АКАДЕМИЯ НАУК СССР
СИБИРСКОЕ ОТДЕЛЕНИЕ
А В Т О М Е Т Р И Я

№ 3 1987

МАШИННАЯ ГРАФИКА

УДК 681.3.06

Э. А. ТАЛНЫКИН
(Новосибирск)

СТРУКТУРА ЛОКАЛЬНОЙ БАЗЫ ДАННЫХ И ОРГАНИЗАЦИЯ ГЕОМЕТРИЧЕСКИХ ПРЕОБРАЗОВАНИЙ В СИНТЕЗИРУЮЩЕЙ СИСТЕМЕ ВИЗУАЛИЗАЦИИ РЕАЛЬНОГО ВРЕМЕНИ

Введение. Настоящая статья продолжает изложение подхода, применявшегося при построении программного обеспечения синтезирующей системы визуализации (ССВ), разработанной в Институте автоматики и электрометрии Сибирского отделения АН СССР в 1981—1983 гг. [1—7].

Общая архитектура аппаратных средств системы рассмотрена в [1]. Нас будет интересовать фрагмент архитектуры ССВ (рис. 1). В данной статье обсуждаются компоненты системы реального времени, размещаемые в процессоре локальной базы данных и клиппирующем процессоре.

Опишем кратко функционирование ССВ. База данных реального времени (БДРВ) хранится на дисках и содержит всю необходимую для визуализации информацию о трехмерной сцене, а также о составляющих ее объектах со всеми уровнями детализации и взаимной подчиненности [7].

Сценарный процессор (СП) имеет доступ к базе данных реального времени через канал массовой памяти и обеспечивает извлечение необходимых для отображения объектов с требуемым уровнем детализации в зависимости от текущего положения активных наблюдателей. Все такие объекты составляют неструктурную часть локальной базы данных. Сценарный процессор управляет работой двух однотипных периферийных процессоров: процессора локальной базы данных (ПЛБД) и клиппирующего процессора (КП). Загрузка программ и данных производится через канал прямого доступа на общей шине, а управление — через программный канал. Основные функции сценарного процессора: поддержка локальной базы данных; синхронизация работы системы; связь с комплексом, моделирующим динамику (либо моделирование динамики); реакция на информационные перегрузки и сбои в аппаратуре ССВ.

Первый из периферийных процессоров хранит локальную базу данных и выполняет единственную функцию: по команде запуска от сценарного процессора генерирует по одному кадру изображения для всех активных наблюдателей и останавливается до следующего запуска. Такая «пассивная» схема работы позволяет легко реализовывать различные схемы синхронизации. Например, программа визуализации базы данных реального времени [4], работающая при поддержке штатной операционной системы, использует де-

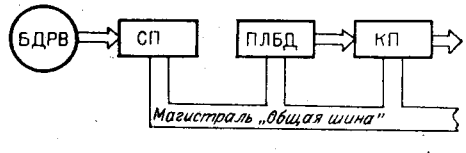

Рис. 1

терминированный алгоритм синхронизации, когда сначала полностью формируется локальная база данных, а затем дается команда на отображение. Система реального времени осуществляет процессы подкачки базы данных и параллельного отображения, причем процесс отображения синхронизуется с полукадровой частотой телевизионной развертки. Процессор локальной базы данных связан прямым каналом с клиппирующим процессором. Генерируемые изображения передаются по каналу в виде приоритетно-упорядоченной последовательности выпуклых многоугольников, заданных в системе координат наблюдателя. Кроме данных, по каналу передаются также команды, обеспечивающие управление процессом клиппирования, переключение наблюдателей и поддержание надежного обмена. Аппаратура канала содержит буфер для сглаживания неравномерностей в работе процессоров.

Клиппирующий процессор, принимая данные от процессора локальной базы данных, обрабатывает их непосредственно «на проходе» (не записывая в память). Обрабатываемые многоугольники клиппируются по границам экрана, подвергаются перспективному преобразованию и передаются в аппаратуру отображения. Программа клиппирующего процессора управляется только через протокол связи с процессором локальной базы данных.

Как отмечалось в [1], в качестве сценарного процессора ССВ используется мини-ЭВМ «Электроника 79», а в качестве периферийных — устройство «Электроника МТ-70М». Конфигурации применяемых периферийных процессоров несколько различаются: процессор локальной базы данных имеет расширенную память данных, а клиппирующий процессор оснащен аппаратным делителем.

Локальная база данных. Основной информационной единицей локальной базы данных является сегмент [7]. Собственно работа процессора локальной базы данных состоит в последовательной обработке необходимых сегментов. Сегмент представляет собой позиционно независимую структуру, которая в неизменном виде переписывается из базы данных реального времени через память сценарного процессора в локальную базу данных. Сценарный процессор занимается динамическим распределением памяти для размещения сегментов. Те сегменты, которые отображаются в настоящий момент, обязаны присутствовать в памяти процессора локальной базы данных. Сегменты, не отображаемые по какой-либо причине (объект «ушел» из поля зрения или сменил детализацию), могут оставаться в памяти для возможного последующего отображения, пока в ней есть свободное место. Сегмент содержит приоритетно-упорядоченный набор граней, так что грани, стоящие впереди, не могут быть закрыты на изображении гранями, следующими за ними по списку, независимо от положения наблюдателя [7].

Каждый объект сцены представляется последовательностью сегментов. При изменении направления, с которого наблюдается объект, может меняться порядок следования сегментов, а при удалении или приближении — их состав (различные по детальности описания объекта). Порядок отображения сегментов устанавливается сценарным процессором на основе анализа базы данных при известном положении наблюдателя. Все сегменты объекта описываются в одной системе координат и не меняют взаимного расположения в пространстве при всех возможных перемещениях. В объекте, т. е. в его системе координат, может быть задан другой объект, двигающийся относительно первого и т. д. Такие объекты называем подвижными, хотя аналогичный механизм используется также для размещения одного и того же фрагмента сцены в различных ее местах без дублирования описаний, путем изменения лишь системы координат погружения (в этом случае система координат объекта не изменяется в процессе работы системы, т. е. объект фактически неподвижен).

Итак, структура сцены в процессоре локальной базы данных представляется системой координат сцены, в которой заданы системы коор-

динат подвижных объектов. В каждой из систем координат могут быть заданы системы координат подчиненных подвижных объектов и т. д., причем глубина вложения произвольна. В каждой из систем координат возможно погружение сегментов, не меняющих своего положения относительно этой системы.

Подвижный объект задается записью, содержащей два поля. Первое поле включает ссылку на запись другого подвижного объекта (объемлющего), в системе координат которого задан первый. Пустая ссылка означает, что объект задан в системе координат сцены. Второе поле содержит матрицу преобразования из системы координат данного объекта в систему координат объемлющего объекта. Здесь и далее используем классическую технику однородных координат, описанную, например, в [8, 9]. Если $P(x, y, z)$ — точка в системе координат объекта, то координаты этой точки в системе координат объемлющего объекта $P(x', y', z')$ получаются применением преобразования

$$(x', y', z', 1) = (x, y, z, 1)T,$$

где T — матрица движения вида

$$T = \begin{pmatrix} t_{11} & t_{12} & t_{13} & 0 \\ t_{21} & t_{22} & t_{23} & 0 \\ t_{31} & t_{32} & l_{33} & 0 \\ a_1 & a_2 & a_3 & 1 \end{pmatrix}.$$

Всякая матрица движения представляется в виде произведения

$$T = \mathrm{rot}(T)\,\mathrm{move}(T),$$

где

$$\mathrm{rot}(T) = \begin{pmatrix} t_{11} & t_{12} & t_{13} & 0 \\ t_{21} & t_{22} & t_{23} & 0 \\ t_{31} & t_{32} & t_{33} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

— матрица поворота, а

$$\mathrm{move}(T) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ a_1 & a_2 & a_3 & 1 \end{pmatrix}$$

— матрица параллельного переноса в повернутой системе координат. Поскольку в процессе обработки локальной базы данных встречаются только матрицы чистого движения и масштабирования, четвертый столбец всех матриц будет постоянным и в действительности в памяти не хранится. Действие матрицы преобразования T на вектор P будем записывать в виде произведения PT, подразумевая запись вектора в однородной системе координат с единичной четвертой компонентой.

В сцене должен быть задан как минимум один наблюдатель. Каждый наблюдатель определяется своей пирамидой видимости и системой координат [8, 9]. Пирамида видимости отсекает часть трехмерного пространства, попадающую в поле изображения, и определяется четырьмя плоскостями: левой, правой, верхней и нижней (рис. 2). Достаточными параметрами для задания пирамиды видимости являются α- и β-половинные углы зрения по горизонтали и вертикали соответственно. Поскольку поле изображения

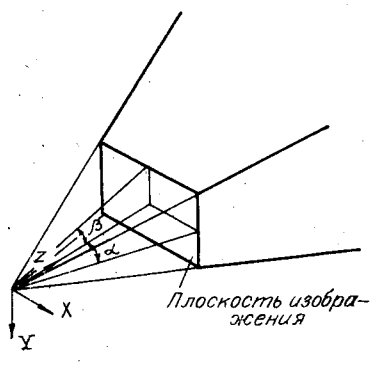

Рис. 2

линейно масштабируется на экран, без потери общности можно считать, что плоскость изображения находится на единичном расстоянии от вершины пирамиды. Система координат наблюдателя задается в вершине пирамиды, как показано на рис. 2. Ось Z направлена по оси пирамиды и задает направление обзора, ось X направлена вправо, а ось Y — вниз.

Всякий наблюдатель должен быть жестко связан с каким-либо подвижным объектом, т. е. пирамида видимости наблюдателя погружена в систему координат объекта в качестве «невидимого» элемента. С объектом может быть связано несколько наблюдателей. При движении объекта изображения, генерируемые для связанных с ним наблюдателей, будут соответственно изменяться.

Наблюдатель в структуре данных задается записью из трех полей. Первое поле содержит ссылку на запись подвижного объекта, в который погружен наблюдатель, второе — матрицу преобразования из системы координат объекта в нормированную систему координат наблюдателя (матрица наблюдателя). Нормировка достигается домножением матрицы преобразования справа на матрицу масштабирования по осям X и Y:

$$S = \begin{pmatrix} \operatorname{ctg}\alpha & 0 & 0 & 0 \\ 0 & \operatorname{ctg}\beta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

где α и β — параметры пирамиды видимости. После такого масштабирования пирамида становится равнобочной, поле изображения определяется простыми соотношениями

$$|X| \leqslant 1,\ |Y| \leqslant 1,\ Z = 1,$$

а уравнения граней пирамиды будут иметь вид

$$Z = Y,\ Z = -Y,\ Z = X,\ Z = -X.$$

Это упрощает алгоритмы клиппирования и делает их независимыми от параметров пирамиды. Третье поле записи наблюдателя содержит положение наблюдателя (вершины пирамиды видимости) в системе координат подвижного объекта. Это поле является избыточным, так как может быть определено по матрице наблюдателя; оно хранится для сокращения вычислений.

Центральное место в структуре данных локальной базы данных занимает дисплейный файл, названный так по аналогии с терминологией традиционных дисплейных систем [8]. Дисплейный файл представляется линейной (векторной) структурой, где отмечена последовательность обработки элементов данных: наблюдателей, подвижных объектов, сегментов. Дисплейный файл генерируется некоторой программой сценарного процессора и только ей, поэтому в его структуре допустима определенная жесткость. Элементы дисплейного файла будем называть командами.

Дисплейный файл обязан начинаться командой определения канала наблюдения

CHANNEL (*наблюдатель*, *логический номер канала*).

Здесь *наблюдатель* представляется ссылкой на запись наблюдателя и соответственно задает положение и параметры пирамиды видимости, для которой будет генерироваться изображение от последующих элементов дисплейного файла. *Логический номер канала* в конечном итоге определяет тот иллюминатор в кабине тренажера или видеоконтрольное устройство, куда попадает генерируемое изображение. *Логический номер канала* — это условное число, которым сопровождается вся относящаяся к данному каналу наблюдения информация. Каждый из блоков видеопреобразования [3] отбирает из общего потока данных только

«свою» информацию (для этого каждый такой блок содержит регистр, где записан его логический номер).

В дисплейном файле может встречаться несколько команд CHANNEL, которые разбивают его на участки по каналам наблюдения. Участки, относящиеся к одному каналу наблюдения, могут быть слиты в один в том порядке, как они следуют в файле.

Команда OBJECT (*подвижный объект*) отмечает, что последующий участок дисплейного файла относится к *подвижному объекту*, ссылка на который имеется в команде, и будет описываться в системе координат этого объекта. Замыкается участок командой OBJEND. Команды OBJECT и OBJEND действуют как скобки и могут быть вложены на достаточную глубину.

Команда SEGMENT (*сегмент*, *фв*, *фн*, *фл*, *фп*) предписывает обработку *сегмента*, заданного ссылкой. Четыре параметра *фв*, *фн*, *фл* и *фп* задают признаки необходимости клиппирования данного сегмента относительно соответствующих плоскостей пирамиды видимости: верхней, нижней, левой и правой. Если заранее известно, что геометрический образ сегмента не пересекает какой-либо плоскости пирамиды, то соответствующий признак устанавливается в нуль. Это экономит вычислительные ресурсы клиппирующего процессора. Признаки устанавливаются сценарным процессором путем анализа положения сфер, охватывающих объекты [7], относительно пирамиды видимости. Для технических целей имеется команда NOP, не предписывающая никаких действий. Команда SKY вызывает генерацию для данного наблюдателя картины звездного неба (см. ниже). Завершается дисплейный файл командой EOF.

В машинном представлении любая команда запаковывается в одно слово данных (38 разрядов), и дисплейный файл представляется вектором. Поскольку дисплейный файл и матрицы подвижных объектов независимо друг от друга и асинхронно загружаются в локальную базу данных из сценарного процессора, то для обеспечения синхронизации имеются по две копии матриц и вектора для размещения дисплейного файла. Сегменты включаются в дисплейный файл, только когда они загружены в память. В качестве единственного параметра для процедуры генерации задается указатель текущего дисплейного файла.

При отсутствии перегрузок в процессорной части системы (до 1000 видимых граней) происходит генерация новых изображений на каждом телевизионном полукадре. При этом матрицы подвижных объектов меняются также с частотой полукадров (50 Гц), обеспечивая максимальную плавность движений, а дисплейный файл фактически изменяется значительно реже. При увеличении загрузки смена изображений начинает производиться на временах, кратных полукадру.

На рис. 3 показан пример представления локальной базы данных, который будет использоваться при дальнейшем изложении. В сцене имеются четыре подвижных объекта, заданных матрицами M_1, M_2, M_3 и M_4. Объект M_3 подчинен объекту M_4, т. е. задан в системе координат последнего. Объекты M_1 и M_2 идентичны, так как описываются одними и теми же сегментами S_8, S_9, S_{11} и S_{12} и имеют по три канала наблюдения. Объект M_4 представлен сегментами S_4, S_5 и S_6, а его подвижная часть (M_3) — сегментами S_2 и S_3. В объекте M_4 и его подвижной части M_3 есть по наблюдателю. Сегменты $S_{13} - S_{18}$, очевидно, представляют неподвижную часть сцены, а сегменты S_1, S_7 и S_{10} загружены в память, но в отображении не участвуют.

Дисплейный файл локальной базы данных на рис. 3 состоит из трех участков. Первый участок описывает вид для одного из наблюдателей на объекте M_1 (это может быть переднее окно кабины), второй — для аналогичного окна на объекте M_2, а третий — для наблюдателя на подвижной части M_3 объекта M_4. Первый участок дисплейного файла содержит объекты M_2, M_4, в который вложен M_3, и неподвижную часть сцены. Отсутствие объекта M_1 в первом участке файла соответствует

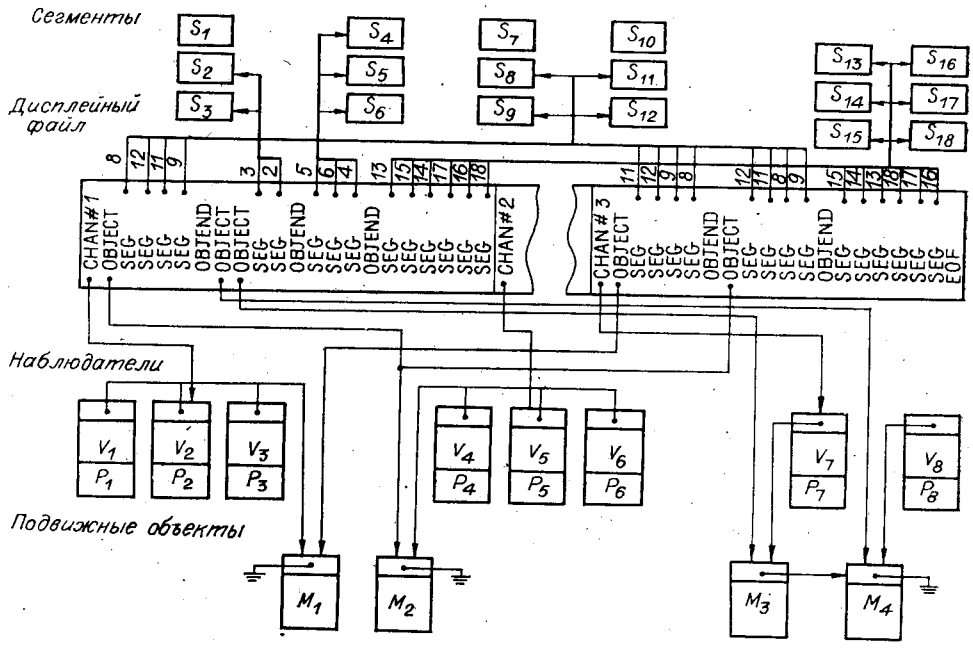

Рис. 3

тому факту, что из окна кабины не виден «свой» объект, хотя в общем случае это возможно. Второй участок (на рисунке не показан) аналогичен первому, только вместо объекта M_2 должен содержать M_1. Третий участок дисплейного файла задает отображение объектов M_1 и M_2, а затем неподвижной части сцены. Ссылки от команд типа SEGMENT на рисунке помечены цифрой с номером сегмента, на который они ссылаются (номера введены только для рисунка). Признаки необходимости клиппирования здесь не показаны. Следует обратить внимание, что сегменты, описывающие один и тот же объект для разных наблюдателей, упорядочены по-разному.

Кроме дисплейного файла и связанных с ним структур, локальная база данных содержит ряд переменных, характеризующих условия освещенности сцены (такие, как коэффициент рассеянного освещения, положение и направление источников освещения и др.).

Геометрические преобразования. Наиболее значительные вычислительные ресурсы в процессе генерации изображения затрачиваются на обработку сегментов, содержащих описания видимых элементов сцены. Принятая система организации данных и преобразований направлена на достижение максимальной эффективности этих массовых операций. Обработка сегмента всегда производится в системе координат объекта, в который он входит. Для этого все глобальные элементы сцены (наблюдатель, источники освещения) переводятся в систему координат объекта. В процессе преобразования координат точки осуществляется единственное умножение вектора на матрицу, что делает прозрачным наличие вложенной системы подвижных объектов. При переходе от одного объекта к другому необходимы дополнительные расчеты, но практически они требуют не больших вычислительных затрат, чем обработка одной дополнительной грани.

Реализация геометрических преобразований базируется на понятии текущего контекста, включающего матрицу преобразования из системы координат обрабатываемого объекта в нормированную систему координат наблюдателя T, вектор положения точки наблюдения в системе координат объекта P, положение и направление источников освещения сцены.

При обработке команды дисплейного файла CHANNEL происходит настройка контекста на указанного в команде наблюдателя для работы

в системе координат сцены. Начальным значением для текущей матрицы преобразования становится матрица из структуры данных указанного наблюдателя. Затем она последовательно домножается слева на обратные матрицы соответствующего наблюдателю подвижного объекта и всех объемлющих его объектов (см. рис. 3). Начальным значением для вектора положения наблюдателя служит соответствующий вектор из структуры данных наблюдателя, который последовательно преобразуется с помощью матрицы подвижного объекта, где размещен наблюдатель, и всех объемлющих его объектов. После обработки первой команды CHANNEL из дисплейного файла на рис. 3 значения для T и P будут $T = M_1^{-1}V_2$, $P = P_2M_1$, после обработки второй команды CHANNEL — $T = M_2^{-1}V_5$, $P = P_5M_2$, а после третьей команды — $T = M_4^{-1}M_3^{-1}V_7$, $P = P_7M_3M_4$. Значения параметров для источников освещения поступают из базы данных, где они хранятся в системе координат сцены.

По команде перехода к новому объекту (OBJECT) текущий контекст «прячется» в стек, а новый контекст получается из текущего по формулам

$$T = M_iT,\ P = PM_i^{-1},$$

где M_i — матрица подвижного объекта. Положения источников освещения преобразуются аналогично P, а направления N — по формуле

$$N = N\,\mathrm{rot}\,(M_i^{-1}).$$

По команде OBJEND текущий контекст перекрывается снятым со стека.

При обработке сегмента каждая из граней, составляющих сегмент, проверяется на ориентацию относительно наблюдателя. Если скалярное произведение $(P - P_1)N$ отрицательно, то грань ориентирована от наблюдателя. Здесь P_1 — вершина грани, N — ее нормаль, P — положение наблюдателя. Таким образом исключаются из дальнейшей обработки «задние» грани объекта. Координаты вершин всех оставшихся граней преобразуются в нормированную систему координат наблюдателя путем умножения на текущую матрицу преобразования T. Для расчета влияния источника света на интенсивность цвета в точке используется формула, предложенная А. М. Ковалевым:

$$I = \frac{1 + \cos\Theta}{2} + k\left(\frac{1 - \cos\Theta}{2}\right),$$

где $\cos\Theta = \mathbf{sN}$; k — коэффициент рассеянного освещения в сцене ($0 \leqslant k \leqslant 1$); $\mathbf{s}$ — единичный вектор на источник освещения; $\mathbf{N}$ — единичный вектор нормали (псевдонормали) в точке. Величина $I (0 \leqslant I \leqslant 1)$ используется как дополнительный коэффициент при расчете интенсивности цвета. Такой метод дает достаточно реалистичные изображения, особенно на затененной стороне объекта, и не противоречит закону Ламберта [9]. Коэффициент k определяет количество рассеянного света, отражаемого объектами сцены.

Организация работы клиппирующего процессора. На вход клиппирующего процессора поступают данные из процессора локальной базы данных (см. рис. 1). Входной поток оформлен в соответствии с протоколом, позволяющим обеспечивать надежность передачи и исправление ошибок, если такие возникают. Используемый канал передачи ориентирован на передачу 38-разрядных чисел (формат слова данных в архитектуре процессоров). Далее будет кратко описан смысл основных сообщений.

1. *Переключение канала.* Устанавливает номер канала отображения, к которому будет относиться последующая информация.

2. *Настройка клиппера.* Устанавливает режим клиппирования относительно тех плоскостей пирамиды видимости, которые заданы в сообщении единичными признаками.

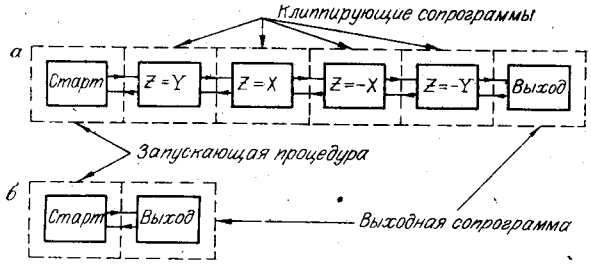

Рис. 4

3. *Грань.* В данном сообщении передается цвет грани и признак запрещения ее отбраковки при перегрузках аппаратуры (в аппаратуре имеется механизм защиты от перегрузок, состоящий в отбраковке незащищенных граней малой площади [2]).

4. *Вершина.* Непосредственно за гранью должны следовать не менее трех вершин. В сообщении этого типа передаются координаты вершины в нормированной системе координат наблюдателя X, Y, Z и интенсивность цвета грани в данной вершине, уже рассчитанная с учетом положения объекта относительно источников освещения.

5. *Строка точечных объектов.* Предписывает генерацию линейки одинаковых точечных объектов (см. ниже). Задаются положение первого объекта, их количество, вектор приращения, диаметр объектов, цвет, интенсивность, функция мигания и признак запрещения отбраковки (как в грани).

6. *Небесная сфера.* Предписывает генерацию на данный канал наблюдения картины звездного неба. Передается нормированная матрица текущего наблюдателя (точнее, матрица поворота с масштабированием).

7. *Конец кадра.* Закрывает кадр изображения сразу по всем каналам, которые подключались после предыдущего конца кадра.

Используемый механизм клиппирования организован в виде динамически настраиваемых сопрограмм, каждая из которых написана для клиппирования относительно своей плоскости пирамиды (рис. 4, *а*). Тексты отдельных сопрограмм отличаются только проверкой видимости точки относительно плоскости.

Каждая из клиппирующих сопрограмм выглядит как «главная» программа, обрабатывающая бесконечный поток вершин, разделенных признаком конца грани. Следующая вершина запрашивается через процедуру ввода, а обнаруженные видимые вершины (результат клиппирования) выдаются через процедуру вывода. В результате запрос на ввод попадает в активную точку сопрограммы, находящейся слева по цепочке, а вывод — на сопрограмму справа, причем точные соседи слева и справа неизвестны и определяются при настройке. Активная точка сопрограммы — это точка, где она последний раз оставила управление (т. е. сделала запрос на ввод или вывод).

Клиппирующие сопрограммы коммутируемы в произвольном порядке, так как структуры потоков данных на их входах и выходах совпадают. Для замыкания цепочки имеются две несимметричные сопрограммы, обязательные для настройки в начале и в конце цепочки. Сопрограмма, настраиваемая в начале цепочки (см. рис. 4), обеспечивает внешний интерфейс для всего клиппирующего модуля (запускающая процедура). Она вызывается извне как процедура, имеющая в качестве параметра вершину или признак конца грани. Сопрограмма в конце цепочки (выходная сопрограмма) обеспечивает выходное преобразование отклиппированных вершин

$$(X, Y, Z) \rightarrow \left(\frac{X}{Z}, \frac{Y}{Z}, \frac{1}{Z}\right)$$

и передачу их в аппаратуру непосредственно в вещественном формате. Дальнейшая обработка описана в [2, 3].

Любая из клиппирующих сопрограмм может быть опущена; минимальный вариант цепочки показан на рис. 4, *б*.

Идея организации процесса клиппирования в виде последовательной

обработки потока вершин серией процессоров (фильтров), а также и сам используемый алгоритм предложены Сазерлендом — Ходгманом [10]. Сопрограммная организация позволяет сделать клиппирующий модуль более симметричным и независимым по управлению. Динамическая настройка обеспечивает совершенно прозрачное отключение клиппирующих сопрограмм (в процессе работы не требуется проверки отсутствия сопрограммы в цепочке). Как и в [10], сопрограммы могут быть реентерабельными (использовать один программный модуль), но с целью получения максимальной эффективности мы применили четыре копии, текстуально настроенные каждая на свою грань пирамиды.

Генерация точечных объектов. При отображении сцен в сумеречных или ночных условиях наблюдения на первый план по обеспечению реальности генерируемых изображений выходят точечные светящиеся объекты, например ночные огни или звезды. В базе данных такие элементы сцены кодируются строками (линейками), составленными из однотипных объектов, и размещаются в структуре сегмента наравне с гранями в соответствии с их приоритетом. Светящийся объект характеризуется диаметром, цветом, направленностью свечения, возможным режимом мигания. В аппаратуре системы для отображения таких объектов имеется специальный генератор. Процедуры преобразования точечных объектов, развертки строк и клиппирования тривиальны и обсуждаться здесь не будут.

Для генерации картины звездного неба используется специальная техника. Поскольку состав звезд постоянен, их нецелесообразно хранить в базе данных, обрабатывать и передавать из процессора в процессор, как другие объекты сцены. Звезды хранятся сразу в памяти клиппирующего процессора в системе координат небесной сферы (ось Z направлена на северный полюс мира, X — в точку весеннего равноденствия небесного экватора, а Y дополняет их до правоориентированного репера). Звезда задается единичным вектором направления и яркостью, которые легко рассчитываются из склонения, восхождения и звездной величины, публикуемых в официальных звездных атласах. Кроме того, в клиппирующем процессоре имеется матрица поворота S из системы координат небесной сферы в систему координат сцены. Текущая матрица преобразования для звезд вычисляется как $S\,\mathrm{rot}(T)$, где T — текущая матрица наблюдателя, передаваемая из процессора локальной базы данных. После преобразования вектора направления он клиппируется как точка и, если попадает в изображение, подвергается перспективному преобразованию и выдается в аппаратуру в виде точечного объекта минимального диаметра. Яркости рассчитаны так, чтобы воссоздать реалистичное восприятие звезд различных звездных величин.

Файл звезд вычисляется заранее специальной программой, на вход которой подается машинное представление звездного атласа. В системе реального времени он просто загружается в память клиппирующего процессора. При подготовке файла звезд можно производить определенный отбор: ограничиваться порогом звездной величины или участком небесной сферы, например северным полушарием. Выбор зависит от условий конкретной задачи.

Матрица поворота небесной сферы относительно системы координат сцены может динамически изменяться сценарным процессором.

Заключение. Представленная организация данных и преобразований использовалась в системе реального времени с разделением вычислительных функций между несколькими процессорами. Средняя производительность, которая была достигнута, составила 5 мкс на вершину и 3 мкс на точечный объект на выходе клиппирующего процессора. Применение данного подхода возможно также и в машинной графике в контексте универсальной вычислительной системы.

Работа выполнялась в тесном взаимодействии с С. Л. Ивашиным и А. В. Гусевым, ответственными за реализацию ряда наиболее важных компонент системы.

ЛИТЕРАТУРА

1. **Ковалев А. М., Талныкин Э. А.** Машинный синтез визуальной обстановки // Автометрия.— 1984.— № 4.
2. **Буровцев В. А., Власов С. В., Вяткин С. И. и др.** Геометрический процессор синтезирующей системы визуализации // Автометрия.— 1986.— № 4.
3. **Богданов В. В., Ковалев А. М., Нефедов И. Б. и др.** Канал видеопреобразования синтезирующей системы визуализации // Там же.
4. **Гусев А. В., Ивашин С. Л., Иоффе А. В., Талныкин Э. А.** Программные компоненты синтезирующих систем визуализации // Там же.
5. **Талныкин Э. А.** Внутренний язык для описания визуальных моделей // Автометрия.— 1985.— № 4.
6. **Гусев А. В., Талныкин Э. А.** SDL — язык описания трехмерных сцен в системах динамической машинной графики // Автометрия.— 1986.— № 4.
7. **Гусев А. В., Ивашин С. Л., Талныкин Э. А.** Математические модели сцен в синтезирующих системах визуализации реального времени // Автометрия.— 1985.— № 4.
8. **Newman W. M., Sproull R. F.** Principles of Interactive Computer Graphics.— N. Y.: McGraw-Hill, 1973.
9. **Foley Y. D., van Dam A.** Fundamentals of Interactive Computer Graphics.— Addison — Wesley, 1982.
10. **Sutherland I. E., Hodgman G. W.** Reentrant polygon clipping // CACM.— 1974.— V. 17, N 1.

Поступила в редакцию 29 декабря 1986 г.