уничтожить рассматриваемую частицу и сгенерировать новую, данные которой будут сохранены в ячейку $M'[i]$. Присвоим t' (вычисляемые для массива M' параметры частиц здесь и далее будем отмечать штрихом) значение 0. Поскольку все новые частицы «рождаются» в пределах эмиттера, то их начальные координаты удобно генерировать в

локальной СК эмиттера, а потом преобразовывать в СК WCS:

$$P' = M_{model}\,(W\,(f_{rand} - 0.5),\, H\,(f_{rand} - 0.5),\, 0,\, 1),$$

где M_{model} – матрица модельного преобразования из СК OCS эмиттера в мировую СК WCS, f_{rand} – функция генерации псевдослучайного вещественного числа из отрезка [0, 1]. Поскольку в нашем примере мы рассматриваем прямоугольный эмиттер, ось Z которого перпендикулярна его плоскости, то z-координата в момент генерации частицы равна 0. Однородная координата $w = 1$ используется, поскольку матрица M_{model} может содержать перенос, влияющий на координаты точек. Вектор $\boldsymbol{v}'$ скорости генерируемой частицы определим с учетом коэффициента k_v случайного изменения заданной начальной скорости $\boldsymbol{v_0}$:

$$\boldsymbol{v}' = \boldsymbol{v_0} + 2k_v V_{rand},$$

где $V_{rand} = (f_{rand} - 0.5, f_{rand} - 0.5, f_{rand} - 0.5)$ – псевдослучайный вектор. Начальные углы поворота частицы относительно осей X, Y, Z системы WCS' зададим по формуле:

$$\boldsymbol{r_0}' = 2\pi(f_{rand}, f_{rand}, f_{rand})$$

И, наконец, вычислим отклонения $\boldsymbol{r}'$ углов поворота от исходного состояния $\boldsymbol{r_0}'$. Пока частица живет, ее поворот может происходить относительно осей СК WCS' как по часовой стрелке, так и против нее. Текущее направление поворота для каждой оси можно кодировать знаком соответствующей компоненты $\boldsymbol{r}'$, а абсолютное значение этой компоненты задавать в отрезке $[0,\ 2k_{rot}]$, где полуинтервал $[0, k_{rot})$ соответствует повороту против часовой стрелки, а отрезок $[k_{rot},\ 2k_{rot}]$ – повороту по часовой стрелке. Тогда

$$\boldsymbol{r}' = 4k_{rot} V_{rand}\,.$$

Таким образом, мы определили параметры новой частицы. Если же при сложении t и ΔT мы получаем, что $t' < t_{max}$, то необходимо вычислить текущее состояние уже существующей частицы и записать его в ячейку $M'[i]$. Положение и скорость частицы определяются, исходя из их прошлых значений, взятых из $M[i]$, ускорения $\boldsymbol{a}$, действующего на систему, и времени ΔT, прошедшего с предыдущего расчета массива M:

$$P' = P + \boldsymbol{v}\,\Delta T + 0.5\boldsymbol{a}(\Delta T)^2,$$
$$\boldsymbol{v}' = \boldsymbol{v} + \boldsymbol{a}\,\Delta T.$$

Начальные углы $\boldsymbol{r_0}$ поворота частицы остаются неизменными, т.е. $\boldsymbol{r_0}' = \boldsymbol{r_0}$. Расчет углов $\boldsymbol{r}'$ рассмотрим на примере компоненты r'_x. Для этого определим ее по формуле:

$$r'_x = |\,r_x\,| + s(r_x)\,k_{vel}\,\Delta T,$$

где $s(x) = 1$ при $x \geq 0$, $s(x) = -1$ при $x < 0$. Далее следует оценить, осталось ли полученное значение r'_x в допустимом диапазоне $[0,\ 2k_{rot}]$. Если $r'_x \leq 0$, то достигнуто максимальное значение угла поворота против часовой стрелки и r'_x нужно приравнять к 0, а при $r'_x \geq 2k_{rot}$ – по часовой и $r'_x = 2k_{rot}$. В обоих случаях выхода за границы диапазона знак r'_x устанавливается противоположным знаку r_x, т.е. направление поворота изменяется. При $r'_x \in (0,\ 2k_{rot})$ знак r'_x совпадает со знаком r_x. Значения остальных компонент $\boldsymbol{r}'$ определяются аналогичным образом.

1.3. Визуализация частиц

Для отображения частиц в буфере кадра необходимо выполнить синтез их геометрии и вычислить попиксельную освещенность с учетом используемых материалов и текстур. На Рис. 2 представлена схема выполнения второго этапа для нашей технологии реализации системы частиц с использованием вершинного, геометрического и фрагментного шейдеров. Рассмотрим ее подробнее.

Вычисленные ранее данные для визуализации частиц, находящиеся в вершинном буфере VBO_{out}, подадим на вход вершинного шейдера. В рассматриваемом подходе этот шейдер не несет никакой вычислительной нагрузки, но поскольку он является обязательной частью графического конвейера видеоадаптера, то он должен лишь передать полученные входные данные на следующую ступень конвейера – геометрический шейдер.

Геометрический шейдер реализует синтез полигональной модели частиц. Он преобразует одну поступающую на вход точку P', которая в нашей технологии соответствует центру частицы, в вершины V_1, V_2, …, V_n модели, а также вычисляет нормали, текстурные координаты и другие требуемые параметры для этих вершин. Форма модели, а значит и количество добавля-

Рис. 2. Синтез формы и визуализация частиц

емых вершин, зависит от той задачи, для решения которой используется система частиц. Она может быть как плоской, так и трехмерной. В данной работе, например, падающие снежинки мы реализуем «спрайтами» (квадратами, постоянно повернутыми лицевой стороной к виртуальной камере) с наложенной на них текстурой, а капли дождя – полупрозрачными тетраэдрами (подробнее об этом будет сказано в п. 2).

Для ускорения процесса визуализации будем строить в геометрическом шейдере модели только тех частиц, которые попадают в полубесконечный конус, описывающий пирамиду видимости виртуальной камеры (вершина конуса совпадает с точкой P_{cam} размещения камеры). Пусть угол раствора конуса равен γ. Тогда условием принадлежности рассматриваемой частицы с центром в точке P' конусу будет выполнение неравенства $\varphi \le \frac{\gamma}{2}$, где φ – угол между вектором $\overline{P_{cam}P'}$ и единичным вектором $\bar{v}$ направления взгляда виртуальной камеры. Отметим, что координаты всех точек и векторов должны быть представлены в одной СК. В нашем случае используется мировая система координат WCS, так как в ней производился расчет положений частиц.

Поскольку графический процессор эффективно выполняет операции скалярного произведения, то указанное выше неравенство можно заменить на

$$\left(\frac{\overline{P_{cam}P'}}{\left|\overline{P_{cam}P'}\right|}, \bar{v} \right) \ge \cos\frac{\gamma}{2}. \qquad (1)$$

Косинус половины угла γ раствора конуса определим, исходя из горизонтального α и вертикального β углов раствора камеры. Пусть l – расстояние до картинной плоскости, a, b и c – половины соответственно ширины, высоты и диагонали прямоугольника пересечения данной плоскости с пирамидой видимости камеры (Рис. 3). Тогда:

$$\cos\frac{\gamma}{2} = \frac{l}{\sqrt{l^2 + c^2}} = \frac{l}{\sqrt{l^2 + a^2 + b^2}}.$$

Так как $a = l\,\mathrm{tg}\frac{\alpha}{2}$ и $b = l\,\mathrm{tg}\frac{\beta}{2}$, то

$$\cos\frac{\gamma}{2} = \left(1 + \mathrm{tg}^2\frac{\alpha}{2} + \mathrm{tg}^2\frac{\beta}{2}\right)^{-1/2}.$$

Подставив вычисленное значение в (1), получим:

$$\left(\frac{\overline{P_{cam}P'}}{\left|\overline{P_{cam}P'}\right|}, \bar{v} \right) \ge \left(1 + \mathrm{tg}^2\frac{\alpha}{2} + \mathrm{tg}^2\frac{\beta}{2}\right)^{-1/2}. \qquad (2)$$

Правая часть этого неравенства вычисляется один раз перед выполнением этапа синтеза и визуализации системы частиц, левая – в гео-

метрическом шейдере для каждой частицы. Обработка частиц, для которых не соблюдается неравенство (2), прекращается.

Во фрагментном шейдере для каждого фрагмента (точки) F каждой частицы определим освещенность от источников света, размещенных в виртуальной сцене, с учетом применяемых к частицам материалов и различных текстур (рельефа, отражения, прозрачности и т.п. [8]). Чтобы вычислить реалистичную попиксельную освещенность от каждого из источников, будем использовать расширенную модель расчета из [9], основанную на модели Фонга-Блинна, но учитывающую тени от объектов и направленные источники света:

$$I = \lambda(k_A I_A + (1 - sh)\cdot k_D I_D \max(0, \boldsymbol{L} \bullet \boldsymbol{N}) + \\ + sh \cdot sc + shade \cdot k_S I_S \max(0, \boldsymbol{H} \bullet \boldsymbol{N})^S),$$

$$sh = density \cdot shade,$$

где k_A, k_D, k_S – коэффициенты отражения материалом частицы соответственно рассеянной, диффузной и зеркальной компонент освещения, I_A, I_D, I_S – интенсивности этих компонент, попадающие в точку F, $\boldsymbol{N}$ – нормаль к поверхности в F, $\boldsymbol{L}$ – единичный вектор из F на источник света, $\boldsymbol{H} = (\boldsymbol{L}+\boldsymbol{V})/|\boldsymbol{L}+\boldsymbol{V}|$ - нормализованный средний вектор между $\boldsymbol{L}$ и единичным направлением $\boldsymbol{V}$ из F на наблюдателя, степень s (действительное число от 0 до 128) характеризует резкость бликов (чем выше s, тем блик меньше и резче), *sc* - цвет тени, *density* - коэффициент плотности тени (действительное число от 0 до 1), *shade* – параметр, определяющий наличие или отсутствие тени в точке расчета. Коэффициент λ зависит от параметров источника света. Его вычисление подробно описано в [9].

2. Реализация атмосферных осадков на основе систем частиц

2.1. Моделирование снега

Для синтеза снежинок в виртуальных сценах на основе описанной выше технологии мы предлагаем использовать модель частицы в виде текстурированного спрайта. При реализации спрайта в случае виртуальной камеры с перспективным проецированием будем формировать квадрат в геометрическом шейдере так, чтобы его центр находился в точке P' размещения частицы, а нормаль $\boldsymbol{N}$ к поверхности квадрата из точки P' смотрела в начало O видовой системы координат VCS (Рис. 4а). Тогда $\boldsymbol{N} = |\, O_{WCS} - P' \,|$, где O_{WCS} – точка O, координаты которой представлены в мировой СК.

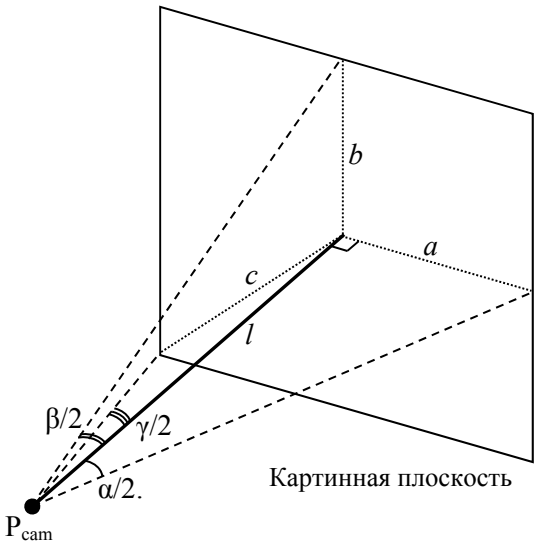


Рис. 3. Вычисление угла γ для конуса

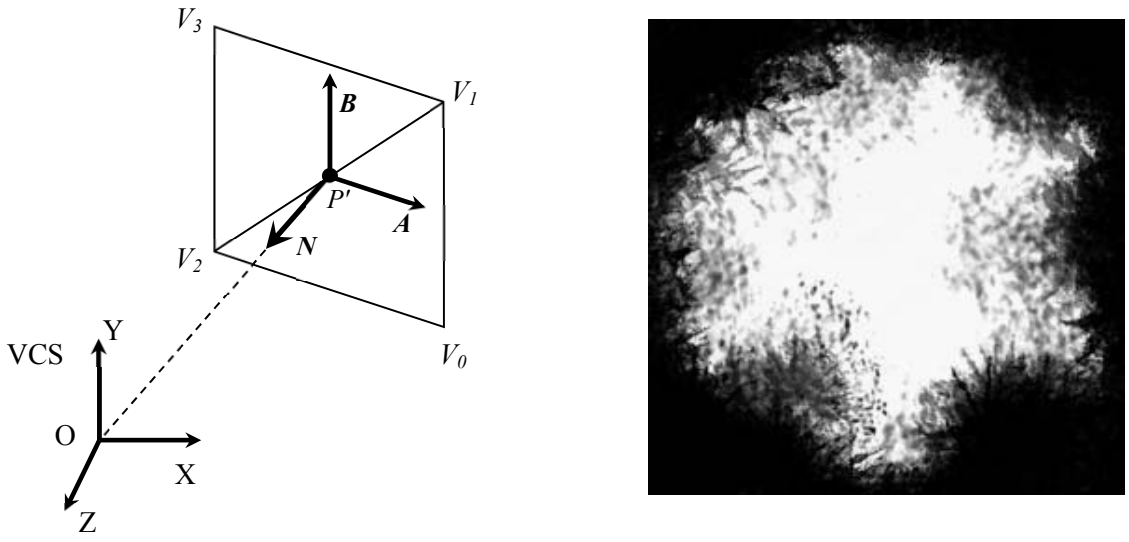
Генерируемый спрайт включает два треугольных полигона с вершинами V_0, V_1, V_2 и V_1, V_2, V_3, которые для оптимизации работы графического конвейера целесообразно представлять как треугольный стрип $V_0V_1V_2V_3$. Чтобы найти координаты вершин, вычислим вначале пару перпендикулярных векторов $\boldsymbol{A}$ и $\boldsymbol{B}$ таких, что $\boldsymbol{A} \parallel V_0V_2$, $\boldsymbol{B} \parallel V_0V_1$:

$$\boldsymbol{A} = [\boldsymbol{Y}_{WCS}, \boldsymbol{N}] \,/\, |\, [\boldsymbol{Y}_{WCS}, \boldsymbol{N}] \,|, \; \boldsymbol{B} = [\boldsymbol{N}, \boldsymbol{A}] \,/\, |\, [\boldsymbol{N}, \boldsymbol{A}] \,|,$$

где $\boldsymbol{Y}_{WCS}$ – вектор $\boldsymbol{Y}$ из базиса системы VCS с координатами в СК WCS, а квадратные скобки обозначают векторное произведение. Тогда, если s – размер стороны спрайта, то:

$V_0 = P' + 0.5s(\boldsymbol{A} - \boldsymbol{B})$, $V_1 = V_0 + s\boldsymbol{B}$,
$V_2 = V_0 - s\boldsymbol{A}$, $V_3 = V_1 - s\boldsymbol{A}$.

Отметим, что прежде чем передавать положения вершин на выход геометрического шейдера, необходимо умножить их на матрицу, равную произведению $M_{proj} \cdot M_V$, где M_{proj} – матрица заданного для виртуальной камеры проекционного преобразования, M_V – матрица перехода из мировой СК WCS в видовую СК VCS.



а) б)

Рис. 4. Построение «спрайта» (а) и применяемая к нему текстура (б)

На созданный спрайт во фрагментном шейдере наложим текстуру прозрачности, изображенную на Рис. 4б, задав основной цвет материала белым. Текстура прозрачности – это двумерная текстура в градациях серого цвета, в которой белые участки указывают на полную непрозрачность, а черные – на абсолютную прозрачность. Все остальные оттенки серого цвета в текстуре соответствуют разной степени полупрозрачности (чем ближе цвет к черному, тем больше прозрачность и наоборот). Текстурные координаты для вершин V_0, V_1, V_2 и V_3 нашего спрайта задаются соответственно как T_0 (1.0, 0.0), T_1 (1.0, 1.0), T_2 (0.0, 0.0) и T_3 (0.0, 1.0). На Рис. 5 представлено изображение виртуальной сцены с моделируемым в ней падающим снегом.

Рис. 5. Моделирование падающего снега в виртуальной сцене

2.2. Моделирование дождя

Моделирование капель дождя выполним с помощью полупрозрачных частиц, имеющих форму тетраэдра, в основании которого лежит равносторонний треугольник с центром в P' (Рис. 6). Пусть ось V_0P' тетраэдра параллельна вектору $\boldsymbol{v}'$ текущей скорости частицы, а высота вычисляется как $h = ks|\boldsymbol{v}'|$, где s – размер частицы, устанавливаемый дизайнером при подготовке виртуальной сцены, k – коэффициент, регулирующий зависимость вычисляемого размера частицы от модуля ее скорости $\boldsymbol{v}'$. Для придания формируемой капле вытянутой формы зададим длину a стороны основания равной $0.1h$.

Единичные направляющие векторы e_x, e_y, e_z осей X, Y, Z локальной системы координат (с центром в P') тетраэдра в СК WCS вычислим по формулам:

$$e_y = -\frac{\boldsymbol{v}'}{|\boldsymbol{v}'|},\ e_z = \frac{\left[\boldsymbol{X}_{WCS}, e_y\right]}{\left|\left[\boldsymbol{X}_{WCS}, e_y\right]\right|},\ e_x = \frac{\left[e_y, e_z\right]}{\left|\left[e_y, e_z\right]\right|},$$

где $\boldsymbol{X}_{WCS}$ – направляющий вектор (1,0,0) оси абсцисс системы WCS. Расположим одну из вершин (V_2) тетраэдра на оси X его СК. Тогда вершины тетраэдра будут определяться как

$V_0 = P' + he_y$, $V_1 = P' - (R/2)e_x - (a/2)\,e_z$,
$V_2 = P' + Re_x$, $V_3 = V_1 + ae_z$,

где $R = \frac{\sqrt{3}}{3}a$ – радиус описанной вокруг основания тетраэдра окружности. Нормали в этих вершинах рассчитаем по формулам:

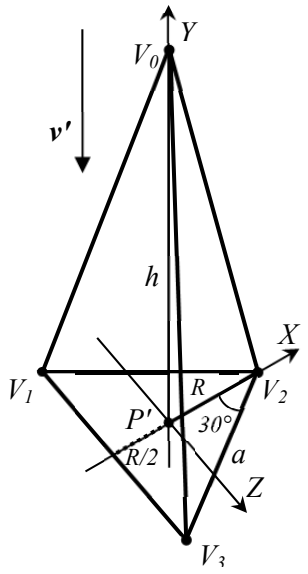


Рис. 6. Капля дождя

$$N_0 = e_y, \qquad N_1 = \frac{V_1 - P'}{|V_1 - P'|} + \frac{V_1 - V_0}{|V_1 - V_0|},$$

$$N_2 = e_x + \frac{V_2 - V_0}{|V_2 - V_0|}, \qquad N_3 = \frac{V_3 - P'}{|V_3 - P'|} + \frac{V_3 - V_0}{|V_3 - V_0|}.$$

Отметим, что, как и в случае со спрайтом (п. 2.1), перед передачей положения вершин на выход геометрического шейдера, необходимо умножить их на произведение матриц $M_{proj} \cdot M_V$.

На Рис. 7 представлено применение описанного подхода для моделирования дождя в трехмерной сцене.

Рис. 7. Моделирование дождя в виртуальной сцене

Литература

1. Михайлюк М.В. Видеотренажерный комплекс управления роботами и манипуляторами // Труды Международного симпозиума «Инновационные технологии в исследовании окружающей среды».– Ларнака-Москва, 2013. – С. 84-85.
2. Михайлюк М.В., Торгашев М.А. Использование технологий виртуальной реальности для моделирования безопасного управления антропоморфными робототехническими средствами // Труды XXI Международной конференции «Проблемы управления безопасностью сложных систем». – Москва, 2013. – С. 290-293.
3. Михайлюк М.В., Брагин В.И. Технологии виртуальной реальности в имитационно-тренажерных комплексах подготовки космонавтов // Пилотируемые полеты в космос. – № 2. – 2013. – С. 82-93.
4. Reeves W.T. Particle systems – a technique for modeling a class of fuzzy objects // In Proceedings of SIGGRAPH ’83. – Detroit, Michigan. ACM, 1983. – Vol. 2. – P. 91-108.
5. Fearing P. Computer modelling of fallen snow // In Proceedings of the 27th annual conference on Computer graphics and interactive techniques. – New York, NY. ACM, 2000. – P. 37-46.
6. CUDA C Programming Guide // NVIDIA Corporation. – 2015. – 240 p. URL: http:// docs.nvidia.com/cuda/pdf/ CUDA_C_Programming_Guide.pdf (дата обращения: 26.03.2015).
7. Боресков А.В. Расширения OpenGL. – СПб.: БХВ-Петербург. – 2005. – 667 с.
8. Мальцев А.В., Михайлюк М.В., Решетников В.Н. Реализация карт отражения окружающей среды в реальном режиме времени // Информационные технологии и вычислительные системы. – 2008. – № 2. – С. 35-42.
9. Мальцев А.В., Михайлюк М.В. Моделирование теней в виртуальных сценах с направленными источниками освещения // Информационные технологии и вычислительные системы. – 2010. – № 2. – С. 68-74.

Мальцев Андрей Валерьевич. Старший научный сотрудник Центра визуализации и спутниковых информационных технологий НИИСИ РАН. Окончил Московский государственный институт радиотехники, электроники и автоматики в 2008 году. Кандидат физико-математических наук. Автор 34-х научных публикаций. Область научных интересов: компьютерная графика, системы виртуальной реальности, информационные технологии. E-mail: avmaltcev@mail.ru