УДК 681.3.069

АНИМАЦИЯ СИСТЕМ ЧАСТИЦ НА ГРАФИЧЕСКОМ УСКОРИТЕЛЕ

Д. А. Гладкий, И. В. Белаго, Н. А. Елыков, С. А. Кузиковский

Институт автоматики и электрометрии СО РАН,
630090, г. Новосибирск, просп. Академика Коптюга, 1
E-mail: glad@sl.iae.nsk.su

Рассмотрен подход к реализации анимации систем частиц на графическом ускорителе. Детально описана балансировка нагрузки между центральным процессором и видеокартой. Предложены оригинальные подходы к уменьшению потока данных между системной и видеопамятью. Проведено сравнение производительности классического и предложенного подходов.

Ключевые слова: компьютерная графика, системы частиц, графический ускоритель, анимация.

Введение. Одним из основных методов получения трехмерных изображений в современных приложениях реального времени является растеризация полигональных моделей. Однако существуют задачи, которые сложно решаются данным методом, например моделирование многогранниками объектов, имеющих нечеткую форму, которыми являются различные природные явления: дым, пар, огонь, пыль, брызги воды. Наиболее эффективным методом моделирования данных объектов является система частиц. Каждое природное явление представляется набором частиц, определяющим пространственный объем, заполняемый данным аморфным объектом. Скорость рендеринга и анимации уменьшается с ростом числа обрабатываемых частиц, при этом увеличение их количества помогает получать более реалистичные изображения. Обычно в приложениях реального времени, таких как компьютерные игры, единовременно обрабатывается от 5000 до 10000 частиц, поскольку использование большего количества осложняется ограниченными вычислительными ресурсами.

До недавнего времени анимация частиц реализовывалась на центральном процессоре, а растеризация производилась с помощью графического акселератора. Однако в приложениях виртуальной реальности центральный процессор используется не только для массивных математических расчетов, которые требуются для обновления состояния систем частиц. Процессорное время тратится также на обработку различных событийно-ориентированных систем (графический интерфейс пользователя, системы искусственного интеллекта), загрузку данных с жесткого диска, обработку взаимодействия объектов виртуального мира и другие задачи. С появлением видеокарт, обладающих программируемым конвейером, стало возможным производить математические расчеты без использования центрального процессора. Графический акселератор обладает высокопроизводительным параллельным векторным процессором, позволяющим осуществлять больше векторных операций в единицу времени, чем процессор общего назначения. Использование видеокарты в качестве математического сопроцессора позволяет увеличить количество частиц, присутствующих в трехмерной сцене, до десятков тысяч. Перенос нагрузки с центрального процессора также дает возможность увеличить время обработки других задач, например скелетной анимации.

На данный момент уже существует ряд работ, описывающих использование графического ускорителя в качестве сопроцессора для анимации систем частиц. Так, в работе

[1] описаны основные идеи анимации частиц на видеокарте. В [2] детально описан алгоритм оптимизации сортировки частиц на графическом ускорителе. Однако в работах, посвященных рассматриваемому вопросу, уделялось мало внимания балансировке вычислительной нагрузки между центральным процессором и видеокартой. Также до сих пор не были представлены подходы к уменьшению потока данных между графической и системной памятью, величина которого напрямую влияет на скорость обработки данных системой анимации частиц в целом.

Целью данной работы является создание унифицированной программной системы поддержки различных моделей анимации частиц, использующей эффективные алгоритмы сжатия потока данных между системной и видеопамятью. В ходе ее реализации предложен алгоритм анимации частиц, балансирующий нагрузку между вычислительными аппаратными модулями системы, на которой запускается приложение. Также разработан новый подход к построению и анимации формы частицы, использующий возможности программируемого вершинного конвейера видеокарты.

Постановка задачи. Для современных приложений виртуальной реальности приемлемой частотой обновления изображения считается 30–60 кадр./с, что составляет от 33 до 17 мс на обработку одного кадра. Исследования показали, что приемлемым временем, затрачиваемым на обработку системы частиц, является до 5 мс работы видеоускорителя и центрального процессора при запуске приложения на типичной конфигурации аппаратной платформы (далее «целевой»): центральный процессор Athlon X2 Dual 4600+, видеокарта GeForce 8800. При реализации системы необходимо учесть наличие у пользователей широкого спектра графических акселераторов. Данная особенность приводит к требованию масштабируемости приложения на различные аппаратные платформы как новые (GeForce 8xxx, Radeon X2xxx), так и прошлых поколений (GeForce 5xxx, Radeon 9xxx).

Реалистичность получаемого изображения напрямую зависит от количества частиц, используемых для построения того или иного эффекта. При реализации ряда эффектов выявлено, что для достижения высокого уровня правдоподобности получаемого изображения в приложении виртуальной реальности реального времени, например в автомобильном симуляторе, необходимо использовать от 4000 до 7000 частиц. Однако с учетом постоянного роста уровня требований к приложениям виртуальной реальности целесообразно обеспечить поддержку анимации до 16000 частиц.

Типичными требованиями, предъявляемыми к системе частиц, являются большое количество настраиваемых моделей анимации частиц и возможность создания различных по сложности форм частиц (ориентированные к порту просмотра прямоугольники [3], выпуклые многогранники и прочие полигональные модели). При отсутствии, например, какой-либо модели анимации программист, поддерживающий систему частиц, должен иметь возможность быстро, не внося изменений в существующий программный код, реализовать требуемую функциональность, т. е. система частиц должна быть легко расширяемой и настраиваемой.

Таким образом, на основе предъявленных требований можно сформулировать следующие характеристики системы:

1) поддержка анимации до 16000 частиц;

2) малое время, затрачиваемое системой на обработку кадра (до 5 мс) на целевой платформе;

3) поддержка широкого ряда графических акселераторов;

4) простота расширения алгоритмов анимации частиц.

Проектирование системы. Обработка систем частиц, как уже отмечалось ранее, является вычислительно сложной задачей. Обновление состояния частицы требует как векторных операций (обновление позиции и скорости, обработка столкновений, вращение

и т. д.), так и скалярных (анимация прозрачности, размера и т. д.). Обновление состояния системы частиц можно разделить на следующие этапы:

1) создание и удаление частиц;
2) вычисление новых позиций и скоростей;
3) обработка столкновений с другими объектами виртуального мира;
4) сортировка частиц;
5) построение буфера вершин моделей, описывающих форму частицы;
6) анимация формы частицы.

Балансировка нагрузки центрального процессора и видеокарты. Создание и удаление частиц являются операциями, требующими произвольного доступа к памяти и большого количества ветвлений для проверки условий необходимости удаления, что в силу аппаратных ограничений не может быть эффективно реализовано с помощью графического ускорителя. Поэтому наиболее оптимальным будет производить данную операцию на процессоре общего назначения.

Обработка движения и столкновения частиц с другими объектами виртуального мира требует большого числа векторных операций. Для уменьшения количества вычислений при анимации движения не учитывается взаимодействие частиц друг с другом, что позволяет обрабатывать частицы параллельно и независимо друг от друга. Поэтому наиболее эффективно проводить обработку движения частиц, используя возможности программируемого конвейера графического ускорителя. Программируемый блок вершинного конвейера не позволяет напрямую сохранять в память какие-либо результаты вычислений, что создает трудности при учете влияния на движение частицы стохастических сил. Наиболее подходящим алгоритмом является итеративная анимация движения с использованием программируемого пиксельного конвейера видеокарты, аппаратная архитектура которого позволяет производить запись и чтение из специальной области видеопамяти — текстурной памяти.

Упорядочение частиц по удаленности относительно наблюдателя необходимо для корректной работы алгоритма взвешенного смешивания, с помощью которого реализуются полупрозрачные частицы. По данным, приводимым компанией ATI (Канада) [4], сортировка 16000 чисел с плавающей точкой одинарной точности составляет 4,2 мс на Radeon X800 XT, что не удовлетворяет ограничению, наложенному на временные затраты при постановке задачи, и требует проведения сортировки частиц на центральном процессоре.

Задача построения полигональных моделей на основе вычисленных координат центров частиц разбивается на два этапа. Необходимо анимировать параметры формы частицы (прозрачность, размер и др.) и сформировать буферы вершин с последующей растеризацией видеокартой содержащихся в них полигональных моделей. Анимацию формы частицы, в силу независимости обрабатываемых данных и небольшого числа выполняемых операций на одну вершину, оптимально разместить на вершинном конвейере видеокарты.

Поскольку графический акселератор и центральный процессор могут производить вычисления одновременно, важно не только распределение вычислительных задач между ними, но и параллельность выполнения этих задач. Порядок выполнения этапов обработки частиц в реализованной системе показан на рис. 1.

Обновление позиций частиц. Как было упомянуто выше, в разработанной системе выбран итеративный способ вычисления позиций частиц на графическом ускорителе. Координаты частицы на текущем кадре вычислялись по координатам предыдущего кадра.

В качестве алгоритма обработки анимации использован подход, предложенный в [1]. Необходимые для анимации движения частиц данные передаются на видеокарту в виде двух текстур, содержащих в своих RGB-каналах XYZ-компоненты векторов координаты и скорости частиц с предыдущей итерации обработки. Каждый канал текстуры представлен

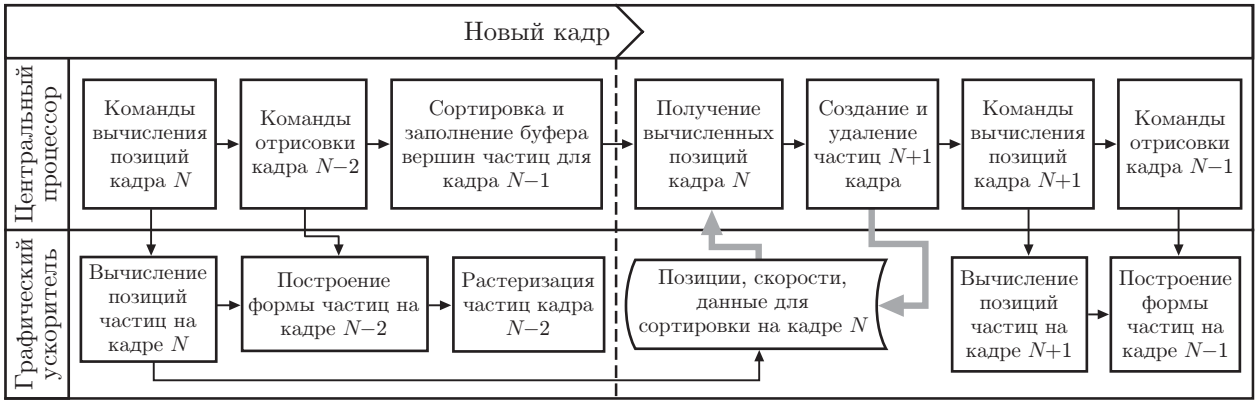

Рис. 1. Порядок выполнения задач анимации частиц

вещественным числом с плавающей точкой для достижения требуемой точности вычислений. Наличие четвертого канала (альфа-канала) в текстуре дает возможность передавать в видеопамять дополнительные параметры. Так, в альфа-канале текстуры позиций размещалось значение силы, действующей на частицу, что позволило достичь более реалистичной анимации, а альфа-канал текстуры скоростей использовался для сохранения времени столкновения частицы с другим объектом виртуального мира. Каждой частице соответствовал пиксель из прямоугольной области текстур координат и скоростей. Каждая область содержала пиксели, хранящие состояние частиц, обладавших одинаковой моделью анимации движения. Обработка частиц достигается растеризацией прямоугольника, занимающего в буфере кадра пиксельную область, равную области текстуры, содержащей информацию о состоянии частицы данного типа. Запись результатов рендеринга производится не в буфер кадра, а в текстуры-приемники данных. Использование технологии MRT (multiple render target) [5] позволило производить рендеринг в несколько текстур одновременно. После очередной итерации работы алгоритма содержимое текстур позиций, скоростей и других данных текущего кадра копируется из видеопамяти в оперативную память. Затем текстуры-источники и текстуры-приемники данных меняются местами. Вследствие аппаратного ограничения на количество инструкций, выполняемых программируемым блоком пиксельного конвейера на фрагмент данных, необходимо реализовывать каждую модель анимации движения частицы (полет в поле гравитации, прямолинейное движение с обработкой столкновений и т. д.) в виде отдельного блока программного кода (пиксельного шейдера), параметризуемого набором чисел (шейдерных констант). Типичными параметрами шейдера анимации движения являются: матрица преобразования из мировой в систему координат наблюдателя, время, прошедшее с последнего обновления состояния частиц, и координаты плоскостей, с которыми рассчитывались столкновения.

Возможность сохранения в память чисел с плавающей точкой позволяет использовать пиксельный конвейер не только для расчета скоростей и позиций частиц, но и для вычисления других величин, таких как z-координата центра частицы в системе координат наблюдателя или время и точка столкновения частицы с каким-либо объектом виртуального мира. Для размещения таких величин использовалась дополнительная четырехканальная текстура.

Анимация формы частиц. После сортировки частиц происходит заполнение вершинных буферов и передача команд на растеризацию графическому ускорителю. Так же как и пиксельный, программируемый вершинный конвейер большого числа видеокарт имеет ограничение на количество операций, выполняемых над одной вершиной (до 256 векторных инструкций для графических ускорителей минимальной поддерживаемой платформы), что делает невозможным написание единственной программы с операторами услов-

ного перехода для всех возможных типов анимаций. Реализация каждой анимационной модели представляет собой отдельную программу — вершинный шейдер. Типичными анимационными параметрами вершин геометрии частицы являются: цвет, прозрачность, ориентация относительно наблюдателя, размер, текстурные координаты, скорость и другие атрибуты. Кроме анимационных параметров необходимы данные для построения самой формы, например матрица преобразования мировой системы координат в систему координат наблюдателя для построения постоянно повернутых к наблюдателю спрайтов [6], применяемых при реализации таких эффектов, как дым или гало вокруг источника света.

Передача данных на графический ускоритель. Анимация формы частицы требует обработки большого количества параметров, процесс изменения каждого из которых задается различными способами, например прозрачность и цвет определяются графиком кусочно-линейной функции, а угол поворота и размер вычисляются по начальному значению и скорости роста. Существует два способа передачи данных в видеокарту для обработки на вершинном конвейере: первый — атрибуты вершины (цвет, текстурные координаты и т. д.), второй — шейдерные константы (набор чисел с плавающей точкой, остающийся неизменным в течение отрисовки набора примитивов видеокартой).

Анимация большинства параметров требует начального значения, по которому рассчитывается текущее. При моделировании природных явлений начальное значение большинства параметров является случайной величиной и может быть разным у частиц, обладающих одинаковой моделью анимации (начальный размер облака пыли, время жизни клуба дыма и т. д.). Передача таких параметров анимации через шейдерные константы приведет к большому числу вызовов процедуры отрисовки примитивов, что негативно скажется на производительности системы [7]. Поэтому в разработанной системе начальные значения анимационных параметров передаются в вершинный шейдер через атрибуты вершин.

Частицы, обладающие одинаковой анимационной моделью формы, также обладают набором общих параметров. Таковыми являются проекционная матрица и матрица наблюдателя, длительность текущего кадра, графики анимации прозрачности и цвета, набор текстурных координат и другие данные. Эти параметры передаются в вершинный шейдер в виде констант, что уменьшает поток данных в видеопамять из системной памяти.

Упаковка вершинных данных. В процессе разработки было выявлено, что операция заполнения вершинного буфера требует неприемлемо много времени. Для уменьшения временных затрат на формирование буфера вершин были использованы два метода. Первый, известный как “pseudo instancing” [8], позволяет уменьшить количество данных, передаваемых с каждой вершиной и необходимых для ее сдвига при анимации полигональной формы частицы. Второй метод сокращения объема передаваемых данных использует снижение точности передаваемых анимационных параметров — чисел с плавающей точкой (скорость вращения, коэффициент линейного расширения и др.).

Каждое число упаковывалось в два байта. В старший байт записывалась целая часть, в младший байт — дробная. Распаковка данных проводилась в вершинном шейдере. Благодаря использованию описанных приемов размер буфера вершин был уменьшен на 30 %, что сократило время его заполнения на целевой платформе в 2 раза (с 5 до 2,5 мс/кадр). Следствием использования упаковки данных стало увеличение времени работы вершинного шейдера анимации формы, составившее для 16000 частиц 0,5 мс/кадр.

Тестирование производительности обработки частиц. В качестве тестовой системы частиц был реализован эмиттер, испускавший от 1 до 100 частиц за кадр. Каждая частица обладала рядом анимируемых параметров: прозрачностью, размером, углом поворота, текстурными координатами. Форма частицы представляла собой четырехугольник, ориентированный к наблюдателю. Моделью движения выбран полет в поле гравитации

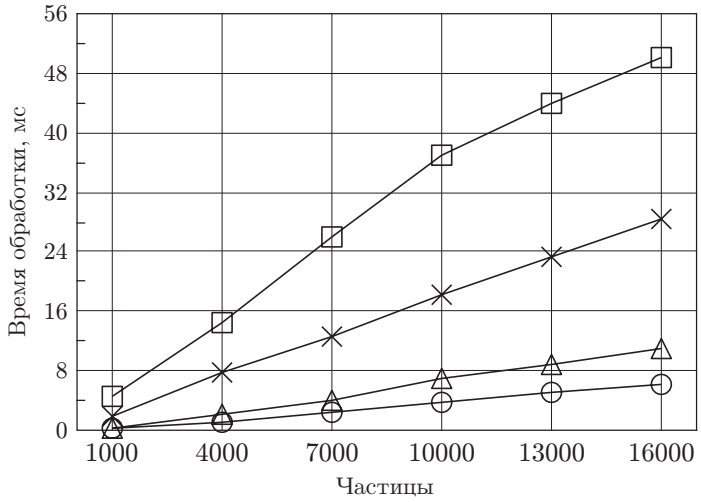

Рис. 2. Производительность анимации частиц. (Круги — смешанная реализация (Athlon X2 4600+, GeForce 8800 GTS), треугольники — смешанная реализация (Pentium 4 2,41 GHz, GeForce 6800 Ultra), крестики — классический подход (Athlon X2 4600+), квадраты — классический подход (Pentium 4 2,41 GHz))

с обработкой столкновений с одной плоскостью. Приведенные характеристики анимационной модели типичны для большинства систем частиц, моделирующих такие эффекты, как дым, огонь, пыль, туман, искры и другие природные явления. Производительность реализации описанной в предлагаемой работе системы, использовавшей смешанный подход к обработке частиц, сравнивалась с производительностью классической реализации системы частиц, осуществлявшей анимацию полностью на центральном процессоре.

Как уже упоминалось, после обработки движения частиц происходит передача рассчитанных на графическом ускорителе скоростей, позиций и данных для сортировки из видеопамяти в системную память. Время выполнения этого этапа зависит от шины данных, на которой работает видеокарта. Установлено, что передача информации занимает 5 мс по шине AGP 4x и 0,136 мс по шине PCI Express x16. Поскольку предполагаемой шиной данных целевой платформы является PCI Express, то такие результаты можно признать удовлетворительными.

Так как видеокарта и центральный процессор производят вычисления параллельно, то за время обработки частиц системой, использовавшей смешанный подход, выбрано максимальное время работы одного из устройств. В ходе тестирования на всех аппаратных платформах время работы центрального процессора превышало время работы видеокарты. Как видно из графика, представленного на рис. 2, использование графического акселератора в качестве математического сопроцессора дает пятикратное увеличение производительности по сравнению с классической реализацией.

Заключение. В данной работе реализована программная система анимации частиц. Для обеспечения ее быстродействия применен новый подход к обработке частиц на графическом акселераторе, заключающийся в анимации движения частиц на пиксельном конвейере, анимации формы на вершинном конвейере видеокарты. Предложен подход к уменьшению объема буфера вершин, содержащего данные для построения и растеризации геометрии частиц путем упаковки чисел с плавающей точкой в два байта и использования технологии “pseudo instancing”.

Предлагаемая система использует технологии, предоставляемые видеокартами, реализующими архитектуру “Shader Model 2.0”, что позволяет ей поддерживать широкий

ряд графических ускорителей.

Областями применения разработанной системы являются приложения виртуальной реальности и компьютерные игры. Ее внедрение в будущие проекты позволит повысить скорость обработки систем частиц и увеличить реалистичность моделируемых природных явлений, таких как дым, огонь, пар, пыль, искры и т. д. Предложенные в работе алгоритмы применимы не только для обработки частиц, но и в других областях компьютерной графики. Например, подход к упаковке данных в вершинном буфере может быть использован при рендеринге произвольных полигональных моделей.

На данный момент основные модули разработанной системы прошли этап отладки и тестирования на различных процессорах и видеокартах. Во время тестирования реализованная система показала высокий уровень производительности. По сравнению с классической реализацией системы частиц на центральном процессоре использование смешанного подхода в предлагаемой работе дало увеличение производительности в 5 раз.

СПИСОК ЛИТЕРАТУРЫ

1. **Latta L.** Building a million particle system // http://www.gamasutra.com/features/20040728/latta_01.shtml, 2004.
2. **Kipfer P., Westermann R.** Improved GPU sorting // GPU gems 2: programming techniques for high-performance graphics and general-purpose computation /Ed. M. Pharr. Boston: Addison-Wesley, 2005. 814 p.
3. **Lander J.** The ocean spray in your face // http://www.double.co.nz/dust/col0798.pdf
4. **Kipfer P., Segal M., Westermann R.** Uber-flow: GPU-based particle engine // Proc. of the ACM SIGGRAPH/EUROGRAPHICS conf. on Graphics hardware. Grenoble, France, 2004. N.Y.: ACM Press, 2004. P. 115–122.
5. **Thibieroz N.** Deferred shading with multiple render targets // ShaderX2: shader programming tips and tricks with DirectX9. Plano: WordWare Publishing, USA. 2004. P. 251–269.
6. **Hicks O'dell.** Screen-aligned particles with minimal vertex buffer locking // ShaderX2: shader programming tips and tricks with DirectX9. Plano: WordWare Publishing, USA. 2003. P. 107–112.
7. **NVIDIA** Corporation. GPU programming guide // http://developer.download.nvidia.com/GPU_Programming_Guide/GPU_Programming_Guide.pdf
8. **Zelsnack J.** GLSL pseudo-instancing // Technical Report. NVIDIA Corp., 2004. ftp://download.nvidia.com/developer/SDK/Individual_Samples/DEMOS/OpenGL/src/glsl_pseudo_instancing/docs/glsl_pseudo_instancing.pdf, 2004.

Поступила в редакцию 10 марта 2009 г.