Растеризационные методики и архитектуры систем визуализации реального времени

С.И. Вяткин, Б.С. Долговесов, В.М. Фомичев
Институт автоматики и электрометрии СО РАН,
Новосибирск, Россия
Центр визуализации и спутниковых информационных
технологий НИИСИ РАН, Москва, Россия
sivser@mail.ru, bsd@iae.nsk.su, vld@niisi.ras.ru

Аннотация

Рассматриваются основные растеризационные методики, мозаичный и немозаичный краевые случаи сцен, способы распараллеливания вычислительного процесса, тенденция применения GPU в системах визуализации реального времени для тренажерных задач и развитие унифицированной архитектуры графических ускорителей с большей универсальностью и гибкостью.

Ключевые слова: *Фрйм-буферный метод, виртуальная методика, PC-базируемые системы визуализации, унифицированный подход.*

1. ВВЕДЕНИЕ

В работе [1] была дана характеристика десяти растеризационным методикам, исследованы методы распараллеливания вычислительного процесса и описаны основные графические примитивы. В компьютерной графике используется несколько типов представления геометрических объектов, каждый из которых в силу своих свойств используется в различных областях от систем 3D - моделирования и CAD систем, до систем визуализации реального времени. Основные достоинства функционального задания [2] это точность представления (такое представление наиболее точно из всех существующих описывает геометрию объекта) и наименьший размер данных, необходимых для описания геометрии объекта. К недостаткам можно отнести сложность геометрической обработки и визуализации в реальном времени. Способы функционального задания демонстрируют компактность, и гибкость задания поверхностей, а также объектов являющихся результатами логических операций над объемами в конструктивной геометрии CSG (constructive solid geometry). Полигональное задание поверхности, по сути, является кусочно-линейной интерполяцией некой поверхности. Достоинства такого задания заключаются в исключительной простоте представления и универсальности. Ведь кусочно-линейное представление существует для любой поверхности. Так же стоит отметить небольшие вычислительные затраты необходимые для визуализации и геометрических преобразований. Недостатки заключаются в больших размерах баз данных необходимых для хранения информации о геометрии поверхности, и ломаная структура силуэта объекта, проявляющаяся при его визуализации. Задание поверхностей на основе сплайнового представления [3], наряду с аналитическим описанием используется в основном в моделирующих системах типа AutoCAD, 3D Studio. Отличается высокой точностью представления как двух, так и трехмерных объектов. В системах визуализации такое представление поверхностей пока не нашло своего применения так как требует очень высоких вычислительных затрат на обработку и визуализацию. Существует так же ряд специфичных способов задания поверхностей, в виде карты высот, в виде набора сечений или контуров и т. д. Такие способы задания используются в основном для решения узко – специализированного круга задач и применяются, в основном, в программах моделирования и проектирования. Воксельные данные применяются для объемной визуализации [4].

Используемые на данный момент алгоритмы генерации синтетических изображений можно разделить на три основных, различающихся между собой типа.

1. Алгоритмы растрирования на плоскости.
2. Алгоритмы трассировки лучей (Ray Tracing).
3. Воксельно - базируемые методы.

Первый тип алгоритмов является самым распространенным на данный момент. Они применяются для генерации изображений в системах реального времени, при этом используют полигональный способ задания объектов сцены. Такие методы получили широкое распространение в силу аппаратной поддержки.

Алгоритмы трассировки лучей используются для создания высококачественных изображений близких по уровню реалистичности к фотографическим изображениям. Основное достоинство такого подхода – высокая реалистичность получаемых изображений, недостаток – время генерации, которое может занимать от нескольких минут до сотен часов.

Классификация воксельно-базируемых методов дана в работе [4]. Существующие акселераторы объемной визуализации, например, VolumePro-500 и VolumePro-1000 применяются в основном в медицинских приложениях для достоверной визуализации томографических данных, а также в геофизике, археологии и промышленности [5].

С постоянным ростом требований к более высокой сложности изображения, а также более высокому разрешению отображаемых сцен в реальном масштабе времени, в работе [1], как уже упоминалось выше, дана классификация и характеристика эффективности основных растеризационных методик, а также возможные пути развития графических архитектур. Для полигональной растровой графики существуют два основных базисных

подхода, это фрейм-буферная и виртуальная методики визуализации.

2. ФРЕЙМ-БУФЕРНЫЙ МЕТОД (FRAME-BUFFER TECHNIQUE)

Фрейм-буферные методы - это методы, предусматривающие хранение и модификацию каждого пикселя изображения в памяти кадра, и растрирующие примитивы в эту память произвольного доступа. С применением данной методики реализованы всем известные графические кристаллы GPU: GeForce (NVIDIA) и Radeon (ATI). К системам визуализации реального времени относится система «Альбатрос» [6, 7], в основу которой легла такая методика, которая была разработана в нашей Лаборатории. Наиболее производительные алгоритмы растрирования разработаны с целью максимального применения когерентности в пространстве изображения, однако в них редко используется возможность эффективного определения видимости и, как следствие, тратится много времени на растрирование невидимой части сцены. В основу архитектуры видеопроцессора системы "Альбатрос" (рис. 1), положена идея рекурсивной процедуры деления экрана [8], локализующей внутреннюю область многоугольников в процессе растрирования.

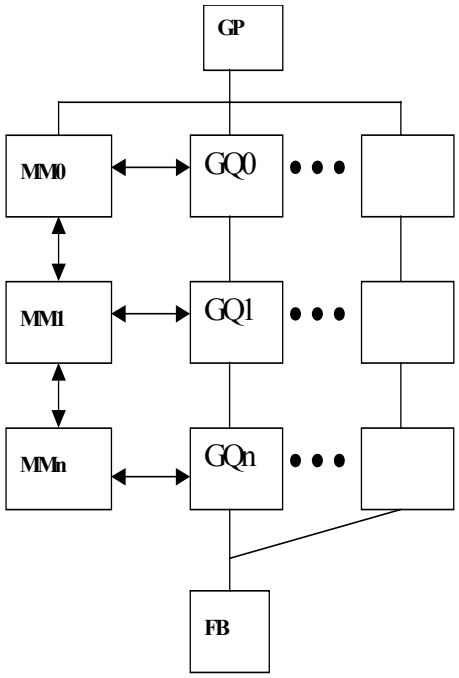

Рис. 1. Конвейеры однотипных клеточных процессоров (GQ) с многоуровневой памятью масок (MM); FB-память кадра, GP- геометрический процессор

Характерными основными чертами метода являются описание многоугольников наборами прямых, проходящих через ребра, и рекурсивное деление экрана на клетки, площадь которых уменьшается в 4 раза с каждым шагом деления. Для устранения дефектов, связанных с дискретностью телевизионного растра, определение принадлежности элемента изображения многоугольнику осуществляется на повышенном разрешении растра: каждый пиксель разбит на 16 субпикселей. В результате формируется субпиксельный код фрагмента, который сравнивается с кодом маски клетки, сформированным из ранее обработанных более приоритетных многоугольников. В коде фрагмента остаются отмеченными только видимые субпикселы, а код маски дополняется вновь замаскированными субпикселями. Память масок модифицируется, а субпиксельный код фрагмента сворачивается в площади пикселей и передается в каналы вычисления цвета. Координаты, цвет и площадь видимой части многоугольника в пикселе поступают в блок видеобуфера, который содержит память для цвета всех пикселей экрана. Результирующий цвет пикселя есть взвешенная сумма цветов видимых фрагментов многоугольников, попавших в пиксель.

Производительность видеопроцессора сильно зависит от характера отображаемой сцены, и различают два краевых случая:

1. Мозаичный вариант, в котором при увеличении количества многоугольников **P** в сцене и уменьшении их площади (**A**-количество пикселей в многоугольнике), глубинная сложность **D** остается постоянной. Время отклика **t** есть число модифицируемых пикселей, умноженное на время цикла **Tcyc** для каждого пикселя, с учетом множителя ускорения **Ssqt** для квадратной фрейм-буферной организации, изложенной выше.

t=N*D*Tcyc/Ssqt, (1)

где **N**-количество пикселей в экране.

2. Немозаичный вариант, в котором при увеличении количества многоугольников и увеличении глубинной сложности, размер многоугольников остается постоянным. Время отклика **t** в этом случае есть количество многоугольников **P** умноженное на их среднюю площадь **A** и время цикла **Tcyc**, также с учетом множителя ускорения **Ssqt**.

t=P*A*Tcyc/Ssqt (2)

Полагая, что сцена покрывает весь экран, состоящий из **N** пикселей, и что каждые многоугольники перекрываются в **D** уровнях, будет справедливо следующее равенство:

P*A=N*D (3)

Средняя высота **H_avg** или ширина **W_avg** многоугольника является хорошим приближением к действительным распределениям площади **A** и пропорциональна:

H_avg=W_avg=√(A)=√(N*D/P) (4)

Одной из трудоемких операций, определяющих количество отображаемых за кадр многоугольников, является преобразование их в пиксели, с одновременным удалением невидимых частей отображаемой обстановки. В некоторых устройствах (например, система визуализации “Аксай”) этот процесс осуществляется с предварительным преобразованием многоугольников в сегменты-отрезки, образованные от пересечения ребер многоугольников со строками телевизионного растра [9], также происходит и обработка примитивов в графических кристаллах GPU: GeForce (NVIDIA) и Radeon (ATI) и др. При таком разложении многоугольников в растр не очень эффективно работают маски, так как они не сформированы в течение большей части кадра и приходится вычислять сегменты для многоугольников или их частей, которые закрыты более близкими к наблюдателю объектами. Процесс преобразования многоугольников в сегменты, а затем в

пиксели осуществляется последовательно, так же последовательно осуществляется и процесс заполнения масок с нижнего уровня на верхний, что замедляет процесс обработки отображаемой обстановки. Исходя из среднестатистической формы отображаемых многоугольников, клетка является более адекватной формой разложения в растр, чем строка (сегмент) растра. Использование многоуровневого клеточного маскирования, реализованного в каждом процессоре конвейера, путем сравнения занимаемых на данном уровне многоугольником клеток с клетками, хранящимися в памяти масок от предыдущих многоугольников, позволяет сократить число вычисляемых клеток, а значит и пикселей. Деление формируемых клеток на пересеченные и внутренние позволяет формировать маски на каждом уровне, а не только последовательно с нижнего уровня на верхний (рис. 1). Использование механизма многоуровневого маскирования позволяет эффективно отбраковывать многоугольники или их фрагменты, попавшие в уже занятые более близкими к наблюдателю многоугольниками области экрана. Тем самым уменьшается время обработки сцены. Отметим также, что зависимость времени обработки от глубинной сложности отображаемой сцены становится при многоуровневом маскировании нелинейной: увеличение глубинной сложности не приводит к существенному увеличению времени обработки сцены. Производительность видеопроцессора в большей степени определяется суммарным периметром многоугольников в сцене. Другой отличительной чертой разработанной нами архитектуры является ее однородность, наращиваемость производительности как полигональной, так и пиксельной без коммутационных связей между конвейерами клеточных процессоров. Несколько систем визуализации “Альбатрос” изготовлено и установлено с 1990 года в ЦПК им. Ю.А. Гагарина для тренировки космонавтов.

3. ВИРТУАЛЬНАЯ БУФЕРНАЯ ТЕХНИКА (VIRTUAL BUFFER TECHNIQUE)

Виртуально-буферные методы - это растеризация примитивов в промежуточные порции экранной памяти и повторное использование этих виртуальных порций для конструирования полного кадра. В семействе систем визуализации реального времени "Ариус" [10, 11], которые были разработаны тоже в нашей Лаборатории, применена виртуальная методика, суть которой заключается в наличии промежуточной памяти фрагментов - памяти спанов (рис.2).

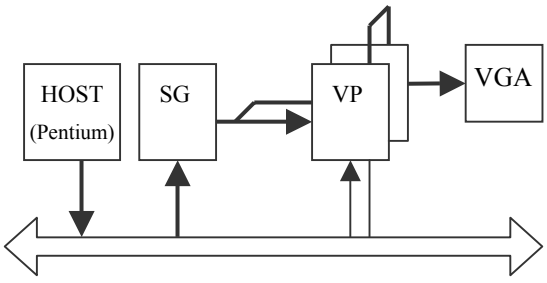

Рис. 2. Структурная схема системы «Ариус»: HOST – центральный процессор, PCI– шина PCI, SG (span generator)– модуль, формирующий промежуточное описание кадра, VP (videoprocessor)– модуль, формирующий изображение, VGA– монитор с VGA-входом

В отличие от большинства систем визуализации, использующих традиционный фрейм-буферный (frame-buffer technique) подход, система «Ариус» построена с применением так называемой виртуальной методики. Отличительными чертами данного метода являются разделение экрана на клетки и конвейеризация вычислений с использованием промежуточного описания кадра в виде списка примитивов.

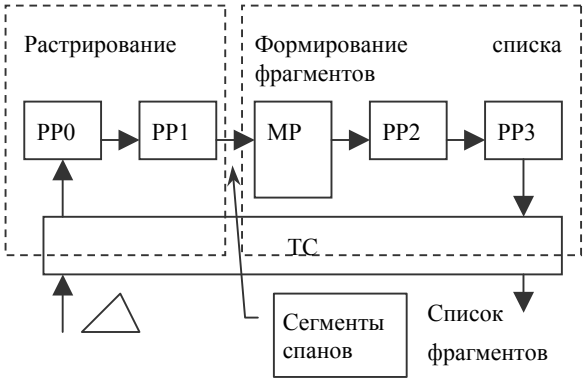

Рис. 3. Модуль SG

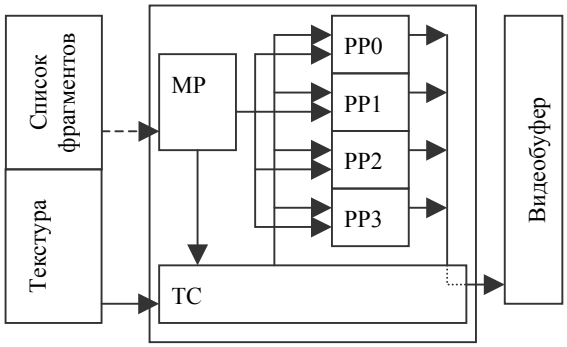

Рис. 4. Модуль VP

В системе «Ариус» изображение формируется из спанов – клеток экрана 8x8 пикселей. Фиксированная сетка спанов позволяет вычислять точные значения параметров отображения граней только в узловых точках. Значения в промежуточных точках восстанавливаются билинейной интерполяцией по четырём узлам. Кроме того, полностью локализуются вычисления, связанные с отображением текстуры. Размер спанов выбран исходя из баланса критериев качества изображения, объёма локальной памяти процессора и вычислительной нагрузки. Разбиение вычислений на две фазы с использованием промежуточного описания кадра позволяет достичь максимальной производительности на этапе пиксельных вычислений, требующих наибольших ресурсов и определяющих производительность системы в целом. При этом удваивается эффективный объём кэш-памяти программ, что расширяет функциональные возможности системы.

Загрузку системы и геометрические преобразования осуществляет host-процессор (Pentium для IBM-совместимых компьютеров). Данные в систему «Ариус» поступают по шине PCI. Модуль SG (span generator) выполняет первый этап

вычислений – преобразование поступающих на вход системы приоритетно упорядоченных геометрических примитивов в промежуточное описание кадра. Примитивами промежуточного описания являются фрагменты пересечения геометрических примитивов со спанами. Второй этап вычислений - обход списка примитивов, определение видимости и цвета пикселей, возложен на модули VP (video processor). Благодаря применению виртуальной методики вычисления в VP легко распараллеливаются. В зависимости от требуемой производительности система может содержать до двух модулей SG и до четырёх модулей VP. На модуль SG возлагаются две основные функции – растрирование геометрических примитивов в сетке спанов и формирование списка фрагментов с вычислением всех необходимых параметров. Для выполнения этих задач организован конвейер (рис. 3) с двойной буферизацией данных. Данные, за исключением экранных координат геометрических примитивов, поступают в виде коэффициентов линейных уравнений. Процессор MP обходит список фрагментов и контролирует раздачу заданий для PP. Процессоры PP0-PP3 работают параллельно (рис. 4). Каждый из них производит пиксельные вычисления в каждом отдельном спане. Параметры отображения фрагментов и данные (текстура, субпиксельные маски) передаются в процессоры PP контроллером TC под управлением процессора MP. Модули SG и VP работают в конвейере, и, следовательно, производительность системы «Ариус» определяется большим из времён работы модулей. Поскольку модули SG и VP работают с разного типа объектами, как минимальные, так и максимальные времена для каждого из модулей определяются различными критериями. Время обработки кадра модулем SG зависит, прежде всего, от количества генерируемых фрагментов, т.е. от суммарной площади граней. Максимальная производительность достигается на больших гранях (из-за выигрыша в инкрементных вычислениях). Использование механизма маскирования приводит к тому, что производительность VP нелинейно зависит от глубинной сложности кадра. Минимальная производительность как для SG, так и для VP наблюдается при мозаичном расположении мелких граней.

Несколько систем визуализации "Ариус" изготовлено и установлено с 1997 г. в ЦПК им. Ю.А. Гагарина для тренировки космонавтов.

С применением виртуальной методики также реализованы система визуализации MaxVue (CAE) и GPU KYRO (STMicroelectronics). Еще эту методику называют тайловой технологией визуализации (Tile Technology).

4. РАСПАРАЛЛЕЛИВАНИЕ ВЫЧИСЛЕНИЙ

В работе [12] показаны известные способы распараллеливания вычислений в системах Reality Engine фирмы Silicon Graphics и Freedom фирмы Evans & Sutherland и др. Два возможных соединения показаны на рисунках 5 и 6. Соединение (кроссбар) *A* используется для маршрутизации (routing) примитивов (треугольников) в соответствующие растеризаторы (рис. 5). Другим возможным способом (соединение *B*) является маршрутизация фрагментов (спанов пикселей) после растеризации в соответствующие устройства (рис. 6). Например, маршрутизация примитивов между геометрическим устройством и растеризаторами используется в системе Reality Engine [13] фирмы Silicon Graphics, а система Freedom фирмы Evans and Sutherland является примером маршрутизации фрагментов.

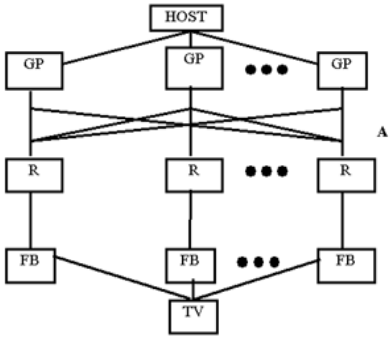

Рис. 5. Общая схема графической части системы Reality Engine фирмы SGI

Маршрутизация примитивов по шине «*A*» может быть охарактеризована как *потенциально узкое место* такой архитектуры. С помощью грубой силы высокая пропускная способность шины может быть, конечно, увеличена еще больше. Однако каждый растеризатор должен принимать и обрабатывать каждый примитив. Распараллеливание в этом месте при данном подходе невозможно, что и устанавливает фундаментальное ограничение масштабируемости систем, использующих маршрутизацию примитивов.

На рисунке 6 (соединение *B*) представлена концептуальная схема графической системы Freedom фирмы Evans & Sutherland [14]. Здесь вместе объединены геометрические преобразования примитива и его растеризация. Несколько модулей работают над несколькими примитивами параллельно. Получившиеся фрагменты маршрутизируются в модули буфера кадра посредством так называемого соединителя буферов, реализованного в виде коммутатора. Если необходимо увеличить производительность с помощью дальнейшего распараллеливания, то очевидными *узкими местами* станут параллельный доступ к модулям буфера кадра и маршрутизация. Часто будет происходить так, что два или больше растеризатора захотят послать данные в один и тот же модуль буфера кадра. Чтобы избежать замедления системы в подобных случаях, коммутирующая сеть должна обеспечивать некоторые средства буферизации данных. Затраты, необходимые для такой коммутирующей сети, растут более чем линейно, что и устанавливает практические ограничения на масштабируемость таких систем.

Альтернативная стратегия распараллеливания может рассматриваться как комбинация независимых графических систем. Выходные данные этих независимых систем, каждая из которых визуализирует часть сцены, объединяются в смешивающем конвейере или дереве, где видимость пикселей определяется с частотой их поступления. Преимуществом такой архитектуры является то, что расширение системы возможно без возникновения "узкого места". По сути, мы имеем здесь идеальный случай, когда производительность увеличивается линейно с числом используемых столбцов. Однако *узким горлом* становится уже место слияния (composition) информации в пикселе. В качестве примера можно привести идеологию PixelFlow [15]. Отличие от архитектуры конкатенации (concatenation) состоит в том, что параллельно обрабатываются не части

экрана, а примитивы. Для каждого пикселя список всех фрагментов обрабатывается и сортируется в конвейере обработки списков.

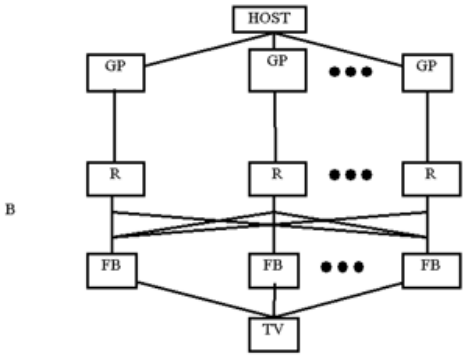

Рис. 6. Общая схема графической части системы Freedom фирмы E&S

4.1 Распараллеливание вычислений в виртуальной методике

В виртуальной методике снижаются требования к пропускной способности шин для подкачки текстурных карт и передачи пиксельного потока в видеобуфер, поскольку в память кадра происходит только запись готовых порций кадра (рис. 7), без модификации последней, а также уменьшается количество вычислений, за счет применения итерационных вычислений пошаговыми приращениями и эффективного механизма маскирования. В системах базируемых на виртуальной методике изображение формируется из спанов (span) – клеток экрана nxn пикселей. Фиксированная сетка спанов позволяет вычислять точные значения параметров отображения граней только в узловых точках. Значения в промежуточных точках восстанавливаются билинейной интерполяцией по четырём узлам. Кроме того, полностью локализуются вычисления, связанные с отображением текстуры. Размер спанов выбирается исходя из баланса критериев качества изображения, объёма локальной памяти процессора и вычислительной нагрузки. Ниже показаны разработанные нами подходы к распараллеливанию вычислений для данной методики. На рисунках 7, 8 и 9 показаны следующие обозначения.

FG - Fragment generator/Генератор фрагментов;
FL - Fragment list/Память-список фрагментов;
DM - Data memory/Кадровая память данных;
PG - Pixel generator/Генератор пикселей;
Shader - Вычислитель цвета;
SpP - Subpixel processor/Субпиксельный процессор;
SpM - Subpixel memory/Субпиксельная память;
VM - Video memory/Видеопамять.

Распараллеливание вычислений можно организовать с помощью двух подходов. Первый вариант это распределение одного фрагмента по нескольким процессорам (рис. 8), (табл. 1). Второй вариант это закрепление одного фрагмента за одним процессором (рис. 9), (табл. 1).

В таблице 1 приведены преимущества и недостатки каждого варианта распараллеливания вычислений.

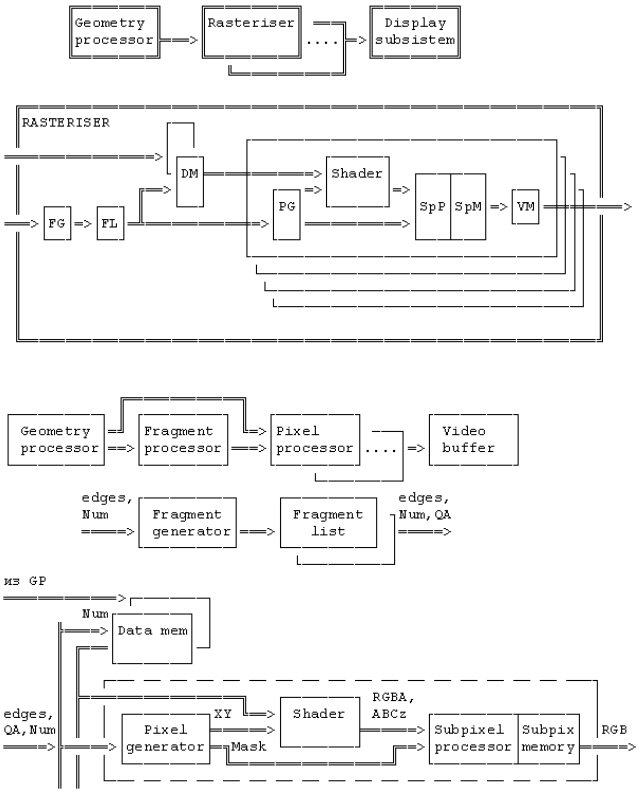

Рис. 7. Общая схема графической части системы виртуальной методики

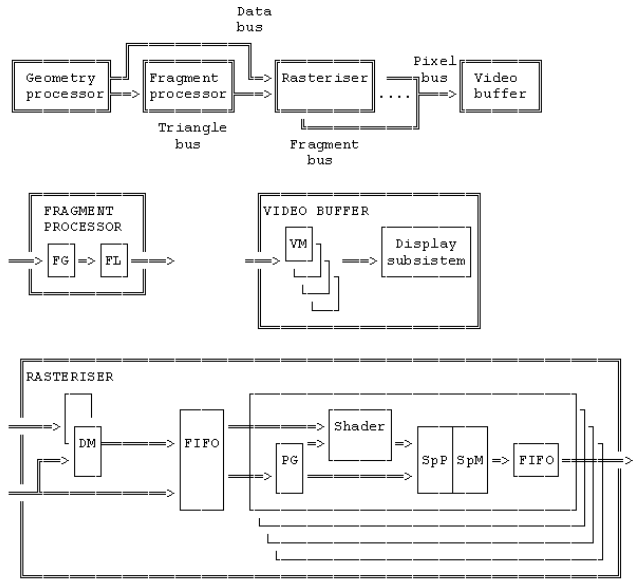

Рис. 8. Распределение одного фрагмента по нескольким процессорам

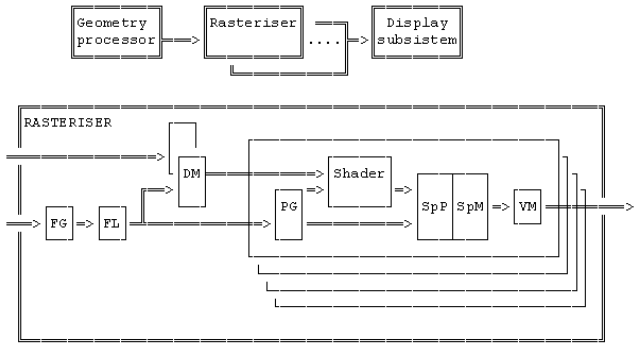

Рис. 9. Закрепление одного фрагмента за одним процессором

Таблица 1. Сравнение двух подходов

Распределение фрагментов по процессорам	1 фрагмент на M процессоров	1 фрагмент на 1 процессор
Работа процессоров	Синхронная по входу	Асинхронная
Распределение банков видеобуфера	Закрепленные за процессорами	Возможно незакрепленные за процессорами
Адресация памяти фрагментов	Поддержка 1-го списка	Поддержка M списков
Память одного субпиксельного процессора	n*n*Nsub*L / M (Nsub- количество субпикселей в пикселе, L - длина слова на субпиксел)	n*n*Nsub*L (Nsub- количество субпикселей в пикселе, L - длина слова на субпиксел)
Эффективность загрузки процессоров	при M/(n*n) = 1/4 около 70%	при M/(n*n) = 1/4 около 100%
Модульность	Модульность ограничена. Рост производительности зависит от M/(n*n)	Произвольное кол-во процессоров с линейным ростом производительности (при M/(n*n)<=1/4)

5. PC-БАЗИРУЕМЫЕ ТРЕНАЖЕРНЫЕ СИСТЕМЫ ВИЗУАЛИЗАЦИИ РЕАЛЬНОГО ВРЕМЕНИ

В современных системах визуализации таких фирм как Evans and Sutherland, CAE и др. существенно повышен реализм отображаемых сцен за счет повышения производительности систем, и за счет воспроизведения различных визуальных эффектов (различного вида текстура, газ, дым, дождь, снег и др.). Наряду со снижением сложности и стоимости систем, происходит расширение функциональных возможностей, вот основные из них:

1. Многоканальность (угол обзора больше 180 градусов – для военной авиации).
2. Масштабируемость (наращиваемость производительности как пиксельной, так и полигональной для каждого канала).
3. Динамическая коррекция дисторсии (необходима только для военной авиации) [16].
4. Транспортная задержка: не более 60 мсек при 60 Гц. Она также зависит от типа самолета, например, для истребителей она не должна превышать 20 мсек.
5. Дисплейный формат: растровый и каллиграфический (необходим только для гражданской авиации).

Современная тенденция – повсеместное применение чипов GPU GeForce и Radeon (в мультипроцессорном исполнении) от NVIDIA и ATI в генераторах изображений EPX и Medallion-S компаний Evans & Sutherland (США) и CAE (Канада), которые исторически разрабатывали свои собственные графические процессоры. Так система EPX-500 базируется на simFUSION (E&S) PC-генераторе изображений, с применением стандартных PC материнских плат в индустриальном шасси с большой масштабируемостью и применением ATI RADEON графических процессоров.

Генератор изображений Fabriano XL10 от Primary Image (Великобритания) реализован на базе GeForce (nVidia Geforce 7800 series chipset).

Компания ORAD (Израиль) масштабирует свою систему несколькими платами графических акселераторов с GPU GeForce и Radeon с помощью плат Compositor собственной разработки (compositing graphic with the DVG). Таким образом, можно увеличивать производительность одного канала отображения.

Система Independence от Quantum 3D реализована на базе QuadroFX. Для максимальной модульности конфигураций, в IDX 2500 применяется “building block” подход, который обеспечивает гибкость системы.

6. УНИФИЦИРОВАННЫЙ ПОДХОД К АРХИТЕКТУРЕ GPU

Эволюция развития GPU началась с относительно простых устройств, вроде GeForce 256, обладавших довольно ограниченным набором фиксированных возможностей. Такие устройства еще нельзя было назвать графическими процессорами в полном смысле этого слова, так как они были неспособны к исполнению уникального программного кода. Первым по по-настоящему программируемым графическим процессором стал кристалл NVIDIA GeForce 3 (NV20), способный исполнять пиксельные и вершинные шейдеры стандарта DirectX 8.0. До недавнего времени всем GPU было присуще одно фундаментальное ограничение: деление исполнительных устройств на пиксельные и вершинные шейдеры. Соответственно, любой графический процессор содержал в своем составе два отдельных набора блоков для обработки каждого вида шейдеров.

Однако, такое разделение на вершинные и пиксельные шейдеры, хотя и имело ряд плюсов, негативно сказывалось на общей эффективности работы GPU. Ведь в сценах, насыщенных пиксельными шейдерами, производительности имеющихся в наличии пиксельных процессоров могло не хватать. В то время как вычислительные мощности вершинных процессоров простаивали впустую, и наоборот

(см. 2 главу данной работы: мозаичный и немозаичный случаи сцен).

Следующий шаг в развитии GPU, это решение проблемы дисбаланса с помощью унификации шейдерных процессоров, при которой нагрузка между ними могла бы распределяться динамически, в зависимости от особенностей рассчитываемой в конкретный момент времени сцены. Новая разработка NVIDIA, графический процессор GeForce 8800 (G80), воплотил в себе эту концепцию. GPU состоит из восьми универсальных вычислительных блоков (шейдерных процессоров), в каждом из которых сгруппированы 4 TMU и 16 ALU. Всего, таким образом, имеется 128 ALU и 32 TMU, но гранулярность исполнения составляет 8 блоков, каждый из которых в один момент может выполнять свои функции, например, исполнять часть вершинного, или пиксельного, или геометрического шейдера над блоком из 32 пикселей (или блоком из соответствующего числа вершин и других примитивов). Все ветвления, переходы, условия и т.д. применяются целиком к одному блоку. Кроме управляющего блока и восьми вычислительных шейдерных процессоров в наличии еще шесть блоков ROP, исполняющих определение видимости, запись в буфер кадра и MSAA, сгруппированные с контроллерами памяти, очередями записи и КЭШем второго уровня. Результат унификации – это разработка полностью универсального потокового процессора [17].

В сумме эти новшества делают возможным полностью аппаратную реализацию шейдерами ранее недоступных алгоритмов, таких как гладкие поверхности с рекурсивным разбиением, трассировка лучей, трехмерный морфинг, а также физические задачи, например, столкновение объектов и т.д.

7. ЗАКЛЮЧЕНИЕ

Итак, рассмотрены основные растеризационные методики, мозаичный и немозаичный краевые случаи сцен, способы распараллеливания вычислительного процесса, тенденция применения GPU в системах визуализации реального времени для тренажерных задач и развитие унифицированной архитектуры графических ускорителей. В системах визуализации реального времени используется, как правило, приоритетный порядок обработки примитивов. Типичная система визуализации реального времени разделяется на два функционально независимых блока: геометрический процессор и видеопроцессор. Такое разделение не случайно и основано на существенном различии их функций. Функции генератора изображения при генерации сцены, можно выделить в следующие категории: создание (постановка) сцены, приоритезация, геометрические преобразования и видеопреобразование. Подвижные объекты в базе данных осложняют обстановку, поскольку они должны быть приоритезированы не только по отношению друг к другу, но и по отношению к фиксированным объектам, поэтому существует ограничение на число подвижных объектов. Z-буфер имеет преимущества в разрешении приоритетных проблем, но имеет свои недостатки. Как и при обратно-приоритетном порядке записи каждый элемент изображения (пиксель) должен быть вычислен, даже если он и невидим. Буфер на кадр должен быть увеличен для хранения функции расстояния. Но более серьезным недостатком является проблема с фильтрацией и полупрозрачностью (для точного вычисления полупрозрачности необходим обратно-приоритетный поток граней). В связи с этим в Лаборатории синтезирующих систем визуализации ИАиЭ СО РАН был разработан многоуровневый Z-буфер (MLZB-Multi Level Z-Buffer), который решает проблемы большой глубинной сложности и полупрозрачности. Аппаратно были реализованы две методики с многоуровневым маскированием. В таблице 2 приведено сравнение фрейм-буферной и виртуальной методик.

Таблица 2. Сравнение двух методик

Основные характеристики	Фрейм-буферные методики	Виртуальные методики
Количество полигонов	Не ограничено	Ограничено
Транспортная задержка	Меньше	Больше
Модификация буфера кадра	Чтение-Запись	Запись
Гибкость конфигурации	Меньше	Больше

Необходимо также отметить, что в системе визуализации «Альбатрос» сбалансирована обработка, как мозаичных сцен, так и немозаичных. Для мозаичных сцен эффективно работают конвейеры процессоров, параллельно обрабатывая несколько небольших полигонов. Для немозаичных сцен, проблему большой глубинной сложности решают несколько видов маскирования. На базе архитектуры «Альбатрос» в 1992 году был разработан GPU с рабочим названием PixGen, однако, к сожалению, он не был реализован. Отличительной чертой архитектуры является ее однородность, наращиваемость производительности как полигональной, так и пиксельной без коммутационных связей, о которых было сказано в 4 главе. В работе [18] дан хороший анализ архитектур с рекурсивным поиском пикселей.

В Институте автоматики и электрометрии были созданы следующие системы визуализации реального времени: «Горизонт» (1976-1979 г.), данная система разрабатывалась для тренажеров морского базирования; «Аксай» (1980-1985 г.), которая разрабатывалась для проекта «Буран»; «Альбатрос» (1985-1990 г.), данные системы визуализации работали с 1990 года в ЦПК им. Ю.А. Гагарина для тренировки космонавтов по стыковке грузовых кораблей к космической станции «Мир», такая же система была установлена на одной из боевых единиц ВМФ РФ для тренировки летного персонала; «Ариус» (1991-1996 г.)- создан ряд систем для авиационных и космических тренажеров. В настоящее время многоканальная система визуализации для комплексного тренажера международной космической станции (МКС), используемая в Российском государственном научно-исследовательском испытательном центре подготовки космонавтов им. Ю. А. Гагарина, обеспечивает, кроме моделирования внешней окружающей среды, имитацию изображений бортовых средств наблюдения за ориентацией МКС и транспортного корабля (ТК).

Пионерами в создании систем визуализации реального времени были такие страны как США (компании Evans and Sutherland, Singer/Link Flight Simulation, McDonnell Douglas Electronics, General Electric), Франция (компания Thomson-

CSF) и СССР (Институт автоматики и электрометрии), значительно позднее была создана первая канадская система компанией CAE Electronics. Наиболее известными зарубежными системами визуализации были NOVOVIEW и CT-5 (Evans and Sutherland), C-130 (General Electric), Link-Miles Image II (Singer/Link), VITAL III и VITAL IV (McDonnell Douglas) [19]. По классу и характеристикам все системы отечественные и зарубежные того периода были примерно одинаковыми. И если раньше, спонсорами развития технологии являлся ВПК ведущих стран мира, то в настоящее время главным образом – индустрия развлечений.

В ближайшем будущем графические ускорители, идущие по пути все большей универсальности и гибкости и центральные процессоры, идущие по пути все большего параллелизма, будут объединены. Один чип будет содержать в себе набор, возможно, разнокалиберных ядер, как выделенных вычислительных или графических, так и общего назначения.

8. БИБЛИОГРАФИЯ

[1] R.F. Sproull, I.E. Sutherland et al. Characterization of ten rasterization techniques // Comput. Graph.-1989.-23, N3.

[2] http://cis.k.hosei.ac.jp/~F-rep/imp_biblio.html

[3] Методы вычислительной математики. Г. Марчук. Наука М. 1980.

[4] Р. Ягель, «Рендеринг объемов в реальном времени», Открытые системы № 5, 1996. стр. 29-33.

[5] H. Pfister, J. Hardenbergh, J. Knittel, H. Lauer and L. Seiler. The VolumePro Real-Time Ray-Casting System Proceedings of Siggraph '99, pp. 251-260, Los Angeles, CA, August 1999.

[6] А.С. 1522240 СССР. Цифровой генератор изображений / А.И. Богомяков, С.И. Вяткин, Б.С. Долговесов и др. - Опубл. 15.11.89, Бюл. N 42.

[7] Асмус А.Э., Богомяков А.И., Вяткин С.И. и др. Видеопроцессор компьютерной системы визуализации "Альбатрос" // Автометрия. N 6. 1994. С. 39.

[8] Вяткин С.И., Долговесов Б.С., Мазурок Б.С. и др. Эффективный метод растрирования изображений для компьютерных систем визуализации реального времени // Автометрия. N 5. 1993. С. 45.

[9] Буровцев В.А., Власов С.В., Вяткин С.И. и др. Геометрический процессор синтезирующей системы визуализации // Автометрия. - 1986. - N 4.

[10] Великохатный Р.И., Вяткин С.И., Гимаутдинов О.Ю., Чижик С.Е. и др. "Ариус" - семейство 3D графических систем реального времени для PC платформ // Тр. 7-й Междунар. конф. "Графикон-97". Москва, 1997.

[11] Вяткин С.И., Долговесов Б.С., Каипов Н.Р., Чижик С.Е. и др. Архитектурные особенности системы визуализации реального времени на основе сигнальных процессоров // Автометрия. 1999. N1. стр. 110-119.

[12] W. Straber, A. Schilling, G. Knittel," High Performance Graphics Architectures", // Graphicon'95 Proceedings, S. Klimenko et al. (Eds). St-Petersburg 1995.

[13] K. Akeley, "Reality Engine Graphics", SIGGRAPH'93 Conference Proceedings, *Computer Graphics*, Vol. 27, August 1993, pp. 109-116.

[14] Evans & Sutherland, Technical Report 517902-904 AA, Evans & Sutherland Computer Corporation, USA, September 1992.

[15] S. Molnar, J. Eyles, J. Poulton, "PixelFlow: High-Speed Rendering using Image Composition" SIGGRAPH'92 Conference Proceedings, *Computer Graphics*, 1992.

[16] Вяткин С.И., Долговесов Б.С., Чижик С.Е. Коррекция дисторсии в компьютерных системах визуализации // Автометрия. 2001. N6. С.46-61.

[17] Suresh Venkatasubramanian. The Graphics Card as a Stream Computer/http://www.research.att.com/~suresh/papers/mpds/mpds.pdf

[18] Ковалев А.М. Производительность генераторов изображений с рекурсивным поиском пикселов// Автометрия. 1995. N1. С.11-20.

[19] Computer Image Generation. Edited by Bruce J.Schachter. A John Wiely&Sons, Inc. 1983.

Об авторах

Сергей Иванович Вяткин – к.т.н, ст.н.с. Лаборатории синтезирующих систем визуализации Института автоматики и электрометрии СО РАН. Адрес: Новосибирск, 630090, просп. Коптюга, 1, ИАиЭ СО РАН. Телефон: (383) 333-36-30 E-mail: sivser@mail.ru

Борис Степанович Долговесов – к.т.н, зав. Лабораторией синтезирующих систем визуализации Института автоматики и электрометрии СО РАН. Адрес: Новосибирск, 630090, просп. Коптюга, 1, ИАиЭ СО РАН. Телефон: (383) 333-36-30 E-mail:bsd@iae.nsk.su

Владимир Михайлович Фомичев- к.т.н, ст.н.с. ЦВИСИТ НИИСИ РАН. Адрес: Москва, Нахимовский просп., 36, к. 1, (ЦВИСИТ) НИИСИ РАН. Телефон 8-(495)-124-48-64

Rasterization Techniques and Architectures of Real Time Visual Systems

Abstract

This paper highlights two significant rasterization techniques, tessellated and non-tessellated scenes, high performance graphics architectures, PC-based visual systems and unified approach to GPU.

Keywords: *Frame-buffer technique, virtual technique, PC-based visual systems, stream-processing.*

About the author(s)

Sergei I. Vyatkin (Ph.D.) is a senior scientific researcher of Synthesizing Visualization Systems Laboratory at Institute of Automation and Electrometry SB RAS. His contact email is sivser@mail.ru,

Boris S. Dolgovesov (Ph.D.) is a head of Synthesizing Visualization Systems Laboratory at Institute of Automation and Electrometry SB RAS. His contact email is bsd@iae.nsk.su

Vladimir M. Fomichev (Ph.D.) is a senior scientific researcher of CVSIT NIISI RAS. His contact email is vld@niisi.ras.ru