РОССИЙСКАЯ АКАДЕМИЯ НАУК
СИБИРСКОЕ ОТДЕЛЕНИЕ

АВТОМЕТРИЯ

№ 1 1999

МЕТОДЫ И СРЕДСТВА СИНТЕЗА ВИЗУАЛЬНОЙ ОБСТАНОВКИ

УДК 681.3.06

С. И. Вяткин, О. Ю. Гимаутдинов, Б. С. Долговесов, Н. Р. Каипов, С. Е. Чижик

(Новосибирск)

АРХИТЕКТУРНЫЕ ОСОБЕННОСТИ СИСТЕМЫ ВИЗУАЛИЗАЦИИ РЕАЛЬНОГО ВРЕМЕНИ НА ОСНОВЕ СИГНАЛЬНЫХ ПРОЦЕССОРОВ

Изложены архитектурные особенности системы визуализации реального времени, выполненной на базе сигнальных процессоров фирмы "Texas Instruments". Отмечены достоинства и недостатки архитектуры использованного процессора. Дается обоснование и анализ виртуальной методики, положенной в основу построения системы визуализации. Приведено краткое описание используемых в системе алгоритмов растрирования, маскирования и вычисления цвета. Анализируется производительность системы.

Введение. С ростом производительности вычислительных систем создаются условия для повышения реализма сцен, синтезируемых компьютерными генераторами изображений в реальном времени. Повышение реализма достигается не только усложнением деталей сцен за счет увеличения производительности, но и воспроизведением специальных визуальных эффектов (различного вида текстура, газ, дым, дождь, снег и др.). Значительно расширяются также функциональные возможности систем. Ниже перечислены некоторые требования к современным системам визуализации для тренажеров: многоканальность, повышенная реалистичность и сложность изображения, высокая скорость обновления изображения, возможность имитации индикаторов навигационных систем и подкачки текстуры в реальном времени, коррекция дисторсии при отображении сцен на большие сферические экраны с учетом подвижности наблюдателя и проектора.

Специализированные наборы микросхем [1], являющиеся основой большинства акселераторов трехмерной графики, пока не подходят для решения подобных задач. Несмотря на то что некоторые из них имеют достаточно высокую производительность для работы в реальном времени, возможности их применения в качестве систем визуализации для тренажеров достаточно ограничены. Это связано с тем, что они в первую очередь ориентированы на

другой круг задач – визуализацию в системах проектирования и трехмерные игры.

В данной статье изложен опыт разработки на базе IBM PC системы визуализации реального времени «Ариус» для авиационных и космических тренажеров с применением в качестве основного вычислительного ядра сигнального процессора TMS320C80 фирмы "Texas Instruments". В основу разработки этой системы положены следующие принципы: открытость архитектуры; программируемость на всех уровнях вычислений; однотипность используемых модулей, позволяющая легко изменять конфигурацию системы.

Особенности архитектуры процессора TMS320C80. Процессор TMS320C80 [2] построен по принципу MIMD над общим полем памяти. На кристалле (рис. 1) интегрированы четыре целочисленных сигнальных процессора и один сигнальный процессор с блоком вещественной арифметики. Целочисленный 32-разрядный RISC-процессор PP (параллельный процессор) имеет расширенную систему команд, оптимизированную для мультимедийных приложений [3]. Предусмотрена также возможность выполнения параллельных операций над 8- и 16-разрядными данными. Объем внутренней памяти каждого PP составляет 10 Кбайт, из них кэш инструкций – 2 Кбайт, память данных – 8 Кбайт. Производительность PP до $6 \cdot 10^8$ опер./с.

32-разрядный RISC-процессор MP (мастер-процессор) ориентирован на выполнение интенсивных векторных вычислений с вещественными данными [4]. В число функций MP входит управление остальными устройствами процессора TMS320C80. Объем внутренней памяти процессора составляет 10 Кбайт, из них кэш инструкций – 4 Кбайт, кэш данных – 4 Кбайт, память данных – 2 Кбайт. Производительность MP достигает 120 Мфлопс. Кроме вычислительных структур кристалл содержит коммутатор (crossbar) для параллельного доступа к ОЗУ, контроллер передачи данных TC с 64-разрядной шиной, ориентированный на пакетные передачи многомерных массивов, и видеоконтроллер VC.

Несмотря на то что процессор TMS320C80 разрабатывался специально для мультимедиа-приложений, его архитектура имеет некоторые ограничения, препятствующие эффективному использованию его вычислительных ресурсов в задаче визуализации трехмерных сцен. К наиболее существенным из них следует отнести небольшой объем внутренней памяти процессо-

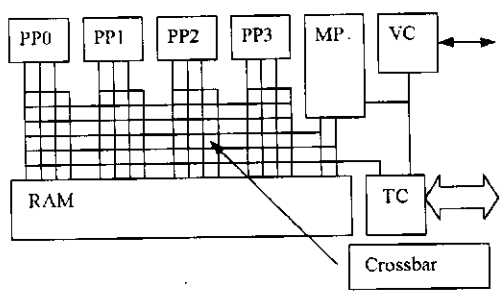

Рис. 1. Структура процессора TMS320C80:
MP – сигнальный процессор с блоком вещественной арифметики, PP0–PP3 – целочисленные сигнальные процессоры, RAM – 50 Кбайт ОЗУ, crossbar – коммутатор для параллельного доступа в ОЗУ, TC – контроллер передачи данных, VC – видеоконтроллер

ра и невозможность оперативного доступа PP к внешней памяти. Первое определяет размер исполняемого кода и как следствие – сложность применяемых алгоритмов. Второе заставляет разрабатывать алгоритмы для PP, оперирующие данными из локальной памяти.

Структура системы «Ариус». Принципы построения системы «Ариус» были выработаны на основе анализа достоинств и недостатков архитектуры процессора TMS320C80. В отличие от большинства систем визуализации, использующих традиционный фрейм-буферный подход, система «Ариус» построена с применением так называемой виртуальной методики. Отличительными чертами данной методики являются разделение экрана на клетки и конвейеризация вычислений с использованием промежуточного описания кадра в виде списка примитивов.

В системе «Ариус» изображение формируется из спанов – клеток экрана 8×8 пикселов. Фиксированная сетка спанов позволяет вычислять точные значения параметров отображения граней только в узловых точках. Значения в промежуточных точках восстанавливаются билинейной интерполяцией по четырем узлам. Кроме того, полностью локализуются вычисления, связанные с отображением текстуры. Размер спанов выбран исходя из баланса критериев качества изображения, объема 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 объем кэш-памяти программ, что расширяет функциональные возможности системы.

Структурная схема системы «Ариус» приведена на рис. 2. Загрузку системы и геометрические преобразования осуществляет host-процессор (Pentium для IBM-совместимых компьютеров). Данные в систему «Ариус» поступают по шине PCI. Модуль спан-генератора SG 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 пикселов) возложен на модули видеопроцессора VP. Благодаря применению виртуальной методики, вычисления в VP легко распараллеливаются. В зависимости от

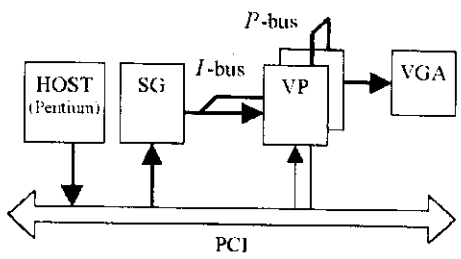

Рис. 2. Структурная схема системы «Ариус»:
HOST – центральный процессор; PCI – шина PCI; SG – модуль, формирующий промежуточное описание кадра; VP – модуль, формирующий изображение; VGA – монитор с VGA-входом

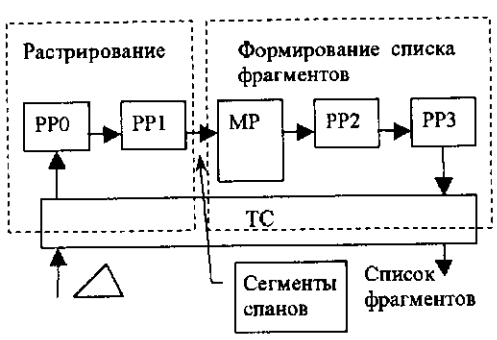

Рис. 3. Модуль SG

требуемой производительности система может содержать до двух модулей SG и до четырех модулей VP.

Модуль SG. 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. Входные параметры, за исключением экранных координат геометрических примитивов, поступают в виде коэффициентов линейных уравнений.

Растрирование. В процессе растрирования геометрические примитивы (треугольники и растровые огни) преобразуются в горизонтальные сегменты спанов, т. е. растрирование производится в масштабе спанов.

Процессор PP0 подготавливает данные для растрирования. В его задачу входит определение габаритов примитива в сетке спанов, классификация ребер треугольников и задание формы огней (кода диаметра и положения в спане). Ребра классифицируются по наклону, признаку левое/правое и вырожденности (ребро укладывается в строку или столбец спанов) – всего девять типов. Для ребер дополнительно вычисляются координаты начала, длина и наклон.

Собственно растрирование примитивов выполняет процессор PP1. Растрирование огней сводится к задаче определения пересечения квадрата с сеткой спанов и в дальнейшем рассматриваться не будет. Обработка треугольника состоит из трех циклов растрирования ребер. Каждый шаг цикла включает определение пересечения ребра с очередным спаном, вычисление границы сегмента, выбор и модификацию соответствующего сегмента. Циклы растрирования для семи из девяти классов ребер состоят из одной инструкции процессора. Во избежание появления разрывов между соседними гранями [5] обход ребер производится в фиксированном направлении – слева направо для горизонтальных и сверху вниз для вертикальных ребер. Принимаются также дополнительные меры для обеспечения замкнутости контура треугольника.

Вычисление параметров фрагментов. В описании геометрических примитивов параметры формы, текстуры и освещенности задаются коэффициентами линейных уравнений. MP, получив список сегментов, вычисляет значения линейных параметров текстуры, тумана и освещенности в узлах сетки спанов по формулам [6]:

$$1/Z(x,y) = A_z x + B_z y + C_z \qquad (1a)$$

$$1/Z(x, y+1) = 1/Z(x, y) + B_z, \tag{1б}$$

$$1/Z(x+1, y) = 1/Z(x, y) + A_z, \tag{1в}$$

$$1/Z(x+1, y+1) = 1/Z(x, y+1) + A_z, \tag{1г}$$

где (x, y) – экранные координаты в масштабе спанов. Затем вычисляются значения текстурных координат и плотности тумана:

$$U(x, y) = (U/Z(x, y))/(1/Z(x, y)), \tag{2а}$$

$$V(x, y) = (V/Z(x, y)/Z(x, y)), \tag{2б}$$

$$F(x, y) = \mathrm{Fog}(Z_0/Z(x, y)), \tag{2в}$$

где Z_0 – дальность видимости тумана, $\mathrm{Fog}(x)$ – табличная функция плотности тумана. Заметим, что непосредственное вычисление параметра (1а) необходимо только для первого (левого) спана в сегменте. Коэффициенты линейных уравнений ребер $Ax + By + C = 0$ приводятся к локальным координатам первого спана сегмента:

$$A_{\mathrm{local}} = A, \tag{3а}$$

$$B_{\mathrm{local}} = B, \tag{3б}$$

$$C_{\mathrm{local}} = C + Ax + By. \tag{3в}$$

Дальнейшая обработка параметров производится в целочисленных процессорах PP2 и PP3. Поэтому полученные результаты округляются и переводятся в формат с фиксированной запятой. Однородность вычислений на этом этапе хорошо согласуется с конвейерной архитектурой MP и позволяет использовать все доступные ресурсы процессора.

В число функций MP также входит поддержание структуры списка фрагментов. В целях экономии памяти описание кадра представляется в виде множества линейных односвязных списков и двумерного массива начальных указателей для списков. Новые фрагменты добавляются в конец списка, так что приоритетный порядок примитивов инвертируется. В зависимости от конфигурации системы список может создаваться для разного количества параллельно работающих модулей VP.

Процессор PP2 подготавливает данные для текстурирования фрагмента. Для вычисления текстурных параметров в первую очередь необходимо определить уровень детальности. В системе «Ариус» текстурные вычисления локализованы, т. е. производятся над небольшим участком карты, загруженным в локальную память. Неточность в определении уровня детальности приводит к переполнению текстурного адреса и возникновению заметных дефектов изображения. Поэтому значение уровня детальности необходимо

вычислять по формуле (4), гарантирующей ограниченный диапазон изменений текстурных координат:

$$\mathrm{Lod} = \mathrm{const}\log_2(\max(U_{\max} - U_{\min}),\ (V_{\max} - V_{\min})), \tag{4}$$

где

$$U_{\min(\max)} = \min(\max)(U(x,y),\ U(x+1,y),\ U(x,y+1),\ U(x+1,y+1));$$

const – постоянный множитель, от которого зависит выбор уровня детальности. Значения текстурных координат в узлах сетки спанов масштабируются в соответствии с уровнем детальности, после чего преобразуются в данные для инкрементных вычислений:

$$U_0 = U(x,y) - U_{\min}, \tag{5а}$$

$$Udx = (U(x+1,y) - U(x,y))/8, \tag{5б}$$

$$Udy = (U(x,y+1) - U(x,y))/8, \tag{5в}$$

$$Uddx = (U(x+1,y+1) - U(x,y+1) - U(x+1,y) + U(x,y))/64. \tag{5г}$$

Смещение участка текстуры, загружаемого в локальную память, вычисляется на основе значений $U_{\min}$ и $V_{\min}$. Из динамической таблицы по номеру текстурной карты определяется адрес в текстурной памяти и размер карты нужного уровня детальности. Определяется также и размер участка требуемой текстурной карты, что минимизирует трафик текстуры:

$$\mathrm{size}U = U_{\max} - U_{\min} + 2, \tag{6а}$$

$$\mathrm{size}V = V_{\max} - V_{\min} + 2. \tag{6б}$$

Остальные параметры фрагмента: плотность тумана и геометрию – подготавливает процессор PP3. Вычисление коэффициентов интерполяции плотности тумана аналогично вычислению текстурных коэффициентов (5а)–(5г). Геометрия фрагмента задается дизъюнкцией субпиксельных масок пересечения каждого из ребер со спаном. Для определения расположения ребра относительно спана анализируются знаки четырех чисел: $\mathrm{sign}(C_{\mathrm{local}})$, $\mathrm{sign}(C_{\mathrm{local}} + A_{\mathrm{local}})$, $\mathrm{sign}(C_{\mathrm{local}} + B_{\mathrm{local}})$, $\mathrm{sign}(C_{\mathrm{local}} + A_{\mathrm{local}} + B_{\mathrm{local}})$, где A_{local}, B_{local}, C_{local} – коэффициенты уравнения ребра в системе координат текущего спана. Разные знаки указывают на пересечение спана ребром. В этом случае из коэффициентов уравнения ребра формируется 15-разрядный код, однозначно определяющий субпиксельную маску заполнения фрагмента. Готовые описания фрагментов передаются в модуль VP, где они запоминаются в двойном входном буфере.

Защита от артефактов. Ввиду того что точные значения параметров вычисляются только в узлах сетки спанов, при отображении фрагментов могут возникнуть дефекты. Наиболее заметные артефакты возникают из-за

неточности интерполяции плотности тумана и в результате накопления ошибки округления при вычислении освещенности для треугольников размером 1–2 пиксела. С целью предотвращения дефектов изображения в каждом спане контролируется градиент соответствующего параметра. Если величина градиента превышает некоторый порог, то значение параметра в спане принимается постоянным и равным среднему значению на грани. Данный метод дает хорошие результаты и практически не влияет на производительность системы.

Модуль VP. В процессоре TMS320C80 модуля VP построена следующая структура (рис. 4). Процессор MP обходит список фрагментов и контролирует распределение заданий для PP. Процессоры PP0–PP3 работают параллельно. Каждый из них производит пиксельные вычисления в отдельном спане. Параметры отображения фрагментов и данные (текстура, субпиксельные маски) передаются в процессоры PP контроллером TC под управлением процессора MP.

Вычисление цвета. В целях сокращения количества вычислений и ускорения передачи текстур в локальную память процессора каждый тексел содержит компоненты цвета двух соседних уровней детальности. Действительно, при таком подходе достаточно вычисления текстурных координат и текстурного адреса только для одного уровня детальности. Поскольку каждый последующий уровень детальности получается фильтрацией предыдущего, ошибка вычислений при таком подходе незначительна. Вычисление текстурных координат параметров производится по формулам:

$$U(0,0)=U_0, \tag{7а}$$

$$U(x,y)=U(x-1,y)+Udx - \text{шаг по строке}, \tag{7б}$$

$$U(x,y)=U(x,y-1)+Udy, \tag{7в}$$

$$Udx=Udx+Uddx - \text{переход на новую строку}. \tag{7г}$$

Целые части U и V составляют адрес текселов в локальной текстурной памяти. Дробные части служат коэффициентами билинейной интерполяции цвета текстуры каждого из уровней детальности в данном пикселе в соответст-

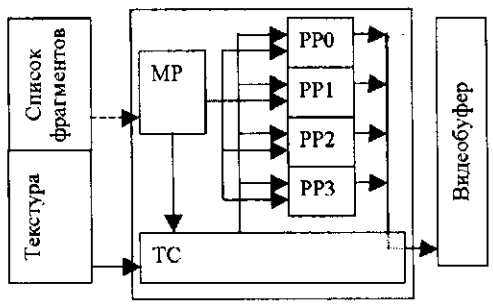

Рис. 4. Модуль VP

вии с выражением (8). Значения (ab), $((1-a)b)$, $(a(1-b))$, $((1-a)(1-b))$ извлекаются из таблицы по адресу, составленному из дробных частей текстурных координат:

$$\text{Color} = C(U,V)(ab) + C(U,V+1)(a(1-b)) +$$

$$+ C(U+1,V)((1-a)b) + C(U+1,V+1)((1-a)(1-b)), \qquad (8)$$

где $C(U,V)$ – значение компоненты цвета из текстуры. Для получения окончательного цвета текстуры интерполируются значения цвета из двух уровней детальности. Вычисление окончательного цвета пиксела с учетом освещенности и тумана производится по формуле

$$\text{Col} = \sum((IS)(1-\text{Fog})\text{Color} + (IS\text{Fog})\text{FogColor}), \qquad (9)$$

где I – освещенность, S – площадь заполнения фрагмента пиксела в субпикселах, Fog – плотность тумана, FogColor – цвет тумана. Суммирование ведется по всем субпикселам, заполнение которых определяется субпиксельной маской фрагмента. Как только все субпикселы данного пиксела оказываются занятыми, пиксел маркируется как занятый, и дальнейшие вычисления цвета в нем не производятся. По мере обработки фрагментов субпиксельная маска в спане накапливается; если фрагмент оказывается полностью занятым, его обработка прекращается. Таким образом, достигается нелинейная зависимость времени обработки кадра от глубинной сложности [7].

Производительность системы «Ариус». Модули SG и VP работают в конвейере, и, следовательно, производительность системы «Ариус» ограничивается самым большим из времен работы модулей. Поскольку модули SG и VP работают с объектами разного типа, минимальные 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 (формулы (1б)–(1г)). Она эквивалентна обработке 3,5 слоев 640 × 480 текстурированных пикселов в кадре за 40 мс. Использование механизма маскирования приводит к тому, что производительность VP нелинейно зависит от глубинной сложности кадра. В конфигурации с двумя VP время формирования кадра с разрешением 640 × 480 и глубин-

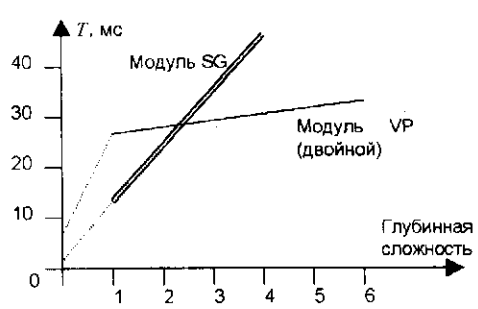

Рис. 5. Производительность системы «Ариус»

Таблица 1

Тип граней	Количество граней за 40 мс	Количество граней за 50 мс
Гуро, туман	2400	4000
Гуро, текстура	1000	2400
Гуро, текстура, туман	1000	2000

Таблица 2

Тип граней	Количество граней за 33 мс	Количество граней за 40 мс
Гуро, туман	3500	5000
Гуро, текстура	1000	2400
Гуро, текстура, туман	1000	2000

ной сложностью 1 составляет 26 мс, а с глубинной сложностью 2 и 3 – 27 мс (рис. 5).

Данные о производительности системы для конфигурации 1SG + 2VP при разрешении 640 × 480 приведены в табл. 1, а расчетная минимальная производительность для максимальной конфигурации системы «Ариус» – в табл. 2. Данные для реальных сцен приведены в табл. 3 (без учета производительности геометрического процессора).

Таблица 3

База данных	Количество граней без текстуры	Количество граней с текстурой	Фрагменты в SG (полное количество)	Фрагменты в VP (отображаемые)	Время обработки кадра, мс
DOS (станция «Мир», панорама)	444	2020	9836	6587	27
DOS (станция «Мир», стыковка)	194	789	22189	10534	41
PORT (ландшафт, панорама)	0	170	9037	7948	30
PORT (аэропорт, панорама)	171	887	9998	8313	36

Заключение. Принципы, положенные в основу системы «Ариус», позволили создать мощную компьютерную систему визуализации. Благодаря наиболее полному и эффективному использованию особенностей архитектуры сигнального процессора, удалось достичь производительности, необходимой для синтеза высокореалистичных изображений в реальном времени. Несомненным преимуществом применения сигнальных процессоров является программируемость системы на всех уровнях. Это позволяет варьировать в некоторых пределах производительность системы и качество изображения, реализовать специфические для тренажеров визуальные эффекты.

СПИСОК ЛИТЕРАТУРЫ

1. **Hill J.** High-end graphics cards // PC Labs Reviews, PC Magazine. June 1998.
2. **MVP** TMS320C80 System-level synopsis. Texas Instruments literature. N SPRU113.
3. **MVP** TMS320C80 PP users guide. Texas Instruments literature. N SPRU110.
4. **MVP** TMS320C80 MP users guide. Texas Instruments literature. N SPRU109.
5. **Bresenham J. E.** Ambiguities in incremental line rastering // IEEE Computer Graphics and Applications. 1987. 7, N 5. P. 31.
6. **Мазурок Б. С., Рожков А. Ф., Сальников Ю. А. и др.** Генерация текстурированных поверхностей и специализированных эффектов в системах «Альбатрос» // Автометрия. 1994. № 6. С. 31.
7. **Асмус А. Э., Богомяков А. И., Вяткин С. И. и др.** Видеопроцессор компьютерной системы визуализации «Альбатрос» // Там же. С. 39.

Поступила в редакцию 28 сентября 1998 г.