УДК 004.946

ИСПОЛЬЗОВАНИЕ ПРОСТРАНСТВЕННОГО РАЗБИЕНИЯ ПРИ РЕАЛИЗАЦИИ ВЗАИМОДЕЙСТВИЯ СИСТЕМ ЧАСТИЦ С ВИРТУАЛЬНОЙ СРЕДОЙ НА GPU

А. В. Мальцев, Д. В. Омельченко
Федеральный научный центр Научно-исследовательский институт системных исследований Российской академии наук, avmaltcev@mail.ru

Одним из наиболее распространенных типов взаимодействия объектов друг с другом являются столкновения, или коллизии. В работе предложены распределенные методы и алгоритмы вычисления точек коллизий при моделировании взаимодействия систем частиц с объектами виртуальной среды. Рассматриваемые решения основаны на пространственном разбиении трехмерной сцены с помощью регулярной сетки, построение которой выполняется в масштабе реального времени на современном многоядерном графическом процессоре с поддержкой архитектуры параллельных вычислений CUDA. Поиск точки столкновения для каждой частицы также производится на GPU. Для этого анализируются возможные пересечения траектории движения частицы с полигонами, принадлежащими ячейкам сетки, пройденным частицей за промежуток времени между рендерингом текущего и предыдущего кадров изображения виртуальной сцены. На основе предложенных методов и алгоритмов созданы программные модули для систем визуализации трехмерных виртуальных сцен, обеспечивающие распределенное моделирование и визуализацию коллизий систем частиц с объектами виртуальной среды на CUDA-совместимом графическом процессоре в масштабе реального времени. Данные модули прошли успешную апробацию и показали свою применимость в имитационно-тренажерных комплексах и системах виртуального окружения.

Ключевые слова: визуализация, система частиц, регулярная сетка, коллизии, CUDA.

USING SPATIAL SUBDIVISION IN IMPLEMENTING THE INTERACTION OF PARTICLE SYSTEMS WITH VIRTUAL ENVIRONMENT ON GPU

A. V. Maltsev, D. V. Omelchenko
System Research Institute, Russian Academy of Sciences, avmaltcev@mail.ru

Collisions are one of the most common types of object interaction with each other. This paper proposes distributed methods and algorithms for collision point calculation under simulation of particle system interaction with objects of virtual environment. Considered solutions are based on spatial subdivision of three-dimensional scene by means of regular grid, which is built in real time on a modern multicore graphics processor supporting the parallel computing architecture CUDA. Collision point search for each particle is performed on GPU as well. In this regard, possible intersections of particle's movement trajectory with polygons in grid cells are analyzed. Moreover this particle passed the grid at the time interval between rendering of the last and current frames of a virtual scene image. Based on proposed methods and algorithms, software modules for visualization systems of three-dimensional virtual scenes are created. They provide distributed simulation and visualization of particle system collisions with virtual environment objects on the CUDA-supported graphics processor in real time. The modules have been successfully tested and demonstrated their applicability in simulation-training complexes and virtual environment systems.

Keywords: visualization, particle system, regular grid, collisions, CUDA.

Реалистичность моделируемой и визуализируемой на компьютере виртуальной среды достигается целым рядом факторов, таких как детальность виртуальных моделей, точность расчета освещения сцены, динамика объектов и т. д. Существенным фактором обеспечения

правильного восприятия такой среды служит взаимодействие ее объектов друг с другом. Распространенным типом взаимодействия являются столкновения, или коллизии [1]. Моделирование коллизий включает в себя два этапа: определение точек коллизии объектов и расчет поведения этих объектов в результате произошедшего столкновения. Методы и алгоритмы реализации данных этапов напрямую зависят от типов и сложности геометрии взаимодействующих объектов. С этой точки зрения одной из трудоемких и ресурсозатратных задач является расчет коллизий систем частиц [2] с другими объектами виртуальной среды, поскольку каждая такая система может содержать огромное количество элементов (до нескольких миллионов и более).

Одним из известных решений описанной задачи нахождения коллизий между виртуальными объектами является применение методов, базирующихся на анализе изображений в виде 2D-проекций трехмерной сцены на заданные плоскости (image-based методы). Тип проецирования и плоскости выбираются исходя из используемых подходов. Так, в статье [3] предлагаются методы поиска столкновений объектов на основе совместного применения буфера трафарета (stencil buffer) и карт глубины (depth map). В статье [4] рассматриваются методы и алгоритмы, ориентированные на нахождение коллизий систем частиц с другими объектами на графическом процессоре (GPU) с использованием шейдеров и *z*-буфера видеокарты. Однако использование шейдеров для решения вычислительных задач, не связанных напрямую с визуализацией, влечет за собой необходимость дополнительных структур данных в виде некоторого массива текстур, каждую из которых надо обновлять в отдельном проходе рендеринга. Работа [5] предлагает распределенные методы реализации систем частиц и расчета точек их столкновения с объектами виртуальной среды на основе карт глубины на GPU с поддержкой архитектуры параллельных вычислений CUDA. Использование CUDA позволяет повысить эффективность вычислительных алгоритмов, поскольку в таком случае отсутствует необходимость привязывать расчеты к структуре графического конвейера, а также появляется возможность хранить параметры частиц в виде классических массивов в видеопамяти. Несмотря на достаточную простоту image-based методов при решении задач поиска коллизий объектов, они обладают рядом недостатков, что ограничивает область их применения. Среди основных минусов можно отметить необходимость создания одного или нескольких изображений-проекций виртуальной сцены для каждой системы частиц в отдельности, зависимость точности определения коллизий от разрешения этих изображений, а также невозможность установления факта наличия или отсутствия коллизии при проникновении частиц в области сцены, закрытые на проекции изображениями каких-либо объектов.

Для решения описанных проблем в данной работе предлагаются новые распределенные методы и алгоритмы нахождения коллизий систем частиц с объектами виртуальной среды на основе пространственного разбиения трехмерной сцены с помощью регулярной сетки. Построение сетки и использование ее при поиске столкновений частиц с объектами выполняется полностью на CUDA-совместимых GPU, что обеспечивает поддержку визуализации даже сложных виртуальных сцен в масштабе реального времени. Далее рассмотрим предлагаемые решения подробно.

Регулярная сетка

Разбиение трехмерной сцены на совокупность объемных областей применяется при решении различных задач компьютерного моделирования и визуализации. В том числе такой подход активно используется для реализации ускоряющих структур данных в методах трассировки лучей [6–8]. Решаемая нами задача поиска коллизий множества частиц с поверхностями объектов виртуальной среды имеет схожую природу, что обосновывает возможность эффективного применения в ней данного подхода. Разбиение виртуального пространства в этой работе предлагается осуществлять с помощью регулярной сетки – разделения сцены плоскостями, параллельными плоскостям мировой *системы координат* (СК) WCS, на трехмерные

ячейки одинакового размера, называемые вокселами. В нашем случае используются кубические вокселы.

Процесс построения регулярной сетки включает в себя несколько шагов, которые в зависимости от активности объектов сцены выполняются единожды или многократно:

- расчет ограничивающего параллелепипеда *AABB* сцены со сторонами, параллельными координатным плоскостям *СК* WCS;
- разделение полученного *AABB* на вокселы;
- составление таблицы индексов полигонов, пересекающих каждый воксел.

Если объекты сцены, кроме систем частиц, в течение всего времени моделирования и визуализации не изменяют своего положения и ориентации, не добавляются и не исчезают, то такая сцена является статической, а разбиение пространства достаточно выполнить один раз при ее загрузке в память. Время построения сетки при этом не служит определяющим фактором, следовательно, выполнить описанные операции можно с использованием CPU. В противном случае, когда объекты перемещаются от кадра к кадру, создаются и удаляются, границы *AABB* и таблица пересечений вокселов полигонами могут значительно меняться. Это приводит к необходимости перестраивать регулярную сетку перед визуализацией каждого кадра. Реализация перестроения сетки в масштабе реального времени, необходимом для правильной работы систем виртуального окружения и имитационно-тренажерных комплексов, требует специальных распределенных методов и алгоритмов. Ранее нами были предложены эффективные решения для построения регулярной сетки виртуальной сцены с использованием CUDA-совместимого GPU в рамках задачи реализации трассировки лучей. С их подробным описанием можно ознакомиться в [8]. Результатами применения данных методов и алгоритмов являются:

- *AABB* сцены в виде двух его крайних точек – минимальной $B_{\min}$ и максимальной $B_{\max}$, разделенный на кубические вокселы с ребром некоторой длины a ;
- массив A_{vp} пар «номер воксела / номер пересекающего воксел полигона», отсортированный относительно номеров вокселов;
- массив A_n количеств полигонов, пересекающих каждый воксел;
- массив A_i индексов первого вхождения пары с номером каждого воксела в массиве A_{vp}.

Номер каждого полигона является уникальным и кодирует индексы виртуального объекта и соответствующего треугольника этого объекта. Номер воксела вычисляется по формуле:

$$C = k_z \times d_x \times d_y + k_y \times d_x + k_x, \quad (1)$$

где k_x, k_y, k_z – индексы воксела по координатным осям X, Y, Z системы WCS, начиная с нуля от точки Bmin;

d_x, d_y – размерности сетки в ячейках по осям X, Y той же СК.

Поиск коллизий частиц с объектами

Далее будем рассматривать случай с динамической виртуальной средой, где перестроение сетки осуществляется при синтезе каждого кадра изображения сцены. После того, как регулярная сетка сформирована по технологии, изложенной в [8], указатели на ее данные предаются в CUDA-ядро, отвечающее за определение текущего состояния системы частиц. Каждый вычислительный поток отвечает за обработку одной частицы в виде точки пространства, соответствующей центру ее трехмерной модели. Поток рассчитывает новое положение, скорость, время жизни и другие параметры своей частицы. Также он выполняет поиск возможной коллизии частицы с каким-либо объектом сцены и ее обработку. Для этого вначале необходимо понять, какие ячейки сетки пересекла частица, преодолев путь от своего предыдущего положения P0 до текущего положения P1. Поскольку вычисление параметров и отображение

системы частиц происходит в момент формирования очередного кадра изображения, а визуализация сцены должна выполняться в масштабе реального времени (т. е. временной интервал между кадрами составляет не более 40 мс), то этот путь аппроксимируется отрезком P0P1 (рис. 1). Номера C0 и C1 вокселов сетки, содержащих точки P0 и P1 соответственно, вычисляются с помощью формулы (1). При этом индексы kx, ky, kz вокселов определяются как:

$$k_x = \left[\frac{1}{a}\left(P_x - B_{\min,x}\right)\right],\ k_y = \left[\frac{1}{a}\left(P_y - B_{\min,y}\right)\right],\ k_z = \left[\frac{1}{a}\left(P_z - B_{\min,z}\right)\right],$$

где a – длина ребра воксела;

$B_{\min}$ – минимальная точка *AABB* сцены;

P_x, P_y, P_z – координаты положения частицы в мировой системе WCS;

скобки обозначают округление до ближайшего целого снизу.

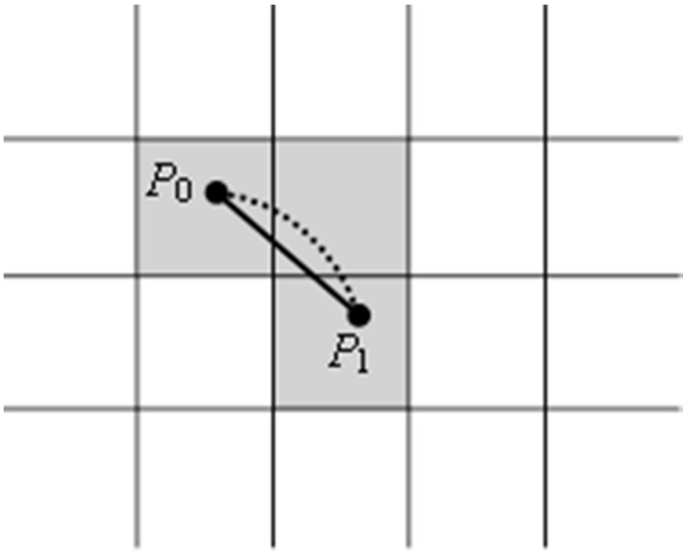

Рис. 1. Аппроксимация пути частицы

Если вокселы C_0 и C_1 не совпадают или не являются смежными, то промежуточные ячейки, пересекаемые рассматриваемым отрезком, можно найти, исходя из направления нормализованного вектора $\overline{P_0P_1}$. Для простоты рассмотрим решение задачи на примере двумерной регулярной сетки, а потом проведем аналогии для трехмерного случая. Пусть точка P_0 находится внутри ячейки, имеющей индексы k_x, k_y, а для координат x, y вектора $\overline{P_0P_1}$ соблюдаются неравенства $x > 0$, $y \geq 0$ (рис. 2). В зависимости от соотношения углов α и β отрезок P_0P_1 перейдет из текущей ячейки либо в $(k_x + 1, k_y)$ – соседнюю справа (при $\alpha < \beta$), либо в $(k_x, k_y + 1)$ – соседнюю сверху (при $\alpha > \beta$), либо по диагонали в ячейку $(k_x + 1, k_y + 1)$, если $\alpha = \beta$. Заменим углы α и β их тангенсами:

$$tg\,\alpha = \frac{y}{x},\quad tg\beta = \frac{1-t}{1-s},$$

где (s, t) – относительные координаты точки P_0, отсчитываемые в долях ячейки от ее левого нижнего угла и вычисляемые как:

$$s = \frac{P_{0,x} - B_{\min,x}}{a} - k_x,\quad t = \frac{P_{0,y} - B_{\min,y}}{a} - k_y.$$

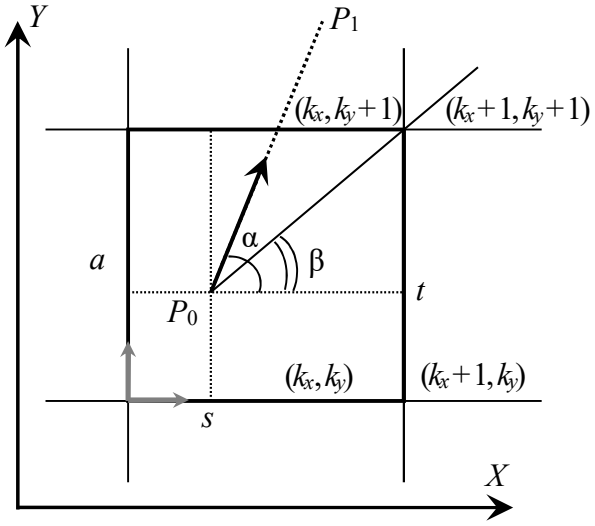

Рис. 2. Двумерная регулярная сетка

Тогда индексы следующей ячейки, в которую перейдет отрезок, равны:

$$k_x' = k_x + 1 - F(\operatorname{tg}\beta - \operatorname{tg}\alpha) = k_x + 1 - F\left(\frac{x(1-t) - y(1-s)}{x(1-s)}\right),$$

$$k_y' = k_y + 1 - F(\operatorname{tg}\alpha - \operatorname{tg}\beta) = k_y + 1 - F\left(\frac{y(1-s) - x(1-t)}{x(1-s)}\right). \quad (2)$$

Значение функции $F(w)$ определяется знаком аргумента w:

$$F(w) = \begin{cases} 1, & w < 0 \\ 0, & w \geq 0 \end{cases}.$$

Так как ранее было определено, что значение $x > 0$, а $s \in [0, 1]$, то значение функции F не зависит от знаменателей ее аргументов. Поэтому данные знаменатели в формулах (2) можно отбросить:

$$k_x' = k_x + 1 - F(x(1-t) - y(1-s)),$$

$$k_y' = k_y + 1 - F(y(1-s) - x(1-t)). \quad (3)$$

Заметим, что при переходе от формул (2) к формулам (3) для координаты x вектора $\overline{P_0P_1}$ становится допустимым значение 0. Аналогично рассмотренному случаю выписываются формулы для векторов $\overline{P_0P_1}$, находящихся во втором, третьем и четвертом квадрантах. Объединив их с (3), получим решение для произвольного вектора $\overline{P_0P_1}$:

$$k_x' = k_x + i(1 - F(|x| f_y - |y| f_x)),$$

$$k_y' = k_y + j(1 - F(|y| f_x - |x| f_y)),$$

$$i = 1 - 2F(x), \quad j = 1 - 2F(y), \quad f_x = 1 - F(x) - i \times s, \quad f_y = 1 - F(y) - j \times t. \quad (4)$$

Чтобы решить задачу поиска следующего воксела для трехмерной сетки, необходимо рассмотреть проекции текущего воксела и отрезка P_0P_1 на плоскости XY, ZY и XZ мировой СК

WCS. Каждая из проекций с точностью до обозначения осей соответствует разобранному ранее двумерному случаю. Каждый из индексов k_x, k_y, k_z будет изменяться при переходе отрезка из одной ячейки в другую, когда он изменяется в двух из трех проекций: *XY* и *XZ* для k_x, *XY* и *ZY* для k_y, *ZY* и *XZ* для k_z. Изменение индекса в проекциях определяется по формулам, аналогичным (4). Объединяя попарно формулы для соответствующих проекций, следующий воксел можно вычислить как:

$$k_x' = k_x + i\,((1 - F(|x|f_y - |y|f_x)) \,\&\&\, (1 - F(|x|f_z - |z|f_x))),$$

$$k_y' = k_y + j\,((1 - F(|y|f_x - |x|f_y)) \,\&\&\, (1 - F(|y|f_z - |z|f_y))),$$

$$k_z' = k_z + k\,((1 - F(|z|f_y - |y|f_z)) \,\&\&\, (1 - F(|z|f_x - |x|f_z))),$$

$$i = 1 - 2F(x), \quad j = 1 - 2F(y), \quad k = 1 - 2F(z),$$

$$f_x = 1 - F(x) - i \times s, \quad f_y = 1 - F(y) - j \times t, \quad f_z = 1 - F(z) - k \times r,$$

где && – означает логическую операцию «И», (s, t, r) – относительные координаты точки P_0, отсчитываемые в долях воксела от его угла с наименьшими координатами в СК WCS и вычисляемые как:

$$s = \frac{P_{0,x} - B_{\min,x}}{a} - k_x, \quad t = \frac{P_{0,y} - B_{\min,y}}{a} - k_y, \quad r = \frac{P_{0,z} - B_{\min,z}}{a} - k_z.$$

Обрабатывающий частицу CUDA-поток на GPU последовательно проходит вокселы, пересекаемые отрезком P_0P_1, начиная с воксела C_0. В каждом из них происходит анализ наличия пересечения отрезка с входящими в ячейку полигонами. Как было описано выше, данные о количестве и номерах таких полигонов хранятся в массивах, в виде которых без учета коллизий представлена регулярная сетка. Так, число полигонов, содержащихся в вокселе, считывается потоком из массива A_n по номеру C этого воксела, который определяется по формуле (1). Номера полигонов хранятся последовательно в массиве A_{vp}, начиная с элемента с индексом A_i [C]. При нахождении хотя бы одного пересечения отрезка P_0P_1 с полигоном, лежащего внутри рассматриваемого воксела, проход по вокселам останавливается, а точка этого пересечения является точкой коллизии частицы с объектом. Если пересечений несколько, то выбирается ближайшее из них к началу P_0 отрезка.

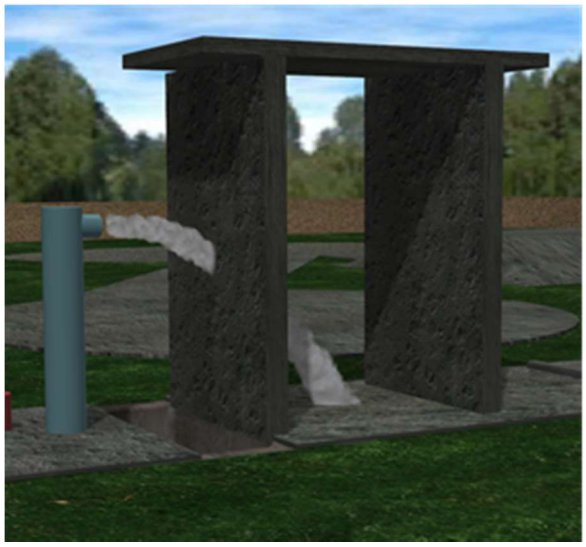

Рис. 3. Моделирование струи пены

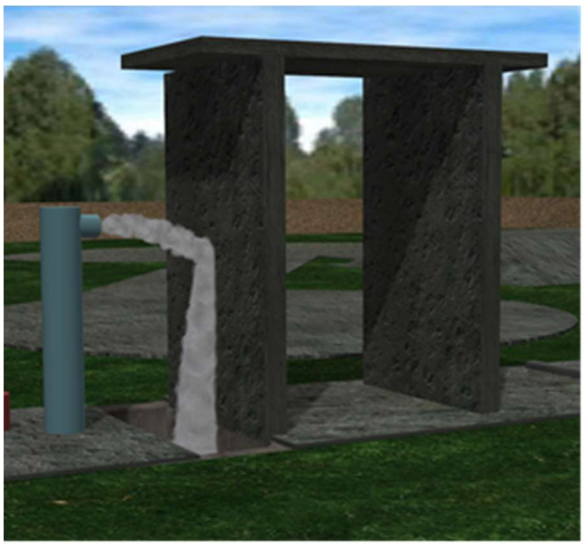

Рис. 4. Моделирование струи пены с учетом коллизий

На рис. 3 и 4 продемонстрирована виртуальная сцена с моделированием струи пены без учета столкновений ее частиц с объектами (рис. 3), а также с поиском и обработкой их коллизий на основе предложенных методов и алгоритмов, использующих регулярную сетку (рис. 4).

Заключение. В результате проведенных исследований были разработаны оригинальные методы и подходы для распределенного моделирования коллизий систем частиц с другими объектами трехмерных виртуальных сцен. Особенностью предлагаемых решений является использование пространственного разбиения сцены с помощью регулярной сетки, построение и применение которой для поиска столкновений частиц с поверхностями объектов выполняется на многоядерном графическом процессоре, поддерживающем архитектуру параллельных вычислений CUDA. Широкое применение распределенных вычислений на GPU обеспечивает работу предлагаемых методов и алгоритмов в масштабе реального времени для высокополигональных сцен, включающих несколько миллионов частиц. Предложенные решения реализованы в виде программных модулей и прошли успешную апробацию в составе системы визуализации, разработанной в ФГУ ФНЦ НИИСИ РАН.

Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 16-07-00796.

Литература

9. Ming C. Lin, Gottschalk S. Collision detection between geometric models: a survey // Proc. of IMA conference on mathematics of surfaces. 1998. V. 1. P. 602–608.

10. Reeves W. T. Particle systems – a technique for modeling a class of fuzzy objects // Computer Graphics. SIGGRAPH. 1983. P. 359–376.

11. Baciu G., Wong S.-K. Image-based techniques in a hybrid collision detector // IEEE Trans. on Visualization and Computer Graphics. 2003. V. 9. P. 254–271.

12. Kolb A., Latta L., Rezk-Salama C. Hardware-based Simulation and Collision Detection for Large Particle Systems // Proc. of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware. 2004. P. 123–131.

13. Мальцев А. В., Страшнов Е. В. Использование z-буфера для поиска столкновений частиц с объектами 3D сцены // Тр. НИИСИ РАН. 2018. Т. 8. № 1. С. 52–55.

14. Havran V., Herzog R., Seidel H.-P. On the fast construction of spatial hierarchies for ray tracing // Proceedings of the 2006 IEEE Symposium on Interactive Ray Tracing. 2006. P. 71–80.

15. Torres R., Martin P. J., Gavilanes A. Ray casting using a roped BVH with CUDA // 25th Spring Conference on Computer Graphics (SCCG 2009). P. 107–114.

16. Мальцев А. В. Построение адаптивной регулярной сетки трехмерной сцены в реальном режиме времени // Программные продукты и системы. 2010. № 4. С. 41–45.