МИНИСТЕРСТВО ОБЩЕГО
И ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

П.В.Вельтмандер

МАШИННАЯ ГРАФИКА
(Учебное пособие в 3-х книгах)

Книга 3

АРХИТЕКТУРЫ ГРАФИЧЕСКИХ СИСТЕМ

Новосибирск
1997

УДК 681.3.06
ББК В 185.2я73–1

Вельтмандер П.В. Машинная графика: Учеб. пособие в 3-х кн. Книга 3. Архитектуры графических систем/Новосиб. ун-т. Новосибирск, 1997. 90 с., ил.

ISBN 5–230–13583–2

Данная книга — последняя из книг курса по машинной графике.

Назначение курса — обучение машинной графике студентов физико-технического профиля. Курс ориентирован на две основные категории будущих специалистов:

- разработчики программно-технических средств машинной графики,
- разработчики прикладных пакетов, приближенные к техническим средствам.

Курс разбит на три части, выпущенные в виде отдельных книг:

1. “Вводный курс”.
2. “Основные алгоритмы машинной графики”.
3. “Архитектуры графических систем”.

Рецензент
канд. физ.-мат. наук, С.И. Упольников

ISBN 5–230–13583–2

Оглавление

ВВЕДЕНИЕ

Цель данной, третьей части курса обучения машинной графике, — дать общий обзор архитектур программных и технических средств машинной графики.

Первый раздел носит вводный характер. В нем формулируются общие принципы построения интерактивных систем машинной графики. Анализируются подходы к построению средств вывода, основанные на использовании специализированных графических языков, синтаксических графических расширений существующих алгоритмических языков и процедурных расширений алгоритмических языков. Показывается, что наибольшее распространение получили процедурные расширения а также специализированные графические языки. Завершается первый раздел анализом организации языков диалога.

Во втором разделе:

1. Разъясняются понятия “рабочей станции” и “суперстанции”.

2. Описываются архитектурные решения используемые в рабочих станциях, в том числе показывается, что современные станции состоят из трех основных компонент — видеопамяти, графического процессора (процессоров) и видеоконтроллера. Особое внимание уделяется вопросам организации видеопамяти.

3. Анализируются основные подходы к построению средств формирования изображений (графических процессоров). Показывается, что основными подходами являются:

- использование специализированных, полностью программируемых процессоров,
- использование непрограммируемых графических сопроцессоров, реализующих фиксированный набор функций,
- использование специализированных СБИС для построения графического процессора требуемых возможностей и архитектуры.

Упомянутые основные подходы иллюстрируются на примерах СБИС фирм:

- Texas Instruments — графические программируемые микропроцессоры TMS-34010, TMS-34020;
- National Semiconductor — набор графических СБИС DP-8500, DP-8510, DP-8512, DP-8515;
- Intel — графический сопроцессор Intel 82786 и RISC микропроцессор с включением графического устройства — Intel 860.

5. Рассматриваются конкретные реализации высокоскоростных 3D суперстанций на примере систем POWER IRIS 4D/380 VGX фирмы Silicon Graphics и GS2000 фирмы Stardent.

Третий раздел посвящен вопросам стандартизации в машинной графике. В нем описывается история работ по стандартизации, ее основные цели и задачи, заключающиеся в обеспечении переносимости программного обеспечения. Вводятся и поясняются модели переносимой графической системы. Рассматриваются организация и направления деятельности по стандартизации в машинной графике. Дается классификация стандартов на интерфейсы и протоколы. Приводятся общие описания различных стандартов на интерфейс в области машинной графики:

- Core System,
- Graphical Kernel System (GKS),
- Graphical Kernel System for Three Dimensions (GKS-3D),
- Programmer’s Hierarchical Interactive Graphics System (PHIGS),
- Programmer’s Hierarchical Interactive Graphics System PLUS (PHIGS+), • CGI (Computer Graphics Interface).

Рассматриваются протоколы передачи графических данных, используемые в различных приложениях машинной графики

В четвертом разделе рассматриваются активно формирующиеся в настоящее время подходы к проблеме обеспечения эффективности разработки человеко-машинного интерфейса. Важнейшей предпосылкой для решения этой проблемы является широкое использование современных растровых дисплейных систем и высокоскоростных каналов связи. В числе прочего в этом разделе рассматриваются:

- системы управления окнами (WMS),
- инструментарий создания пользовательского интерфейса,
- системы управления интерфейсом пользователя, (UIMS — User Interface Management Systems), точнее системы проектирования интерфейса пользователя (UIDS — User Interface Development Systems),
- техника "непосредственного манипулирования" (DM — Direct Manipulation).

Приводится пример реализации UIDS/UIMS.

Пятый раздел посвящен еще одной современной проблеме — проблеме визуализации, особенно обострившейся с широким использованием суперкомпьютеров. В этом разделе обсуждается ViSC инициатива в США (Visualization in Scientific Computing) — "Визуализация в научных исследованиях". На эти работы в США предполагается финансирование порядка 1% от всех затрат на машинную графику.

ViSC — инициатива охватывает (интегрирует) машинную графику, обработку изображений, компьютерное зрение, САПР (дизайн), обработку сигналов, пользовательский интерфейс. Серьезная постановка и подходы к решению этой проблемы стали возможны благодаря появлению суперстанций и формулированию новых идей и методов в программном обеспечении.

Приводится пример коммерческой системы визуализации — AVS.

В последнем, шестом разделе излагаются методы и средства верификации и тестирования графических систем. Приводятся результаты тестирования более чем двадцати рабочих станций.

0.1 ИНТЕРАКТИВНЫЕ СИСТЕМЫ МАШИННОЙ ГРАФИКИ

Задача интерактивной системы машинной графики (рис. 0.1.1) при выполнении вывода заключается в преобразовании информации из исходного высокоуровневого представления предметной области в представление команд графических устройств вывода. При выполнении ввода, наоборот, требуется преобразование низкоуровневой информации от физических устройств ввода в высокоуровневую информацию на языке предметной области.

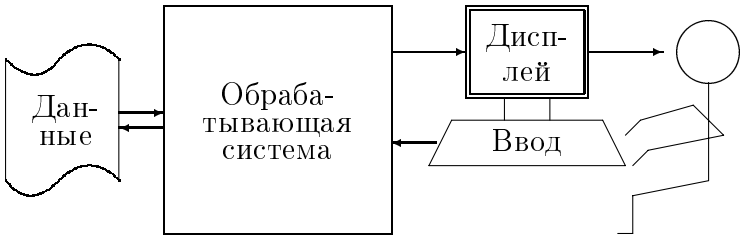

Рис. 0.1.1: Обобщенная модель интерактивной системы

Можно выделить два основных способа построения средств вывода — системы с графическим языком высокого уровня, включающим в себя развитые средства для обработки графической и геометрической информации, и системы с расширенным языком, которые, как правило, представляют тот или иной алгоритмический язык высокого уровня, расширенный средствами обработки графической и геометрической информации. На практике это пакет подпрограмм, реализующих требуемые функциональные возможности.

Ввод информации обеспечивается с помощью языка диалога. Диалог обычно осуществляется в виде команд, содержащих числовые значения, имена, координаты, произвольный текст. Выполняя ввод команд пользователь работает с тем или иным набором вводных устройств, определяемых лексикой языка — алфавитно-цифровой и функциональной клавиатурами, шаровым указателем (track ball), планшетом (tablett) и т.п.

Далее, в п. 1.1–1.3, будут рассмотрены различные варианты построения геометрических языков вывода, в п. 1.4 будут рассмотрены языки диалога.

0.1.1 Графические языки высокого уровня

Имеется два подхода к построению систем программирования с языками машинной геометрии и графики высокого уровня. Первый подход состоит в создании автономного языка, второй — в необходимой модификации того или иного исходного алгоритмического языка.

Первый подход позволяет создать язык, наиболее соответствующий специфике работы с графической и геометрической информацией, но только в том классе приложений, для которых предназначался язык. Исторически основная область приложений таких языков — автоматизация программирования для оборудования с ЧПУ; системы автоматизации проектно-конструкторских работ, требующие средств работы с данными, отсутствующих в широко распространенных алгоритмических языках; системы геометрического моделирования.

Одним из первых проблемно-ориентированных языков, имеющих средства для описания геометрической информации, явился язык APT (AUTOMATED PROGRAMMING TOOLS) [57].

Этот язык послужил основой для разработки разнообразных систем автоматизации программирования для станков с ЧПУ.

В работе [23] предложен базовый проблемно-ориентированный язык описания графической информации ОГРА, предназначенный для описания графических конструкторских документов и операций их формирования в системах автоматизированного проектирования.

В работах [50, 51] описан комплекс САГРАФ, предназначенный для использования в учебной системе коллективного пользования МИФИ для решения задач машинной геометрии и графики. В комплексе имеются подсистемы геометрического синтеза, геометрического анализа и графического моделирования.

В качестве примеров систем с автономным языком высокого уровня могут также служить системы геометрического моделирования трехмерных тел — COMPAC [100, 117] и СИМАК-Д [35].

Система COMPAC (COMPUTER ORIENTED PART CODING) предназначена для формирования описания объемных тел из объемных элементов формы — (метод конструктивной геометрии). Кроме трех базовых объемных элементов (кубы, цилиндры, конусы), могут использоваться профилированные детали, получаемые перемещением замкнутого контура вдоль прямой или дуги, а также тела вращения, получаемые вращением замкнутого контура вокруг оси. Элементы задаются, позиционируются и оразмериваются языковыми конструкциями, напоминающими APT [57]. Составление детали из объемных элементов производится с помощью операций объединения, вычитания и отсечения.

Отличия СИМАК-Д от COMPAC состоят в несколько ином входном языке и ином наборе базовых элементов формы, включающем в себя точку, плоскость, прямоугольный параллелепипед, круговые цилиндры и конус.

* * *

Ясно, что автономные графические языки, как всякая специализированная разработка, обладают высокой эффективностью в своей области приложений, однако разработка и использование таких языков сопряжена с рядом проблем:

- довольно значительные затраты на создание языка и транслятора с него, так, например, трудозатраты на систему COMPAC составили около 40 ч/лет [117];
- затраты на внедрение, на включение языка в работающую систему программирования и на обучение пользователей, которые не всегда охотно берутся за изучение еще одного языка, а предпочитают пользоваться процедурными расширениями известных им алгоритмических языков: ALGOL, FORTRAN, PL-1, PASCAL и т.д.;
- трудности с последующим расширением языка;
- известные в настоящее время языки машинной геометрии и графики, в отличие от процедурных расширений, как правило, не обеспечивают интерактивного режима, а предназначены для написания пассивных программ;
- затруднено объединение в рамках одной прикладной программы графических и геометрических действий и обычных вычислений, которое легко реализуется в случае процедурных расширений.

0.1.2 Синтаксические расширения алгоритмических языков

Ряд из отмеченных выше затруднений снимается при реализации второго подхода — синтаксическом расширении некоторого исходного алгоритмического языка. Удобство этого подхода

заключается в том, что пользователю требуется лишь "доучиться" в уже известном ему языке. Расширение заключается в разработке дополнительных конструкций языка для описания и манипулирования графическими и геометрическими объектами. Возможны несколько способов реализации такого расширенного языка:

- модификация существующего транслятора;
- использование препроцессора, либо макропроцессора, обрабатывающего программу на расширенном языке и превращающего его в программу на исходном языке;
- использование алгоритмического языка, предусматривающего расширение его набора типов данных и операций над ними самим пользователем.

Несмотря на всю неприспособленность FORTRANa для эффективной обработки сложноструктурированных данных, неудобство его структур управления и средств сегментирования программ, большинство реализаций синтаксических расширений было связано именно с FORTRANом из-за его популярности в то время у пользователей.

Одна из первых реализаций графического расширения FORTRANa система GRAF (GRAPHIC ADDITIONS TO FORTRAN) [84]. Расширение основывается на введении нового типа переменных DISPLAY, значениями которых являются последовательности графических команд устройства. Имеются встроенные и пользовательские DISPLAY-функции. Из DISPLAY-переменных и функций строятся выражения этого типа. Оператор DISPLAY-присваивания присваивает вычисленное значение переменной соответствующего типа. Предусмотрены средства стирания и вывода на оконечные устройства дисплейных переменных, чтения/записи дисплейного файла, опроса наличия сигналов внимания с графических устройств ввода.

В качестве более мощного (и позднего) графического расширения FORTRANa можно упомянуть систему GALA [99]. Также вводится новый тип данных — BILD. Значениями этого типа могут обладать переменные и константы. В качестве системных констант этого типа имеются элементарные изображения (в том числе пустое). Над данными этого типа определены операторы преобразований. Для построения сложных изображений используется, так называемый, оператор совмещения. Над изображением определены операторы манипулирования и опроса. Изображения могут проверяться в условных операторах. Имеются операторы вывода изображений с назначением преобразований и окна отсечения. Средства ввода обеспечивают ожидание и получение данных, разрешение и запрет прерываний. GALA-программа может структурироваться на поименованные сегменты с параметрами или без.

В качестве одной из отечественных разработок по графическому расширению FORTRANa можно назвать "АВТОКОД для работы с графическим дисплеем ЕС-7064" [28], при использовании которого обеспечивается пакетный режим работы, когда информация только выводится на дисплей. Операторы АВТОКОДа вставляются в текст на FORTRANe и помечаются в первой позиции буквой А. Предусмотрено 6 групп операторов: резервирования памяти и увязывания с массивами FORTRANa; описания точек и процедур; построения совокупностей точек, ломаных, текстов, вызовов процедур; ветвления; гнездования операторов; фрагментирования изображения на кадры и рисунки.

Наряду с FORTRANом в качестве языка для графического расширения использовался ALGOL. Так в [101] описана система DIGOS, которая состоит из геометрически ориентированного языка, его интерпретатора и трехуровневой структуры данных (PDS — параметрической кольцевой структуры, элементы которой формируются операторами геометрически-ориентированного языка; DDSR3 — структуры данных 3D представления, формируемой из PDS при активации вывода; DDSR2 — структуры данных 2D представления, формируемой из DDSR3 при выпол-

нении преобразований, проецирования и отображения). В языке предусмотрены 2 группы геометрических операндов — простые геометрические элементы (точки, прямые, плоскости, последовательности точек, поверхности, тела) и составные геометрические элементы, образованные либо из элементов одного, либо разных типов. Над геометрическими операндами определены геометрические операции. Графические логические операции служат для проверки значения геометрического операнда и проверки, содержится ли геометрический операнд в некотором другом. Имеются графические операции ввода/вывода и увязывания физико-технических данных с геометрическими операндами.

В качестве примера графического расширения PL-1 можно привести GPL/1 [114], где PL-1 расширен на векторные, двух и трехмерные типы данных и векторные операции. Введены переменные типа IMAGE, которые могут принимать значения, определяемые комбинациями элементов данных изображения вектора, строки, функциями изображения и др. Значения переменных типа IMAGE состоит из двух частей — атрибутной и собственно изображения. Над изображениями определены операции присоединения, объединения, позиционирования, масштабирования, вращения. Введен тип данных GRAPHIC и совокупность атрибутов, описывающих данные этого типа.

Предусмотрено три разновидности элементов данных типа GRAPHIC: DESIGN, из которых строится изображение, все или некоторые элементы которого могут меняться (пример — вывод на дисплей); DISPLAY — аналогично DESIGN, но не может быть изменений (пример — вывод на графопостроитель); STORAGE — для запоминания и последующего воспроизведения изображений.

Оператор GET используется для извлечения изображения из элемента данных, оператор ERASE — для выборочного либо полного стирания. Предусмотрены также средства для ведения очереди ввода и манипулирования ею.

* * *

Анализ вышеприведенных, а также других синтаксических расширений алгоритмических языков показывает, что:

- вводятся данные графических типов;
- определяются операции над ними;
- разрабатываются средства для структуризации, сохранения, манипулирования и отображения изображений;
- устанавливаются правила ведения интерактивной работы.

В целом следует отметить, что подход, основанный на синтаксическом расширении того или иного алгоритмического языка, был характерен для раннего, скорее исследовательского, периода развития машинной графики и имеет следующие основные недостатки:

- требуются значительные затраты труда высококвалифицированных системных программистов как на этапе первоначальной разработки, так и при необходимости расширений;
- из-за использования в определенной мере экзотического языка безусловно нарушается переносимость прикладных программ;
- вводимые в язык графические конструкции не имеют сколько-нибудь серьезных преимуществ перед операторами вызова функций или подпрограмм;
- практически все графические операторы требуют интерпретации, так как они, как правило, программируются как вызовы некоторых библиотечных подпрограмм.

Выше отмеченные недостатки систем с графическим языком высокого уровня привели к тому, что в настоящее время наибольшее применение получили процедурные графические расширения алгоритмических языков, так называемые процедурные языки.

0.1.3 Процедурные графические языки

Процедурные языки — это пакеты графических подпрограмм (графпакеты), доступные из программ на самых различных языках. Процедурные языки особенно удобны для тех приложений, в которых некоторой функции можно поставить в соответствие семантическую подпрограмму. Именно таким приложением и является изобразительная машинная графика, объектами которой являются искусственно созданные изображения. Тесно к этой области приложений прилегает и перцептивная машинная графика, объектами которой являются либо искусственно созданные изображения, либо изображения, выделенные из того или иного представления окружающего мира. Функционально генерацию изображений и манипуляцию с ними естественно представить в виде исполнения команд, использующих имена, координатные и иные данные, характеризующие объект манипулирования. Для выполнения таких действий нет необходимости в наличии графических типов данных, а сами действия удобно представить в соответствующих подпрограммах. Подпрограммы при этом фактически представляют собой мощное и легко модифицируемое семантическое расширение языка. Именно поэтому графические расширения языков за счет создания пакетов графических подпрограмм нашли самую широкую поддержку как со стороны разработчиков системного обеспечения, так и со стороны пользователей. Обрабатывающую систему (см. рис. 0.1.1) можно представить как состоящую из базовой графической системы, осуществляющую чисто графические функции и прикладной программы, осуществляющей необходимые вычисления и управление. Концептуальная модель интерактивной системы при таком подходе представлена на рис. 0.1.2.

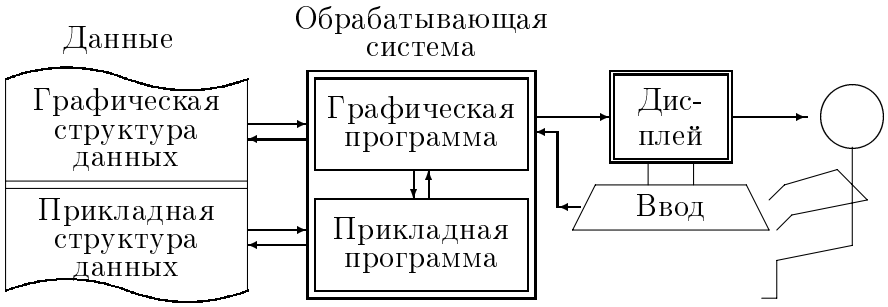

Рис. 0.1.2: Концептуальная модель интерактивной системы машинной графики с явным выделением графической компоненты

Назначение графической системы в такой модели состоит в преобразовании двух или трехмерной модели объекта, формируемой прикладной программой, в графические команды и данные, либо передаваемые на устройства, либо запоминаемые в той или иной графической структуре данных.

На пути использования пакетов графических подпрограмм достигнуты наиболее впечатляющие успехи в использовании и развитии средств машинной графики в том числе связанные с:

- обеспечением аппаратной независимости прикладных программ, т.е. независимости от конкретных используемых графических устройств при условии приблизительного соответствия их возможностей;
- обеспечением адаптируемости прикладных программ, т.е. легкости приспособления к новым функциональным требованиям;

- обеспечением мобильности прикладных программ, т.е. легкости их переноса в иное окружение.

Аппаратная независимость средств отображения обеспечивается их построением по иерархическому принципу в виде некоторого набора логических уровней. Одна из разновидностей иерархического принципа — принцип перевернутой пирамиды [61, 2]. При этом подходе на нижнем уровне располагается аппаратно-зависимый драйвер с минимально допустимым набором возможностей, например, позиционирование, построение отрезка, дуги окружности и т.д. При необходимости использования другого устройства достаточно заменить драйвер. Системы, построенные по принципу перевернутой пирамиды, например, GINO-3D [124], ГРАФОР [3], как правило, ориентированы на работу с несложными устройствами. Использование устройств с более богатыми возможностями влечет за собой либо их неэффективное использование, либо переделку графической системы, т.е. в сущности тормозится развитие средств вывода. В этом заключается основной недостаток таких систем.

Более гибкое и мощное использование иерархического принципа заключается в разбиении средств вывода графпакета на три логических уровня с четкой фиксацией уровней и соблюдением интерфейсов между ними. Каждый уровень является для последующего некоторым виртуальным графическим устройством. Такой подход позволяет безболезненно модифицировать (улучшать) отдельные уровни, легко распределять программное обеспечение между различными ЭВМ, стыковать графическую систему с другими компонентами программного обеспечения.

На первом — нижнем уровне находятся аппаратно-зависимые драйверы устройств (иногда организованные в виде самостоятельных графпакетов [13]), обладающие достаточным набором типичных элементарных функций. Назначение этого уровня — формирование файлов вывода на графические устройства.

На втором — среднем уровне находится аппаратно-независимый графпакет общего назначения.

На третьем — верхнем уровне находятся проблемно-ориентированные графпакеты.

По иерархическому принципу организовано большинство графических систем, наряду с упомянутыми GIND-3D и ГРАФОРом можно назвать, например, СМОГ [43, 44, 45], АТОМ [24, 25, 26, 27, 29, 30, 31, 33], ДИГРАФ [7, 9, 14, 16], ГРАФСМ [40], АНЕГРАФ [19, 47], АТЛАНТ [10], УНИГРАФ [6, 5], ГРАС [37, 38, 41] и целый ряд других.

Адаптируемость прикладных программ в части средств вывода естественным образом обеспечивается последовательным использованием иерархического принципа, т.е. либо заменой требуемого уровня, либо надстройкой необходимых уровней.

Мобильность прикладных программ по отношению к изменению технических средств обеспечивается возможностью использования соответствующих драйверов устройств. Мобильность относительно ЭВМ обеспечивается реализацией функционально идентичных графпакетов для ЭВМ различных типов. Хорошие предпосылки для этого заключаются в решении вопросов стандартизации в машинной графике, которым посвящен раздел 3.

* * *

Основные преимущества подхода с использованием графпакетов состоят в следующем:

1. Легкая обучаемость, так как пользователь пакета не выходит за рамки удобного либо привычного ему языка.
2. Легкая расширяемость за счет написания самим пользователем подпрограмм, необходимых для его приложения. Задача системных программистов состоит в том, чтобы дать “базисный” пакет, пригодный для многих пользователей и многих приложений.

3. Легкая переносимость прикладной программы на другие ЭВМ.
4. Легкая адаптируемость к новым требованиям либо аппаратным возможностям.
5. Легкое обеспечение многоязыковости либо за счет наличия функционально идентичного пакета для другого языка, либо за счет использования одной и той же библиотеки в рамках многоязыковой системы программирования, либо за счет единого исполнительного модуля, доступного с помощью пакетов процедур связи из различных систем программирования.

Очевидными недостатками такого подхода являются:

1. Недостаточная гибкость, связанная с тем, что заранее фиксированный выбор "базисного пакета" со сравнительно ограниченными средствами может значительно усложнить прикладную программу, требующую функций, выходящих за рамки "базисных".
2. Громоздкость, связанная с тем, что простота расширений в сочетании с недостаточной гибкостью провоцирует либо на создание множества подпрограмм по разному выполняющих схожие функции, либо на создание универсальных подпрограмм с большим числом разного рода дополнительных параметров для преодоления ограничений или неэффективностей.

Несмотря на отмеченные недостатки, можно уверенно утверждать, что создание пакетов графических подпрограмм является основным средством разработки графического программного обеспечения.

0.1.4 Языки диалога

Язык диалога, наряду с прочими атрибутами, такими как полнота, точность и скорость решения задачи, является одной из важнейших компонент прикладной интерактивной системы.

Обычно предполагается, что конечным пользователем интерактивной системы является специалист в некоторой предметной области, решающий с помощью ЭВМ требуемую задачу и взаимодействующий с ЭВМ на языке предметной области (входном языке).

Входные языки существенно отличаются от алгоритмических языков как внешне, так и по применению. Фразами такого языка являются изображения и действия. Применение такого языка отличается тем, что входные команды интерпретируются и исполняются по мере их поступления от пользователя, а не транслируются в объектный код и лишь затем исполняются. Наибольшее распространение получили два типа диалога — диалог инициируемый ЭВМ и диалог инициируемый пользователем [52]. Диалог в этих случаях ведется в форме "запрос-ответ". При диалоге 1-го типа пользователь либо заполняет форму, выдаваемую на экран, либо выбирает одну из альтернативных возможностей (команд). При диалоге 2-го типа пользователь подает те или иные допустимые в данный момент времени директивы. Альтернативные языки более просты в обучении и использовании, поскольку не требуют знаний форматов и символики различных директив, действия более просты, так как нет необходимости полностью вводить фразу или директиву и не надо помещать выбранный элемент данных в определенную позицию формата.

Различные аспекты проблемы взаимодействия человек-ЭВМ освещаются во многих работах [11, 12, 17, 20, 46, 53, 54]. Так, например, в [54] сформулированы требования, которым должен удовлетворять диалоговый входной язык:

• эффективность, полнота, естественность;
• расширяемость;
• обеспечение обратной связи;
• устойчивость к ошибкам;
• адаптируемость к пользователю.

В сущности входной язык состоит из двух компонент [54]:
• язык формулирования пользователем команд и входных данных для ЭВМ (реплики человека);
• язык ответов пользователю (реплики ЭВМ).

Реплики человека выражаются в действиях с различными диалоговыми устройствами. Реплики ЭВМ выражаются в графических образах, либо звуковых сообщениях.

Как уже отмечалось, к числу существенных характеристик прикладных программ относятся аппаратная независимость от используемых графических устройств и мобильность — легкость переноса в иное окружение. В пп. 1.1.3. показано, что эти характеристики в части средств вывода — исполнения реплик ЭВМ обеспечиваются использованием виртуальных устройств отображения. Достижение аппаратной независимости при вводе реплик человека также обеспечивается введением понятий виртуальных устройств ввода, являющихся абстракциями реальных физических устройств. Программная поддержка средств ввода строится по иерархическому принципу и организуется в виде пакета подпрограмм. На нижнем уровне находятся драйверы устройств. На верхнем — виртуальные устройства ввода. При этом одно виртуальное устройство ввода может быть реализовано с использованием нескольких физических и наоборот.

В настоящее время общепринятым считается выделение шести классов виртуальных устройств ввода [85, 87, 118, 119]:
• ЛОКАТОР для ввода позиции;
• ШТРИХ для ввода последовательности позиций;
• ДАТЧИК для ввода скалярного значения (числа);
• ВЫБОР для выбора одной из альтернативных возможностей;
• УКАЗКА для указания объекта на изображении;
• КЛАВИАТУРА для ввода строки символов.

В стандарте CGI [66], определяющем интерфейс между аппаратно-независимой и аппаратно-зависимой частями графической системы, дополнительно к перечисленным предлагаются еще два класса виртуальных устройств:
• РАСТР (AREA) для ввода растровых картин;
• ОБЩЕЕ (GENERAL) для ввода иных данных, например, ввод голоса.

Мобильность прикладных программ по отношению к изменению технических средств обеспечивается последовательным использованием только виртуальных устройств. Мобильность по отношению к ЭВМ обеспечивается реализацией функционально идентичных пакетов подпрограмм для ЭВМ различных типов. Хорошие предпосылки для этого заключаются (также, как и для средств вывода) в решении вопросов стандартизации в машинной графике, которые будут рассмотрены в разделе 3.

0.1.5 Выводы

В концептуальном плане выбора архитектуры построения графических систем можно сделать следующие выводы:

1. Построение графической системы на базе специального графического языка целесообразно только при ее массовом специализированном, немодифицируемом применении и при отсутствии необходимости объединения в рамках одной прикладной программы графических и вычислительных модулей.
2. Подход, основанный на геометрическом и графическом синтаксическом расширении того или иного алгоритмического языка, не нашел сколь-нибудь серьезного применения как из-за больших трудозатрат его реализации, так и из-за нарушения переносимости прикладных программ и отсутствия преимуществ в использовании графических конструкций по сравнению с вызовами подпрограмм.
3. Основное и подавляющее распространение получил подход, основанный на процедурных (семантических) графических расширениях алгоритмических языков.

0.2 АРХИТЕКТУРА ГРАФИЧЕСКИХ РАБОЧИХ СТАНЦИЙ

В данном разделе:

1. Разъясняются понятия “рабочей станции” и “суперстанции”.

2. Описываются архитектурные решения, используемые в рабочих станциях.

3. Анализируются основные подходы к построению средств формирования изображений на примерах СБИС фирм:

- Texas Instruments — графические программируемые микропроцессоры TMS-34010, TMS-34020;
- National Semiconductor — набор графических СБИС DP-8500, DP-8510, DP-8512, DP-8515;
- Intel — графический сопроцессор Intel 82786 и RISC микропроцессор с включением графического устройства — Intel 860.

3. Рассматриваются конкретные реализации высокоскоростных 3D суперстанций на примере систем POWER IRIS 4D/380 VGX фирмы Silicon Graphics и GS2000 фирмы Stardent.

Рабочие станции

Рабочие станции (иногда называют “графические станции”) появились в конце 70-х как результат сбалансированного объединения лучших технологий: построение процессоров, работа с графическими объектами и устройствами, организация ввода/вывода, организация связи — в одной системе, удобной для решения инженерных задач.

Многие идеи, появившиеся в 80-е годы, наложили свой отпечаток на рабочие станции, построенные, в основном, из стандартных компонент. Что, в свою очередь, отразилось на производстве компьютеров: увеличение роли ОС Unix, рождение концепции “открытых систем”, разработка новых стратегий производителями компьютеров.

В начале 90-х производительность рабочих станций по многим параметрам приблизилась к большим машинам (mainframe). По закону Джой такой показатель как MIPS удваивается каждые два года с 1 MIPS в 1984 до 64 MIPS в 1990. Наибольшие отличия сегодня можно обнаружить в направлении наиболее быстрого развития — визуализации 3D объектов и росте интерактивных возможностей, — поднимающем производительность труда исследователей и инженеров.

В течении длительного времени было довольно трудно сравнивать производительность рабочих станций, особенно в области графики. Но сейчас ситуация изменилась. В связи с достижением согласия в области стандартов, таких как Unix, X Window, Phigs+ , стало гораздо легче разработать и применить процедуры для оценки производительности рабочих станций. А также, что не менее важно, одинаковым образом проинтерпретировать полученные результаты.

Суперстанции

Суперстанция (superworkstation) — это соединение в одной системе возможностей рабочей станции (3D графика, интегрированность) и суперкомпьютера (быстрый ввод/вывод, векторизация вычислений). В настоящее время на западном рынке представлено около 120-ти моделей рабочих станций и 20-ти моделей X-терминалов. Среди них около 20-ти можно считать суперстанциями, отобранными, например, по производительности при работе с 3D графикой.

Типичную суперстанцию можно описать, как организованную эффективным образом систему из следующих компонент:

- одно или несколько 32/64-битных ЦПУ с кэш-памятью;
- сопроцессоры с плавающей запятой и/или векторный;
- графическая подсистема с процессором, кадровым буфером и Z-буфером;
- не менее чем 32-битная внутренняя шина;
- сетевой контроллер (FDDI, Ethernet Token Ring);
- быстрый дисковый контроллер (IPI, SCSI ...);
- от 16 до 256 мегабайт внутренней памяти;
- стандартная шина ввода/вывода (VME, EISA, MCA ...) для подключения периферийных устройств (диска, магнитофона ...);
- один или несколько асинхронных портов;
- монитор, клавиатура, мышь;
- Unix, X Window, NFS, PHIGS, GKS, C, Fortran, TCP-IP, NCS, эмуляторы графических терминалов, средства отладки ...

Технические характеристики некоторых суперстанций приведены в таблицах 1–4 раздела 6.

Ближайшие перспективы

Рабочие станции развиваются более динамично, чем другие классы компьютеров: рост рынка, снижение цен, рост производительности. Современные достижения в областях: ЦПУ, шины, графика, ОС, диски, ... могут быть использованы в рабочих станциях.

Указанные процессы приведут к тому, что:

- возрастет общая производительность рабочих станций;
- увеличатся периферийные возможности;
- улучшится программное обеспечение;
- улучшатся эргономические характеристики рабочих станций.

0.2.1 Компоненты современных растровых дисплейных систем

Анализируя тенденции развития архитектур растровых графических рабочих станций (ГРС), ориентированных на интерактивную графику можно отметить, что за последние годы сформировалась и стала традиционной архитектура, включающая центральный процессор и растровую графическую дисплейную систему. Центральный процессор выполняет функции обмена информацией между ГРС и внешним миром (базовой ЭВМ или вычислительной сетью), диспетчеризации потоков данных между компонентами рабочей станции и предварительной обработки данных. Растровая графическая дисплейная система осуществляет функции формирования и модификации наборов данных в памяти изображения (видеопамяти) и управляет режимами вывода графической информации на растровый монитор.

Растровая графическая система современной архитектуры состоит из следующих функциональных компонент:

1. Видеопамять служит для хранения графических данных в растровой форме.
2. Графический процессор (либо несколько таких процессоров и, возможно, геометрический процессор) реализует основные функции по формированию изображений в видеопамяти. В современных 2D системах графические процессоры, как правило, выполняют два класса

операций: преобразование графических примитивов в растровую форму (функционально-растровые преобразования) и копирование прямоугольных блоков видеопамяти (растровые операции — Raster Op [79]).

3. Видеоконтроллер формирует управляющие сигналы для организации доступа к видеопамяти со стороны графических процессоров (возможно, и со стороны центрального процессора), а также обеспечивает регенерацию экранного буфера видеопамяти — части видеопамяти, отображаемой на экран монитора. Кроме этого в состав видеоконтроллера, как правило, входит аппаратура управления графическим монитором, схемы таблицы цветности для управления оттенками цветов и градациями яркости изображения и, возможно, средства поддержки ряда атрибутов изображения таких ка, например, мерцание, подсветка, наложение и т.п.

0.2.2 Видеопамять

В растровых дисплейных системах видеопамять организована в виде прямоугольного массива точек. Элемент видеопамяти, стоящий на пересечении конкретных строки и столбца видеопамяти, хранит значение яркости и/или цвета соответствующей точки. Отображаемая на экране часть видеопамяти называется экранным буфером (буфером регенерации или экранной битовой картой). Регенерация изображения осуществляется последовательным построчным сканированием экранного буфера.

Так как каждый элемент видеопамяти определяет один элемент отображения размером в точку на экране монитора, то каждая точка экран (и соответствующий ей элемент видеопамяти) обозначаются термином пиксел (pixel — picture element).

Регенерация видеопамяти

Задача системы вывода изображений (видеоконтроллера) состоит в циклическом построчном просмотре экранного буфера от 25 до 100 раз в секунду. Адреса видеопамяти генерируются синхронно с координатами растра и содержимое выбранных пикселов используется для управления цветом и интенсивностью луча. Общая организация системы вывода изображений приведена на рис. 0.2.1.

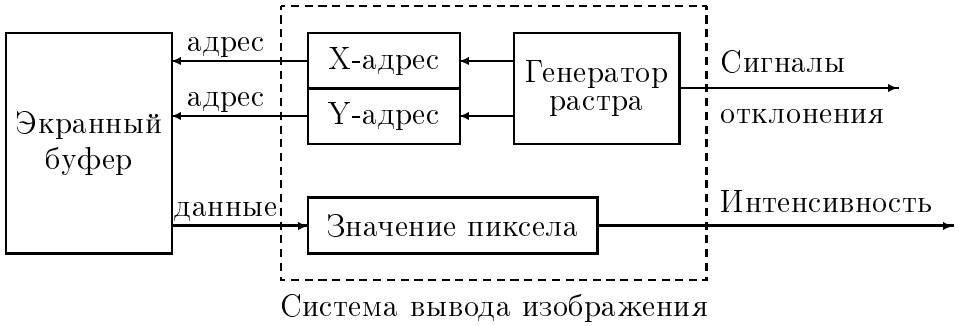

Рис. 0.2.1: Экранный буфер и система вывода изображения

Генератор растровой развертки формирует сигналы отклонения и управляет адресными X и Y регистрами, определяющими следующий элемент буфера регенерации.

В идеальном случае время, требуемое для регенерации экранного буфера, должно быть много меньше, чем время, необходимое для манипуляций с данными, что позволит быстро обновлять или двигать изображение. Это означает, что усилители отклонения и усилитель, управляющий интенсивностью луча, должны быть очень широкополосными, чтобы обеспечить требуемую скорость передачи данных между экранным буфером и системой вывода изображения.

Частота регенерации для графических дисплейных систем среднего разрешения лежит в пределах 50 Мгц, а для систем высокого разрешения достигает 100–125 Мгц, с явной тенденцией к частотам более 125 Мгц в последнее время. При таких частотах таймирование регенерации экранного буфера становится важной задачей при проектировании подсистемы графического вывода. Так ка обычная DRAM память не обеспечивает времени доступа, подходящего для существующих мониторов высокого разрешения, то регенерация видеопамяти на таких частотах требует ее специальной организации. Пример организации видеопамяти, построенной на обычной динамической памяти с произвольным доступом (DRAM) приведен на рис. 0.2.2.

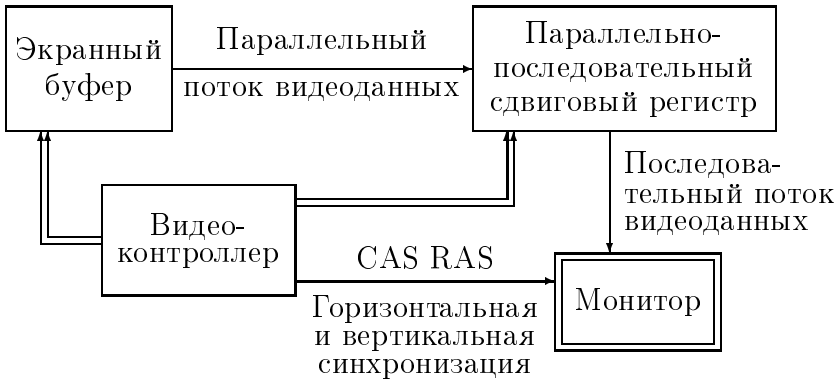

Рис. 0.2.2: Регенерация экранного буфера, построенного на обычной динамической памяти (DRAM)

В такой системе регенерация экранного буфера видеопамяти осуществляется с помощью параллельно-последовательного преобразования. Выполняя регенерацию, видеоконтроллер выставляет адрес слова, требуемое слово данных видеопамяти (обычно 16–32–64 бита) затем трансформируется в последовательный видеопоток (videostream) с помощью внешнего сдвигового регистра под контролем аппаратуры регенерации. На рис. 0.2.2 показана реализация регенерации экранного буфера для системы с одним слоем. Системы регенерации со многими слоями требуют такого же количества (16–32–64) битовых слов, подлежащих регенерации и параллельно-последовательных сдвиговых регистров, что и число битовых слоев видеопамяти.

Если частота регенерации экранного буфера составляет порядка 100 Мгц, то такое параллельно-последовательное преобразование уменьшает требования к частоте тактирования параллельно считываемого слова из экранного буфера видеопамяти до 6.25 Мгц, что требует времени доступа порядка 160 нс. При такой организации видеопамяти манипуляции с данными и обновление экрана должны происходить во времена межстрочного и межкадрового интервалов, когда регенерации не происходит. Таким образом, узкое место для обычной DRAM памяти в качестве видеопамяти в графических дисплейных системах вытекает из двух противоречивых требований:

- для растровых дисплейных систем должна осуществляться постоянная регенерация экранного буфера видеопамяти, что требует считывания выводимой на экран монитора графической информации с периодическим, жестко заданным циклом;
- с другой стороны, требуется время для обновления больших массивов данных видеопамяти со стороны собственно аппаратуры генерации изображений, работающей, как правило, в цикле чтение-модификация-запись.

Доступные в настоящее время DRAM устройства даже с наиболее быстрыми режимами доступа не обеспечивают быстрого чтения их содержимого для поддержки требуемого ритма регенерации, оставляя крайне мало времени графическому процессору для модификации изображения. Таким образом, ограниченная полоса пропускания DRAM памяти ограничивает доступ аппаратуры формирования изображений к данным видеопамяти на время значительных периодов регенерации экранного буфера. Проблема усложняется по мере увеличения экранного буфера из-за возрастания числа отображаемых пикселов для мониторов высокого разрешения или при увеличении числа битов на пиксел в системах с большим количеством отображаемых цветов.

Для решения этой проблемы разработаны различные архитектуры видеопамяти, включая двухпортовую видеопамять, двойное буферирование и др.

Однако лучшее решение этой проблемы достигается за счет применения нового типа DRAM памяти, получившей название VRAM (Video Random Accses Memory), например. Texas Instrument 4161, разработанной специально для использования в качестве памяти изображения в растровых дисплейных система. Структурная схема подобной памяти приведена на рис. 0.2.3.

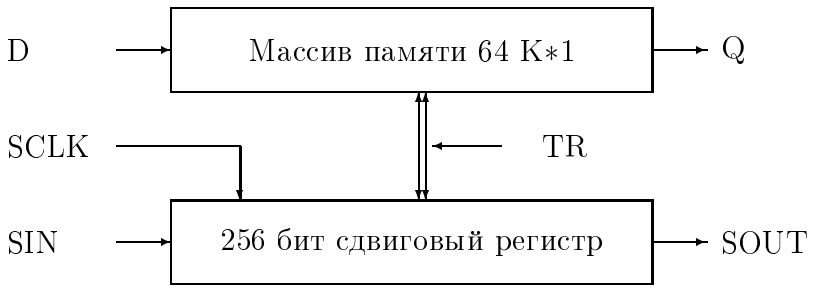

Рис. 0.2.3: Структурная схема VRAM памяти

Эта видеопамять содержит 2 порта, обеспечивая независимый доступ со стороны видеоконтроллера для регенерации и аппаратуры формирования изображений — графических процессоров. VRAM фактически представляет собой обычную DRAM память, которая была “внутренне” модифицирована посредством добавления сдвигового регистра. D и Q — это обычные входы и выходы порта с произвольной выборкой. Сигнал TR активируется на время передачи данных между сдвиговым регистром и видеопамятью. Сигналы SIN и SOUT — последовательные вход и выход сдвигового регистра, а сигнал SCLK — последовательный вход, управляющий сдвиговым регистром. Сдвиговый регистр загружается параллельным потоком в 256 бит из массива памяти за один цикл регенерации экрана. Длительность этого цикла не длиннее, чем стандартный цикл памяти. Обычно сдвиговый регистр загружается 1 раз во время обратного хода луча. Когда обратный ход заканчивается, на вход SCLK подается сигнал, вызывая сдвиг данных на последовательном выходе SOUT.

На рис. 0.2.3 показан модуль видеопамяти объемом 64 Кбайт. Видеопамять объемом 256 Кбайт может быть построена из 4 модулей по 64 Кбайт (рис. 0.2.4).

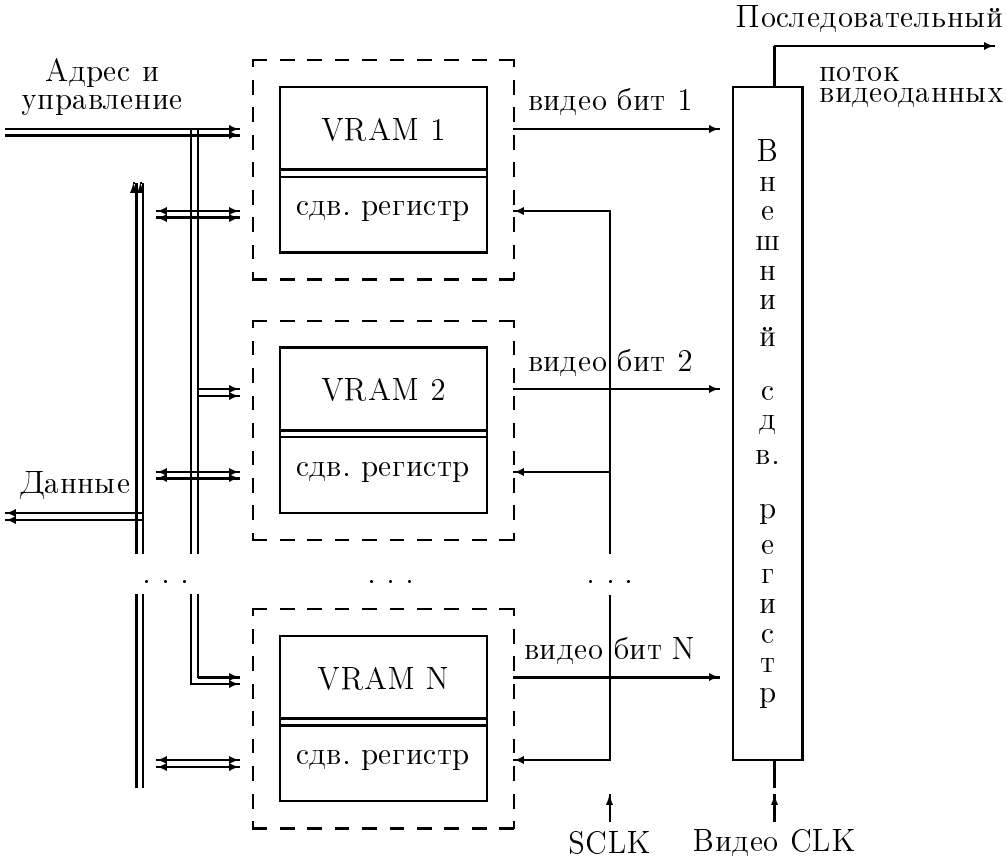

Рис. 0.2.4: Структурная схема многослойной VRAM памяти

В этом случае выходы SOUT от нескольких VRAM модулей подаются на параллельные входы внешнего сдвигового регистра, последовательный выход (CLK) которого тактируется со скоростью вывода точек (видеопотока битов), требуемой для регенерации экрана монитора.

В видеопамяти с такой организацией время на регенерацию экранного буфера (отображения на экран монитора) составляет менее 1.5% времени доступа. В системах же с обычной DRAM памятью время на регенерацию экрана составляет от 40% до 60% времени доступа.

Таким образом, применение VRAM обеспечивает практически полное время доступа для модификации данных видеопамяти, так как на одну строку сканирования растра требуется одна загрузка сдвигового регистра. Следовательно, в то время как предварительно загруженные видеоданные "выталкиваются" из сдвигового регистра в канал графического вывода, одновременно может осуществляться произвольный доступ к видеопамяти со стороны графических процессоров для модификации изображения.

Модификация данных в видеопамяти

Рассмотрим архитектуры видеопамяти с точки зрения манипуляции/обновления данных. Вопросы, относящиеся к выборке и обработке данных в видеопамяти графическим и/или центральным процессором, оказывают существенное влияние как на организацию самой видеопамяти, так и на внутреннюю архитектуру технических средств формирования изображений.

Изображение, хранящееся в видеопамяти, концептуально может быть представлено в виде куба (рис. 0.2.5).

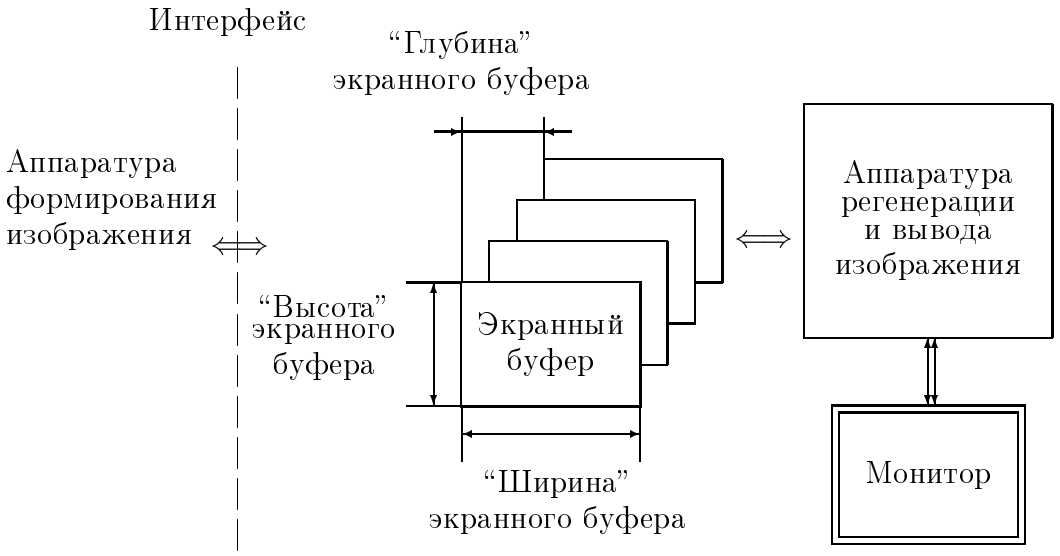

Рис. 0.2.5: Графический экранный буфер

Каждый пиксел, выводимый на экран монитора, состоит из отдельных битов видеопамяти, находящихся внутри куба.

Соотношение между значением пиксела, отображаемого из экранного буфера видеопамяти, и цветом на экране монитора устанавливается с помощью таблицы цветности видеоконтроллера. Доступ к данным, хранящимся внутри куба, необходим для их модификации и манипуляций с ними, регенерации экранного буфера и его обновления. В основном имеются 3 конфигурации: организация видеопамяти “в глубину”, ориентированная на обработку элементов отображения — ЭО (пикселов), организация видеопамяти в виде битовых слоев (разрядных матриц) и “смешанная” архитектура.

Архитектура “в глубину”

При такой организации видеопамяти обрабатываемые в каждый момент данные есть пиксел. В этом случае для многих слоев видеопамяти, генерируемый адрес вызывает слово данных, представляющих композицию битов “сквозь” слои, составляющие видеопамять (отсюда появился термин “глубина пиксела” — “pixel depth”). Такая архитектура применяется в системах высокого разрешения, предназначенных для обработки цветной трехмерной графической информации, например, в обработке изображений и моделировании структур твердых тел, т.е. там где значения каждого пиксела подвергаются интенсивным вычислениям. Эти применения, как правило, требуют “глубины пиксела” от 8 до 22–24 бит. В архитектуре “в глубину” данные в видеопамяти обрабатываются поэлементно. В случае использования для воспроизведения изображений, состоящих из нескольких цветовых плоскостей, адрес, направляемый в экранный буфер, генерирует слово данных, составленное из битов, представляющих собой одноименные разряды требуемых разрядных матриц.

"Слойная" архитектура

В "слойной" ("plane") архитектуре данные видеопамяти обрабатываются как одно слово (обычно 16 бит) в каждый момент времени (пословная обработка) и отдельно для каждого слоя (разрядной матрицы).

Чтобы изменить один разряд слова видеопамяти, вместе с ним необходимо передать и оставшиеся 15 разрядов. Кроме того, для того чтобы обеспечить позиционирование и перемещение изображения с точностью до бита и с удовлетворительной скоростью, требуется специализированная аппаратура, осуществляющая быстрые сдвиги и "слияния" цепочек битов видеопамяти ("barrell shifter" [76]). Однако, несмотря на это условие, "слойные" архитектуры видеопамяти являются наиболее популярными в интерактивных 2D системах, так как требуют менее интенсивных вычислений значений пикселов (по сравнению с архитектурой "в глубину"), но более интенсивных вычислений при создании и перемещении изображения. Такие архитектуры видеопамяти часто находят применение в системах обработки инженерной и экономической информации, поскольку для них характерен значительный объем операций, связанных с манипуляциями данными и перемещении изображения.

Кроме того, достоинством такой архитектуры является возможность пословного доступа к видеопамяти со стороны центрального процессора (при соответствующей организации такая видеопамять для центрального процессора ничем не отличается от обычной оперативной памяти). Пословный доступ при достаточной разрядности слова (16–32 бит) и ограниченных требованиях к цвету (до 16 цветов, что требует четырех слоев видеопамяти) и при наличии аппаратных средств быстрого сдвига дают выигрыш в скорости, так как за один цикл памяти считывается сразу 16–32 битов данных, подлежащих модификации.

"Смешанная" архитектура

В этой архитектуре доступ к данным видеопамяти может производиться как по "глубине" пиксела, так и в "ширину", реализуя лучшие возможности обеих архитектур.

Следует отметить, что такие архитектуры в последнее время применяются в дисплейных системах наиболее дорогих рабочих станций, поскольку требуют значительных аппаратных затрат на их реализацию.

Во многих специальных применениях используются и другие архитектуры, например, [60, 95, 115].

0.2.3 Технические средства формирования изображений

В основе архитектуры современных рабочих станций лежат многопроцессорность и конвейерная обработка. Такой подход позволяет разделить процессы модельных, видовых и функционально-растровых преобразований и дает возможность каждому из них выполняться на выделенном, как правило, специализированном устройстве со своей собственной скоростью. (Модельные преобразования — преобразования, используемые для построения модели объекта в системе координат пользователя. Видовые преобразования — преобразования, используемые после модельных при выполнении отображения в поле вывода. Математически модельные и видовые преобразования имеют одинаковую форму, но применяются в различное время и относятся к разным подсистемам графического конвейера. Функционально-растровые преобразования — преобразование примитивов вывода в растровую форму).

Из вышеперечисленных наиболее длительным и обрабатывающим бóльшие объемы данных является процесс функционально-растровых преобразований.

Ускорение этого процесса достигается за счет усложнения архитектур дисплейных систем, в состав которых вводятся дополнительные вычислительные мощности — от высокопроизводительных процессоров общего назначения [73] и/или специально разрабатываемых процессоров [79, 71] до специализированных графических СБИС [72, 36], берущих на себя основные функции по формированию изображений в растровой форме и управлению видеопамятью.

Специализированные микросхемы для графических дисплейных систем занимают в настоящее время одно из важных мест на мировом рынке. Ведущие фирмы в этой области: Advanced Micro Devices, Intel, NEC, Texas Instruments, Hitachi, National Semiconductor. Разработки этих фирм в области технических средств формирования изображений представляют собой высокопроизводительные графические процессоры, которые требуют минимального вмешательства со стороны центрального процессора для выполнения графических функций высокого уровня. Например, разработки фирм Intel и Texas Instruments выполнены в виде отдельных СБИС, реализующих широкий набор функций. Набором функций, реализуемых этими однокристалльными процессорами, фирмы намерены обеспечить выполнение всех запросов потенциальных потребителей — разработчиков графических дисплейных систем. Фирма National Semiconductor разработала набор СБИС, из которых можно строить графические системы с различными характеристиками, ориентированные на выполнение необходимого набора функций. Предполагается, что в этом случае разработчик сам определит требуемый набор функций и реализует его, используя те или иные компоненты из набора СБИС.

Несмотря на то, что при проектировании этих устройств использовались концептуально разные подходы, (что наложило свой отпечаток на их функциональные возможности и внутреннюю архитектуру) все они ориентированы на формирование изображения в битовых картах (BitMap) и обладают рядом общих характеристик:

- поддержка современных 2D графических стандартов (GKS, CGI и т.д.),
- возможность адресации больших объемов видеопамяти (от 4 Мбайт и более),
- эффективная реализация операции блочной переписи (BitBlt Bit boundary Block Transfer), представляющей мощное средство для создания многооконных графических систем. Скорость выполнения операций блочной переписи превышает 20 Мбит/с.

Рассмотрим некоторые из устройств более подробно.

Графические процессоры TMS-34010 и TMS-34020

TMS-34010 — первый графический микропроцессор, поддерживающий пользовательскую графику вместо встроенных графических примитивов [59]. При постановке задачи разработчики TMS-34010 определили, что простое расширение числа аппаратно реализованных графических функций было бы (на их взгляд) фундаментальной ошибкой по следующим причинам:

1. Спектр графических функций устройства был бы жестко зафиксирован. В этом случае возможна аппаратная поддержка только относительно немногих примитивов вывода (как правило, определенных современными графическими стандартами), а новые примитивы или старые, но с расширенными возможностями, не смогут быть поддержаны.

2. Даже стандартизованные графические примитивы могут требовать многих атрибутов отрисовки, таких как ТИП ЛИНИИ, ШИРИНА ЛИНИИ, ЦВЕТ ЛИНИИ, ПРОЗРАЧНОСТЬ и других. Аппаратная реализация означает “жесткий” выбор поддерживаемых атрибутов, следовательно, некоторые не часто используемые либо незастандартизованные, но существенные

для отдельных применений атрибуты будут опущены, например, ФОРМА КОНЦОВ ЛИНИИ (endpoint shape).

3. Высококачественная графика требует точного контроля над алгоритмами формирования изображений. Концептуально отрисовка в битовых картах означает выбор ближайших пикселов на растровой дискретной сетке, что вызывает ошибки округления, видимые как ступеньки ("зазубрины" — "jaggies"), образующие "эффект лестницы" при отрисовке наклонных линий. Графический пакет качественной графики может требовать доступа "сверху" (со стороны программы пользователя) для исправления такого рода эффектов (алгоритм сглаживания — antialiasing), либо дополнительных параметров для реализации этих требований на аппаратном уровне.

4. Формат дисплейного списка, или команд формирования изображений может варьироваться в соответствии с требованиями пользователя. Например, для формирования шрифтов с фиксированной матрицей знакоместа (непропорциональный шрифт) используется относительно простой формат команды, в то время, как шрифт с переменными размерами матрицы (пропорциональный) требует более сложного формата команды.

Единственный способ удовлетворить всем возможным требованиям при реализации команд формирования изображений — это иметь полностью программируемый процессор, интерпретирующий графические команды.

Таким образом TMS-34010 был разработан чтобы предоставить пользователю максимальную гибкость для реализации графических примитивов и, в то же время, обеспечить требуемую скорость формирования изображений (не сильно уступая в этом отношении графическим процессорам и контроллерам с "чисто аппаратной" реализацией графических функций). С помощью TMS-34010 можно реализовать практически любые алгоритмы отрисовки графических примитивов, которые могут быть востребованы как для отдельных специальных приложений, так и по мере появления новых графических стандартов.

Набор из 120 инструкций поддерживает восемь типов адресации и четыре основных типа данных — массивы упакованных пикселов, X-Y координаты, прямоугольные окна и битовые поля произвольной длины.

Для реализации алгоритмов графических примитивов вывода в TMS-34010 используются инструкции общего назначения, а для повышения скорости отработки этих алгоритмов (собственно манипуляций с пикселами в видеопамяти) используется аппаратно реализованный набор графических инструкций. Таким образом, в TMS-34010 полный набор инструкций общего назначения, который может поддержать программирование на языке высокого уровня, "замешан" с мощным набором специальных графических инструкций, например, такими как перепись блока битов (BitBlt), являющейся базовой операцией в современной растровой графике.

Графический процессор TMS-34010 содержит полностью программируемый 32-х разрядный процессор со схемами адресации памяти и системой команд, ориентированной на операции над пикселами, набор из 31 32-битных регистров и кэш-память инструкций на 256 байт. В дополнение TMS-34010 содержит контроллер управления растровым монитором, отдельный интерфейс с центральным процессором, интерфейс с DRAM/VRAM памятью.

Так как инструкции процессора выбираются из кэш-памяти, то он может выполнять вычисления параллельно с работой с памятью его устройства управления ОЗУ.

Для повышения скорости отработки графических функций используются специальные аппаратные средства: барабанный сдвигатель (barrell shifter); логика маскирования и слияния цепочек пикселов; аппаратура определения левого единичного бита; компаратор окон, связы-

вающий отсечение графических примитивов вывода с прямоугольными зонами графического буфера.

TMS-34010 использует такое упорядоченное расположение пикселов в видеопамяти, при котором оно представляет собой единое линейное адресное пространство. Процессор имеет возможность адресации от 1 до 32 бит в линейном и координатном масштабах. Возможное число битов, описывающих пиксел, ограничено 16-ю битами. Слово данных может быть определено как четыре 4-битных пиксела, как два 8-битных, один 16-битный. Такой подход удачно сочетается с возможностями 32-битной архитектуры, так как он позволяет производить быстрые операции над пикселами любого заранее указанного размера. Однако, скорость выполнения операций чтения-модификации-записи элемента отображения при такой архитектуре не является постоянной при работе в режимах с различным числом бит на пиксел, что несколько затрудняет построение универсальных гибких графических систем.

С использованием TMS-34010 могут быть построены системы с объемом видеопамяти до 8 Мбайт, содержащей некоторое количество неотображаемой “заэкранной” информации, такой как различные шрифты или предварительно подготовленные изображения в виде битовых карт различного формата, например, пиктограммы. Пример использования TMS-34010 при построении графической системы приведен на рис. 0.2.6. Как правило, система, базирующаяся на TMS-34010, в общем случае содержит видеопамять порядка 0.5 Мбайт (VRAM) и 1–3 Мбайт программной памяти (DRAM), ПЗУ с программами инициализации, эмуляции предыдущих видеоадаптеров и графическими библиотеками [48].

Примером системы, реализованной на базе TMS-34010, может служить плата расширения для PC AT GENESIS 1024 фирмы National

Рис. 0.2.6: Использование TMS-34010 в графической системе

Design Inc. При разрешении $1024 \times 768 \times 4$ точек плата обеспечивает скорость рисования до 48 Мпиксел/с и поставляется с CGI, AutoCAD и HPG (Harvard Presentation Graphics) совместимыми интерфейсами.

TMS-34010 тактируется от внешнего 50 МГц генератора. Внутренняя частота составляет 6.25 МГц, что дает возможность выполнения более 6 млн. инструкций/с при работе с кэш-памятью. Устройство выполнено по 1.8 микронной CMOS технологии в 68-выводном корпусе.

Графический процессор TMS-34020

В 1988 г. Texas Instrument анонсировала преемника TMS-34010 — графический процессор TMS-34020. Он включает 32-разрядный шинный интерфейс (со страничным режимом доступа), высокоскоростной тактовый генератор (10 MIPS) и дополнительные графические инструкции (3-операндные PixBlt). Графический процессор не делает различия между программной и дисплейной памятью. Он может адресовать 512 Мбайт. Пикселы могут быть доступны по их X-Y экранным координатам, которые автоматически преобразуются в линейное адресное пространство памяти. Скорость вычерчивания линий достигает 5 Мпикселов/с.

Графический процессор выполняет BitBlt операции не только с 16 булевскими операциями, но и также и с 8 арифметическими функциями, такими как сложение или вычитание значений пикселов. Единственной BitBlt инструкцией графический процессор может извлечь данные для символа с 1 битом на пиксел из бинарной таблицы шрифта, транслировать их в многобитные цветные данные и разместить в некоторое место дисплейной памяти. Скорость выполнения BitBlt операций составляет 25 МБит/с.

Графический процессор поддерживает отсечение во время отрисовки в произвольно определенном окне отсечения. Инструкция теста окна определяет находится точка внутри или вне окна, так что если линия не пересекает прямоугольник, то она не будет вычерчиваться.

Для использования в 3D графических системах предусмотрен сопроцессор TMS-34082 для выполнения операций с плавающей запятой, который подключается к графическому. Этот чип работает со скоростью 40 MFLOPS. Он уместен для вычислений 3D геометрии и освещенности. В дополнение к функциям АЛУ он поддерживает 3D функции типа умножения матриц 4×4, отсечения полигона (прямое вычисление точки пересечения), тестирование заднего плана и генерацию 3D кубических сплайнов. Эти операции выполняются микропрограммно.

Гибкость графического процессора способствует значительному расширению спектра графических акселераторов, построенных на его основе, особенно для индексированных цветных дисплеев.

Использование TMS-34020 в графической системе показано на рис. 0.2.7.

Графический сопроцессор Intel 82786

INTEL 82786 (i82786) предназначен для использования в растровых графических станциях САПР, а также в профессиональных персональных ЭВМ высокой производительности [112]. СБИС i82786 обеспечивает скоростную обработку графических и текстовых данных, осуществляет их высококачественное отображение и поддерживает режим многозадачности.

Основным положением при разработке i82786 была возможность его использования в качестве графического сопроцессора в системах на основе микропроцессоров фирмы Intel 80186, 80286, 80386. В отличие от TMS-34010, который работает автономно, i82786 работает под управлением центрального процессора: для него не существует собственных программ, а лишь поток

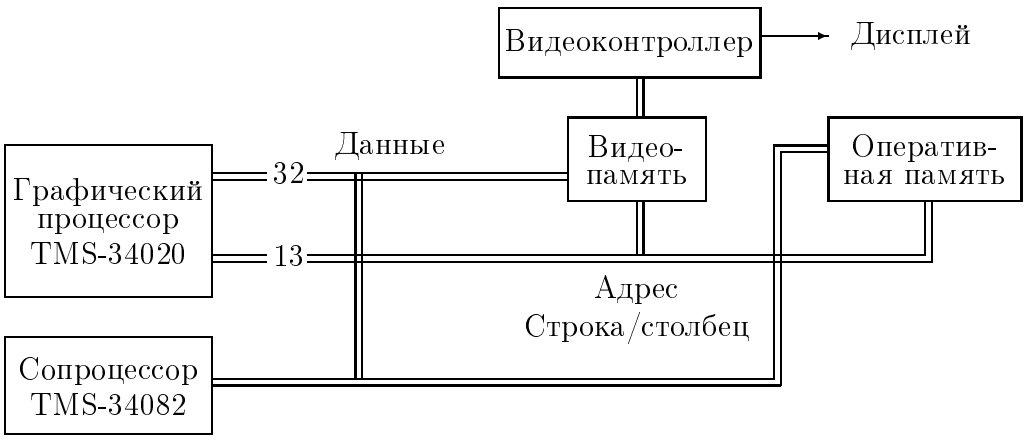

Рис. 0.2.7: Использование TMS-34020 в графической системе

команд и данных, получаемых от центрального процессора графической станции. Графический сопроцессор i82786 имеет собственную 22-разрядную шину адресов (что позволяет адресовать до 4 Мбайт видеопамяти) и 16-разрядную шину данных, осуществляет доступ в системную память и выбирает данные и команды, сформированные в ЦП. Набор команд сопроцессора делится на четыре функциональные группы: общего управления; управления параметрами отрисовки изображений; генерации графических примитивов и операций блочной переписи и формирования окон.

Таким образом, в отличие от TMS-34010, специалисты фирмы Intel стремились создать графический сопроцессор, который покрывает практически весь спектр графических примитивов вывода современных графических стандартов, включая операции блочной переписи.

Особенностью графических систем, построенных на основе i82786, является возможность доступа центрального процессора непосредственно к видеопамяти, независимо от i82786. Важно отметить, что поскольку при этом центральный процессор станции имеет доступ ко всей видеопамяти, то прикладные программы пользователя могут реализовывать не часто встречающиеся, но, возможно, существенные примитивы отрисовки для высококачественной графики. Этим достигается требуемая гибкость реализации “нестандартизованных” графических функций, однако при этом теряется скорость их рисования.

Эффективная поддержка многооконной технологии в многозадачном интерактивном режиме достигается за счет введения специальной внутренней архитектуры. СБИС i82786 содержит четыре основных модуля: графический и дисплейный процессоры, устройство сопряжения с магистралью и контроллер видеопамяти. Все модули работают относительно независимо, что обеспечивает высокие скоростные характеристики микросхемы. Архитектура и использование i82786 в графической системе приведены на рис. 0.2.8.

Графический процессор

Выполняет команды, размещенные в системной памяти и формирует изображения в битовых картах видеопамяти для дисплейного процессора во взаимодействии с контроллером видеопамяти и интерфейсным устройством шины. Процессор оптимизирован для отработки современных графических стандартов (GKS, CGI, Microsoft Windows). Все команды, выполняемые

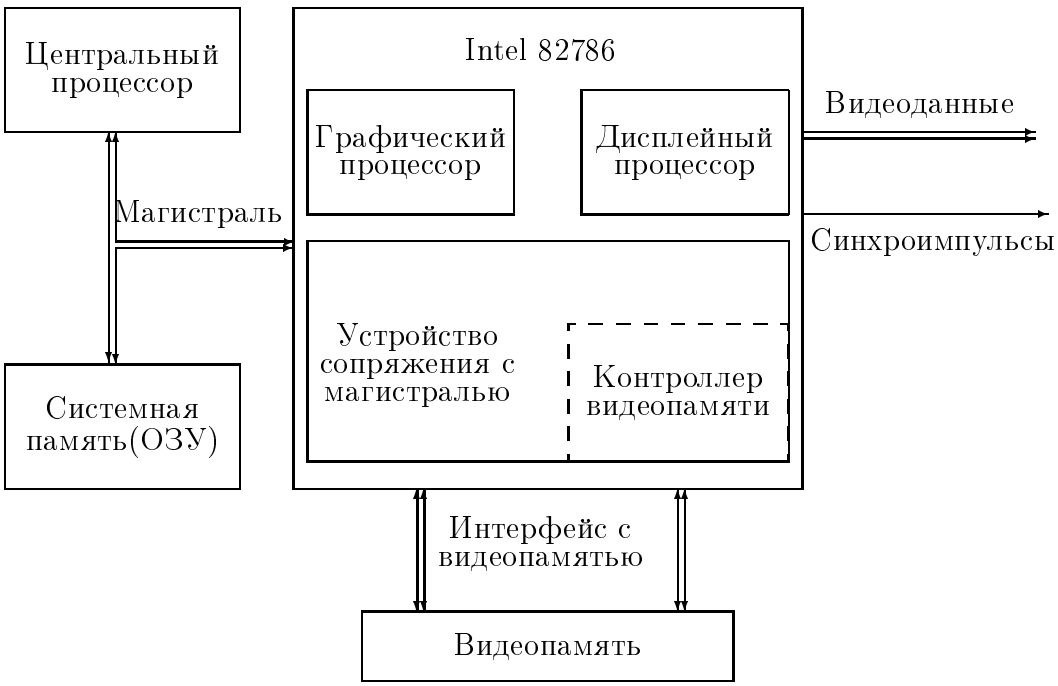

Рис. 0.2.8: Использование сопроцессора Intel 82786 в графической дисплейной системе

графическим процессором, включая BitBlt операции, сопровождаются богатым набором атрибутов отрисовки, наиболее часто используемых в стандартных графических пакетах. Графический процессор обладает развитыми аппаратными возможностями, позволяющими снизить до минимума число параметров в командах для формирования примитивов в растровой форме. Список графических команд и параметров создается центральным процессором и хранится в системной памяти. Графические команды выбираются из связанного списка графических команд при обращении к системной памяти через логику шинного интерфейса и обрабатываются графическим сопроцессором.

Для синхронизации работы ЦП и сопроцессора используется флаг завершения формирования списка параметров. СБИС i82786 начинает выполнение команды, если ЦП завершил формирование списка параметров к этой команде и установил соответствующий флаг. Такой режим обмена позволяет значительно повысить системную производительность.

Производительность графического процессора зависит от частоты доступа к видеопамяти (каждое модифицируемое в видеопамяти слово требует цикла “чтение-модификация-запись”). Скорость работы графического процессора определяется также шириной полосы пропускания видеопамяти. Имея специальную память (VRAM), графический процессор может использовать до 99% полосы пропускания; при использовании в качестве видеопамяти обычных динамических БИС ЗУ (DRAM) — 50%–90%.

Дисплейный процессор

Особенность дисплейного процессора — возможность реализации полиэкранного режима за счет имеющихся аппаратных средств поддержки. Дисплейный процессор преобразует битовые карты, создаваемые графическим процессором в растровые последовательности для видеоконтрольного устройства, которое отображает их в виде отдельных окон на экране графического

монитора. Процессор оптимизирован для данных, представленных в виде битовых карт. Дисплейный процессор работает независимо от графического по собственной программе, описывающей таблицу конфигурации кадра (экранного буфера), которая формируется ЦП и размещается в определенной области видеопамяти. Обрабатывая битовые карты, дисплейный процессор воспроизводит их содержимое в нужных областях экрана, выполняя свои команды в течение межкадрового промежутка, что обеспечивает получение "чистого" (без мерцаний) изображения. Для синхронизации ЦП и дисплейного процессора используется механизм квитирования. После выполнения каждой команды дисплейный процессор устанавливает флаг, указывающий что ЦП может загружать следующую команду.

Дисплейный процессор разделяет экранную область видеопамяти на горизонтальные полосы высотой в произвольной число строк. Полосы представляют собой блоки ("черепицы" — "tiles"), которые относятся к различным сегментам окна (рис. 0.2.9). Структура управляющего блока дескриптора приведена на рис. 0.2.10.

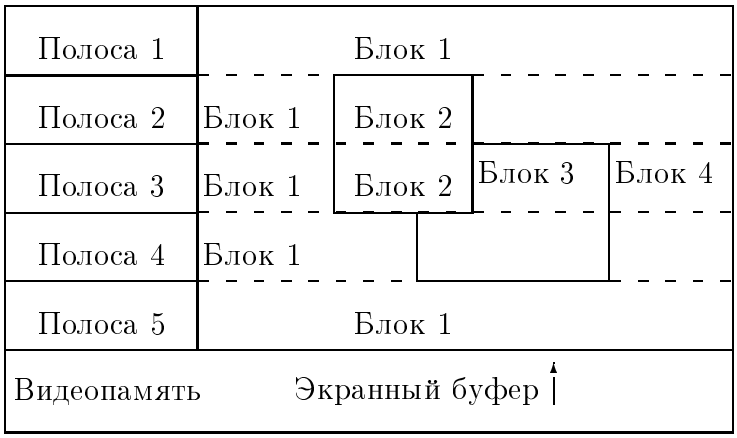

Рис. 0.2.9: Организация оконных сегментов для i82786

Допускается до 16 горизонтальных окон-сегментов на каждую отдельную строку растра. Минимальная ширина окна равна 1/16 длины строки.

Таблица описаний дескрипторов полос ("черепиц") содержит заголовок, в котором указаны число строк в полосе, адрес дескриптора следующей полосы, число блоков в текущей полосе. Затем идут данные, описывающие блоки и включающие указатели, ассоциированные с битовыми картами и атрибуты. Таким образом для каждого кадра формируется таблица конфигурации в виде связанного списка команд дисплейного процессора.

Одной из важных особенностей дисплейного процессора является возможность поддержки аппаратного курсора. Можно устанавливать размер курсора в матрице 8×8 или 16×16 пикселов. Курсор может быть прозрачным, непрозрачным, в виде стрелки, креста и т.д. Цвет курсора, его прозрачность и мерцание задаются программным способом. Программируются также сигналы синхронизации мониторов с разрешающей способностью до 4096×4096 пикселов.

Интерфейсные устройства шины реализуют четыре логических интерфейса: внешний; доступа к видеопамяти для внутреннего графического процессора; доступа центрального процессора к регистрам графического и дисплейного процессоров и видеопамяти; сигналов регенерации видеопамяти.

Рис. 0.2.10: Структура управляющего блока дескриптора оконного сегмента для i82786. T,B,L,R — управляющие биты для верхней, нижней, левой и правой границ; WST — статус окна; Z — увеличение; F — фоновое поле.

В зависимости от числа бит на пиксел изменяется режим работы дисплейного процессора. Так, например, можно использовать следующие частоты тактирования: 25 МГц при 8 бит/пиксел, 50 МГц при 4 бит/пиксел, 100 МГц при 2 бит/пиксел, 200 МГц при 1 бит/пиксел.

Работа i82786 обеспечивается системным тактированием 10 МГц и видеотактированием 25 МГц и обеспечивает следующие скорости формирования изображений:

- построение линий — 2.5 млн. пикс./с,
- построение окружностей и дуг — 2 млн. пикс./с,
- блочная перепись — 24 млн. пикс./с,
- заливка областей — 30 млн. пикс./с (с использованием процедуры заливки горизонтальной линией),
- построение символов — 25 млн. символов/с (шрифт 16×16. Более крупные шрифты требуют программной поддержки со стороны ЦП).

Сопроцессор i82786 может выполнять вертикальный и горизонтальный Roll (перемещение) без дополнительной внешней аппаратуры, обеспечивать аппаратное увеличение с коэффициентом масштабирования до 64, поддерживать разрешающую способность от $640 \times 480 \times 8$ до $1024 \times 1024 \times 2$.

Микросхема выполнена по CHMOS технологии в 88-выводном корпусе.

Набор графических СБИС National AGCS

Набор графических СБИС AGCS 85xx (Advanced Graphics Chip Set) фирмы National Semiconductor предназначен для построения гибких модульных графических систем, имеющих высокую производительность и высокую скорость формирования изображения и обмена графической информацией в виде битовых карт [64]. В набор AGCS входят: растровый графический процессор DP-8500 (RGP — Raster Graphics Processor), процессор обмена блоками информации в виде битовых карт DP-8510 (BPU — BitBlt Processing Unit), генератор тактовых импульсов для вывода видеоизображения DP-8512 (VCG — Video Clock Generator) и высокоскоростной сдвиговый регистр DP-8515 (VSR — Video Shift Register). Набор AGCS соединяет в себе производительность функционально-ориентированных устройств (например, i82786) с возможностями программируемых графических процессоров (например, TMS-34010).

Растровый графический процессор DP-8500

Изготовлен по 2-х микронной CMOS технологии и работает на частоте 20 МГц. Процессор характеризуется 100 нс шинным циклом при работе со следующими непосредственно друг за другом векторными и блочными операциями. Скорость построения линий на экране равна 300 нс/пиксел. Производительность типичной системы на основе растрового процессора лежит в пределах от 10 млн. до 160 млн. пикс./с. Достижение подобных характеристик стало возможным благодаря специальной внутренней архитектуре растрового процессора DP-8500, использующего отдельное АЛУ для адресации и отдельное АЛУ для обработки данных, т.е. разделяющей функции, которые в традиционных процессорах, как правило, реализуются единым АЛУ.

Процессор DP-8500 разделен на два блока, которые функционируют под управление обычного набора микрокоманд и представляют собой арифметико-логические устройства (одно для адресации, а другое для обработки данных), работающих параллельно. Оба АЛУ управляются

единым потоком команд, считываемых из внешней памяти и декодируемых с помощью устройства микропрограммного управления. В число этих команд входят команды межрегистровых передач обоих процессоров и другие вспомогательные средства, размещенные на кристалле. Благодаря архитектуре со сдвоенными АЛУ, процессор DP-8500 можно с успехом применять в системах с различными типами архитектуры дисплейной системы.

Адресный процессор — это 28-разрядное АЛУ со своим набором команд и группой их 16 28-разрядных регистров.

Процессор обработки данных представляет собой 16-разрядное АЛУ, имеющее относительно богатый набор команд, оптимизированный для решения графических задач и группу из 16 16-разрядных регистров. Кроме того, предусмотрены регистры для выполнения ряда специализированных функций, которые связаны с обслуживанием графических операций и операций по регенерации изображения. В составе DP-8500 имеются специализированные схемы для реализации ряда графических операций и, в первую очередь, передач блока битов (блочная перепись), построения линий и отсечения участков изображения, а также видеоконтроллер кадровой развертки.

Располагаясь в тракте конвейерной обработки графической информации, DP-8500 ожидает поступления (завершения формирования) дисплейного списка: списка подлежащих отработке графических примитивов вывода. После его получения DP-8500 приступает к непосредственной интерпретации команд дисплейного списка и графические примитивы, переведенные в растровую форму, записываются в видеопамять. Выполнив последнюю команду в списке, DP-8500 сигнализирует о завершении процедуры, заканчивает обмен с ЦП и дает разрешение на готовность отработки следующего потока команд. Структурная схема растрового графического процессора DP-8500 приведена на рис. 0.2.11.

Следует отметить, что в отличие от конкурирующих графических процессоров, применение DP-8500 не ограничено его использованием с экранным буфером всего лишь одного типа. Процессор можно применять не только с любой из двух основных архитектур видеопамяти (“слойной” или “в глубину”), но также и в комбинированных подходах (“смешанная” архитектура видеопамяти).

Процессор обмена блоками графической информации DP-8510

Представляет собой сопроцессор манипулирования данными, работающий на частоте 20 МГц. DP-8510 — специализированное CMOS устройство, разработанное для использования в приложениях BitMap растровой графики. DP-8510 включает высокопроизводительную логику, конвейеризующую все необходимые функции, связанные с реализацией механизма блочной перепиши: сдвиги (за один такт) на требуемое число разрядов, маскирования и побитные логические операции. Устройство может работать под управлением процессора общего назначения (центрального процессора), машины микросостояний либо под управление растрового графического процессора DP-8500 (один DP-8500 может управлять до 16-ти DP-8510). Устройство поддерживает все манипуляции с данными, необходимыми для построения различных BitBlt систем. Для синхронизации с ЦП либо с DP-8500 используется механизм квитирования.

DP-8510 имеет два режима работы: BitBlt (блочная перепись) и формирование линий в растровой форме. Построение линий может рассматриваться как специальный случай BitBlt с высотой и шириной блока равными единице (точка). Для выполнения операций BitBLt управляющие регистры DP-8510 загружаются четырьмя параметрами: числом сдвигов, левой и правой масками и кодом операции. Далее происходит

Рис. 0.2.11: Структурная схема растрового графического процессора DP-8500 фирмы National Semiconductor

отработка операции блочной переписи под управлением, например, DP-8500.

Видеогенератор DP-8512

Служит для таймирования и управления растровой дисплейной графической системой. Устройство вырабатывает системный клок на частотах 20 МГц для DP-8500 и DP-8510, а также обеспечивает тактовые сигналы для управления и параллельной загрузки сдвиговых регистров DP-8515.

Сдвиговый регистр DP-8515

Преобразует параллельное слово данных из экранного буфера видеопамяти в высокоскоростной последовательный поток данных (пикселов), подаваемый на вход электронно-лучевой трубки, с максимальной частотой до 225 МГц.

Следует отметить, что набор DP-85xx может поддерживать все типы RAM памяти, включая статическую, динамическую и специальную видеопамять (VRAM).

Техническое решение построения дисплейной системы, реализованное фирмой National Semiconductor, сводится к разделению функциональных аппаратных средств графической обработки на две специализированные СБИС (DP-8500 и DP-8510), при этом DP-8500 выполняет все функции адресации и синхронизации, связанные с экранным буфером, и одновременно является интерфейсом, через который осуществляется передача адресов и команд центрального процессора. Манипуляции с данными, связанными с каждым слоем видеопамяти, осуществляются отдельным подчиненным процессором DP-8510, который выполняет операции маскирования, циклического сдвига и операции поблочной передачи данных с адресацией с точностью до бита. Структурная схема типичной графической системы, предлагаемой фирмой National Semiconductor, приведена на рис. 0.2.12.

В типичной графической системе с несколькими слоями видеопамяти растровый графический процессор DP-8500 производит формирование графических примитивов в видеопамяти и управляет подчиненными процессорами DP-8510, осуществляющими обмен блоками графической информации. После того как требуемая графическая функция выполнена, графический растровый процессор больше не участвует в графических манипуляциях (собственно в формированиях графических примитивов вывода в растровой форме). При необходимости передачи данных между различными слоями видеопамяти один подчиненный процессор DP-8510 выступает в качестве источника информации, а любая комбинация оставшихся подчиненных процессоров — в качестве места назначения передачи.

0.2.4 RISC-процессор с графическим устройством (i860)

В качестве альтернативы фирма Intel [78, 39] предприняла разработку RISC микропроцессора с включением графического устройства. i860 в первую очередь является высокоскоростным процессором и, во-вторых, он может выступать в рабочих станциях в качестве процессора, используемого в 3D графических подсистемах.

В состав i860 входят (рис. 0.2.13) управление памятью и шиной, кэш-память, центральный RISC-процессор для целых чисел, процессор плавающей арифметики из устройств для сложения и умножения вещественных и графическое устройство. Тактовая частота i860 — 50 МГц.

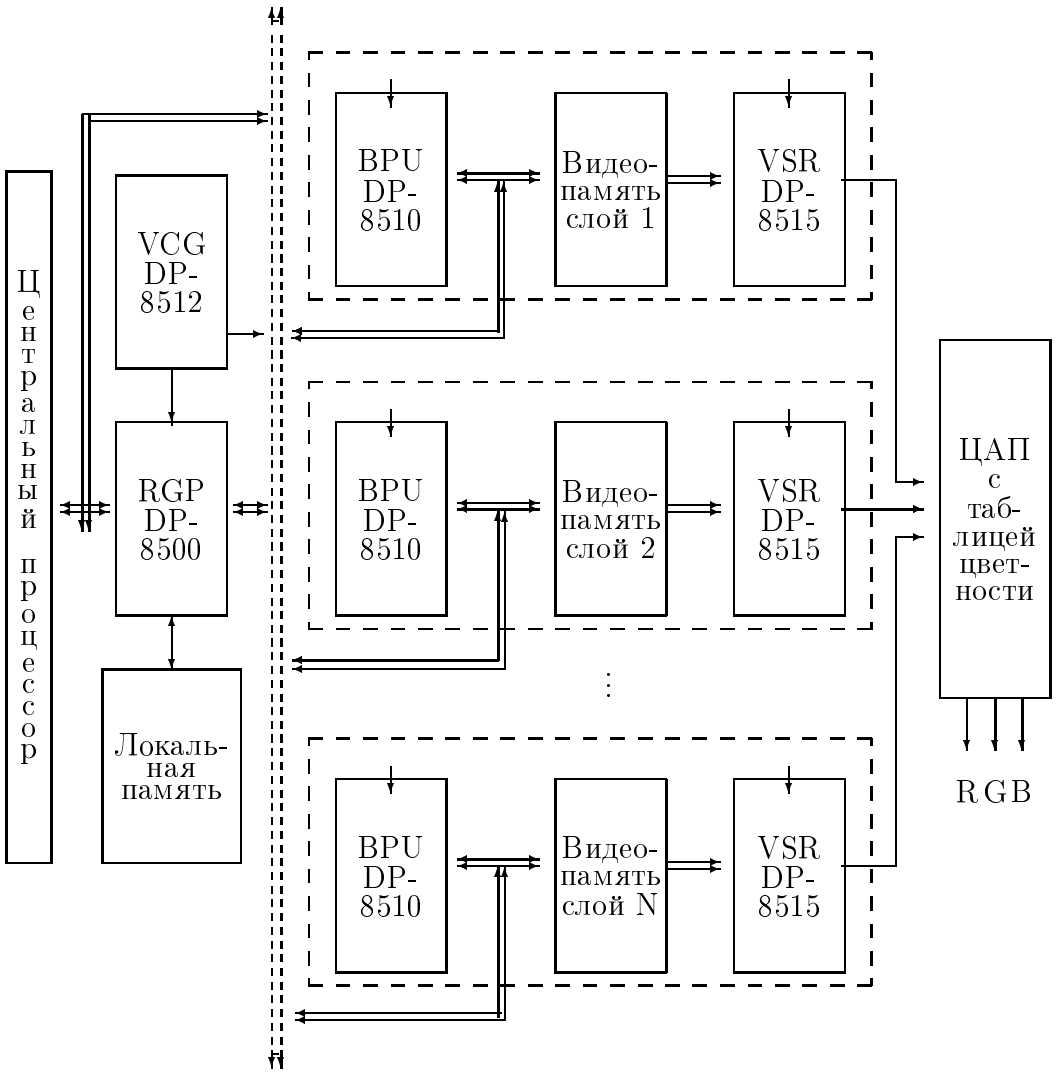

Рис. 0.2.12: Архитектура дисплейной системы со слойной организацией видеопамяти на базе набора СБИС AGCS фирмы National Semiconductor

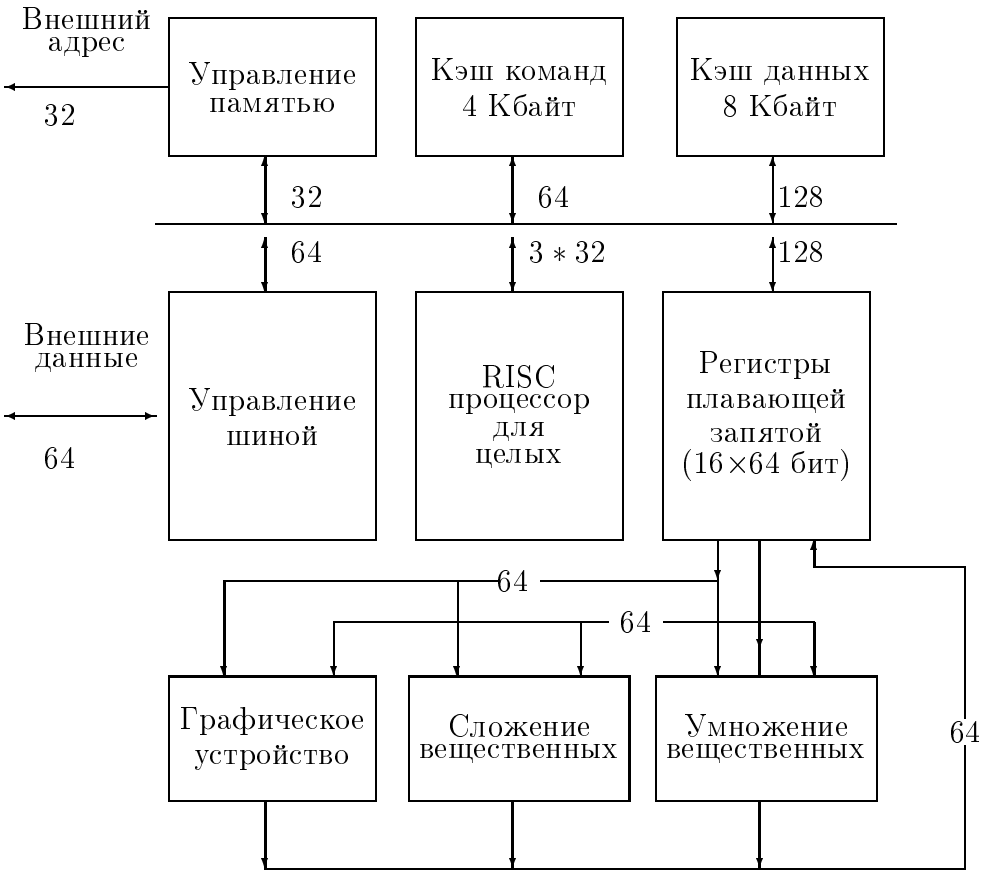

Рис. 0.2.13: Архитектура i860

Применение конвейера позволяет в каждом такте выполнять до трех команд — одну команду для целых, а также сложение и умножение вещественных.

Выборку и декодирование команд выполняет центральный процессор. Кроме этого он осуществляет следующие операции:

- загрузка и запоминание в целочисленных регистрах операндов, выбранных из памяти;
- пересылка операндов в регистры процессора с плавающей запятой;
- арифметические операции над 8, 16 и 32-битными целыми числами (операции с 64-битными целыми поддерживает графический процессор);
- сдвиговые и логические операции (И, И-НЕ, ИЛИ и исключающее ИЛИ);
- управление пересылками, к которым относятся команды переходов (непосредственные и косвенные) и команды вызова;
- управление системой;
- загрузка регистров управления, работа с кэш-памятью и шиной.

Процессор плавающей арифметики (ППЗ) содержит файл из 32 регистров и два самостоятельных устройства для сложения и умножения вещественных. ППЗ работает в скалярном или конвейерном режимах, используя для них различные наборы инструкций. В скалярном режиме на выполнение одной операции тратится от трех до четырех циклов. В конвейерном режиме результаты выдаются, в основном, после каждого цикла или после двух.

Скалярные инструкции ППЗ включают в себя:

- операции сложения, вычитания, сравнения, преобразования в целое и округления до целого (эти операции выполняет устройство сложения);
- операции умножения, получения обратной величины и извлечения квадратного корня (их выполняет устройство умножения);
- операции пересылки в целочисленные регистры. К конвейерным инструкциям, также выполняемым или устройством сложения или устройством умножения, относятся операции умножения, сложения, вычитания, преобразования в целое и округления до целого.

Графическое устройство фактически использует данные, поступающие от ППЗ и поддерживает размер пиксела в 8, 16 или 32 бита. Вне зависимости от размера пиксела оно может обрабатывать до нескольких пикселов одновременно в одном 64-битном слове. Конфигурация цветов пикселов следующая:

Размер пиксела	Битов на цвет Компонента 1	Битов на цвет Компонента 2	Битов на цвет Компонента 3	Битов на атрибут
8	$N(\Longleftarrow 8)$ битов на интенсивность			8-N
16	6	6	4	
32	8	8	8	8

Поля компонент цвета могут быть назначены для RGB или другой цветовой модели в зависимости от потребностей приложения. При 8-ми битных пикселах до 8 бит могут использоваться для интенсивности, а оставшиеся биты могут использоваться для любых других атрибутов.

Для поддержки 3D операций отображения в графическом устройстве имеется дополнительный режим. Сравнения в Z-буфере (для 16 и 32 бит) обеспечиваются аппаратурой. Интерполяция цветов и Z значений облегчают реализацию алгоритмов закраски. Может быть использована любая комбинация размера пиксела и глубины Z-буфера. Скорость закраски Гуро достигает 50 000 треугольников в сек для треугольников из 100 8-битных пикселов.

В состав i860 входят четыре файла регистров:

- файл из 32 32-битных целочисленных регистров, используемых для адресации и скалярных целочисленных вычислений;
- файл из 32 32-битных регистров процессора с плавающей запятой. Они используются в операциях вещественной арифметики и при выполнении графических целочисленных операций. Эти регистры могут использоваться как 64-х или 128-битные. Файл регистров ППЗ имеет два порта чтения, порт записи и два двунаправленных порта. Все порты 64-битные и могут работать одновременно.
- файл регистров управления;
- файл специальных регистров.

Пиковое быстродействие i860 в 40 MIPS и 80 MFLOPS хорошо соответствует высокоскоростной геометрической обработке. Достигается скорость до 500 000 однородных преобразований векторов. Большая ширина шины данных (64 бита внешняя и несколько внутренних 64-битных параллельных шин) обеспечивают высокоскоростной доступ к памяти.

0.2.5 Высокоскоростные графические системы

Требования к высокоскоростным графическим системам

В дополнение к задачам растеризации, высокоскоростные графические машины требуют сбалансированной обработки моделирования, геометрических вычислений, освещенности и свойств материала. Эти требования необходимы для:

- реального времени: изображение должно генерироваться в 1/30 секунды для обеспечения малого времени ответа на ввод от пользователя а также для отображения "живых" сцен.
- реализм: многие приложения требуют возможностей генерации высококачественных картин (фотореализм),
- стандарты: с повышением сложности графического программного обеспечения важна поддержка стандартизованных интерфейсов и систем типа PHIGS, X Window System и т.д. с целью обеспечения переносимости программного обеспечения.

Система Silicon Graphics POWER IRIS 4D/380 VGX

POWER IRIS 4D/380VGX (рис. 0.2.14) фирмы Silicon Graphics является одной из наиболее мощных суперстанций. Она представляет собой мультимикропроцессорную систему, построенную на 8 RISC процессорах MIPS RS3000 с сопроцессорами плавающей запятой RS3010, разделяющих общую память. Система предоставляет более 200 MIPS и более 30 MFLOPS.

Графическая подсистема VGX системы IRIS VISION отображает в секунду более 1 миллиона Z-буферизованных треугольников, закрашенных по Гуро или Фонгу, со скоростью доступа к кадровому буферу более 20 млн. пикселов/с. Она содержит 85 соответствующих СБИС десяти различных типов. VGX архитектура содержит 4 подсистемы объединенных в конвейер:

- геометрическую подсистему,
- подсистему растеризации,
- растровую подсистему,
- дисплейную подсистему.

Геометрическая подсистема подсоединена к параллельному ЦП через 64-битный шинный интерфейс MPlink. Она принимает геометрические данные, определенные в мировых координатах, выполняет преобразования и вычисление освещенности и передает данные в пространстве экрана подсистеме растеризации. Компонентами геометрической подсистемы являются Командный Процессор и SIMD Геометрическая Машина. Командный Процессор принимает данные из шины MPlink и распределяет команды между четырьмя модулями Геометрической Машины. Он преобразует все принятые слова данных в форму с плавающей запятой вне зависимости от представления входных данных. Геометрическая Машина содержит четыре идентичных процессорных устройства с плавающей запятой, использующих чипы Texas Instument 74ACT8867, дающими пиковую скорость в 128 MFLOPS. Эти 4 устройства вместе работают как SIMD машина под управлением секвенсора, одновременно обрабатывая до четырех вершин многоугольника.

Подсистема растеризации выполняет вычисления по преобразованию данных по вершинам, переданных геометрической подсистемой, в индивидуальные пикселы. С каждым пикселом увязаны X,Y экранные координаты и ряд параметров, включающих Z-координату, R, G, B, Alpha и текстурные данные. Все эти параметры линейно интерполируются между вершинами многоугольника. Подсистема растеризации содержит четыре процессора: сортировщик вершин, машину полигонов, машину области и Span процессор.

Первый процессор — сортировщик вершин упорядочивает поток команд и данных с плавающей запятой, принятых от Геометрической Машины, в точки, отрезки, треугольники и т.д. и передает от одного до четырех примитивов Машине Полигонов. Он также сортирует треугольники и квадрилатеральные вершины для последующих вычислений.

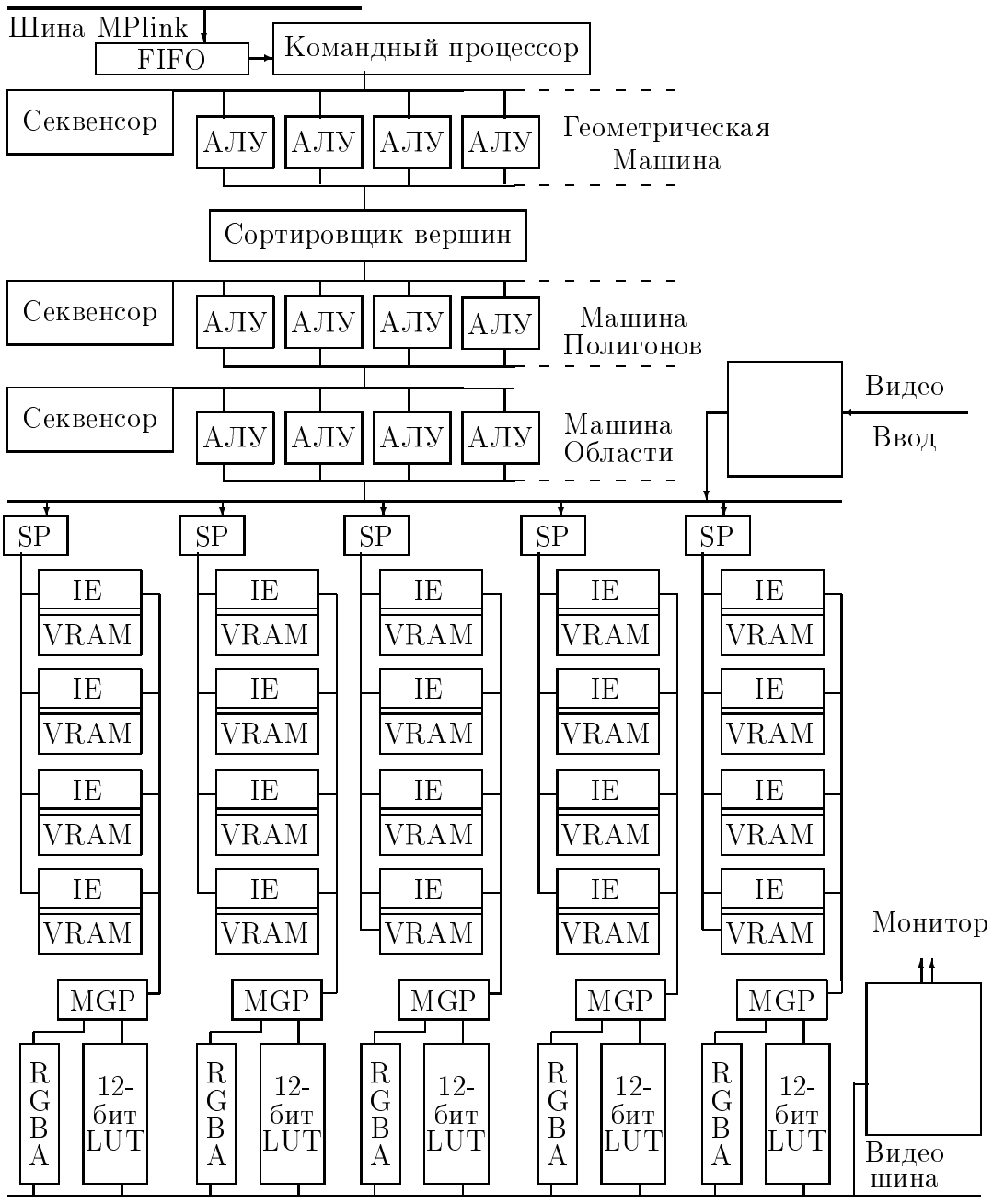

Рис. 0.2.14: Архитектура системы POWER IRIS 4D/380VGX

Машина полигонов представляет собой SIMD процессор с плавающей запятой, содержащих четыре устройства, и структурно подобна Геометрической Машине. Она начинает преобразование сканирования треугольников вычисляя углы наклона сторон и параметры в X и Y направлении.

Машина областей представляет собой SIMD процессор с фиксированной запятой, также содержащий четыре устройства, работающих под управлением микропрограммируемого секвенсора. Она преобразует исходные значения с плавающей запятой и информацию об углах для формирования X,Y и значений параметров для вертикального расстояния между нижним и верхним ребрами треугольника. Она передает эту информацию с углами наклона в Y-направлении Span-процессору.

Параметры расстояний и углов принимаются от 5 до 10 Span процессорами. В базовой конфигурации с пятью Span-процессорами, каждый процессор управляет 1/5 из видимых 1280 столбцов экрана. Span-процессор номер 0 управляет столбцами 0, 5, 10 и т.д., процессор номер 1 управляет столбцами 1, 6, 11 и т.д. В связи с тем, что полосы, генерируемые через некоторый треугольник, всегда являются близлежащими, то обработка полос равномерно распределяется между пятью процессорами. Конфигурация с 10 процессорами распределяет загрузку подобным образом, но каждый Span-процессор управляет каждым 10-м столбцом.

Каждый Span процессор итерирует между заданным и конечным Y адресами, используя данные по углам для генерации Z-координаты, цветовых и текстурных координат в заданном столбце. Эти значения передаются растровой подсистеме. Растровая подсистема содержит 20 или 40 Машин образов, каждая из которых контролирует прямоугольную область экрана. В конфигурации с 20-ю машинами отдельная машина образов имеет доступ к 64 K пикселов. Каждый пиксел содержит, по крайней мере, 140 бит данных, т.е. кадровый буфер должен иметь емкость в 23 Мбайта. Растровая подсистема выполняет чтение, запись и копирование между банками кадрового буфера со скоростью до 20 млн. пиксел/сек. 140-битный пиксел устроен следующим образом: кадровый буфер запоминает по одному байту для R, G, B и Alpha-информации для каждого пиксела плоскости отображения дисплея размером 1280×1024. В режиме двойной буферизации это суммируется до 64 бит. Система VGX поддерживает 24-битный Z-буфер со знаком. Девять бит глубины банка служат для различения раскрашенных планов. Машины образов имеют возможности условного стирания, записи, инкремента и декремента этих планов независимо от 24-битного значения глубины. 32 бита используются в качестве текстурных планов для R, G, B и Alpha информации. 9 бит используются для перекрытия и оконных планов, поддерживаемых для использования системой управления окнами. Последние 4 бита, называемые Display WID планы, определяют стиль отображения каждого пиксела, например, однократное буферирование или двойное буферирование, RGB или индекс цвета.

Дисплейная подсистема принимает пиксельную информацию из кадрового буфера, маршрутизирует ее в соответствии с требуемым режимом отображения и передает ее на отображение.

Система Stardent GS2000

Системы типа Silicon Graphics POWER IRIS используют специализированную аппаратуру для быстрого вычисления изображений. Многие из этих вычислений, в частности геометрические преобразования, близки к обработке, требуемой во многих видах научных вычислений. Stardent GS2000 разработана для предоставления ее вычислительных ресурсов как для генерации изображений, так и для ускорения других вычислительно интенсивных заданий, которые могут вызываться на машине.

Stardent GS2000 включает следующие типы процессоров (рис. 0.2.15):

- мультипотоковый процессор (MSP) — единое устройство, которое работает как четыре идентичных логических процессора. Каждый процессор или поток может выполнять некоторое множество машинных инструкций. Эти потоки обеспечивают скорость 20–25 MIPS;
- процессор векторных операций и операций с плавающей запятой (VFP) — отдельное устройство интегрированное в MSP. VFP обеспечивает скорость 80 MFLOPS на данных одинарной точности и 40 MFLOPS — на двойной. VFP играет важную роль в обработке графики, работая в качестве "front-end" для процессора отображения. Специальные векторные инструкции выполняют координатные преобразования (600 K 3D преобразований векторов в сек), отсечение и вычисления освещенности. Геометрические данные и результаты расчета освещенности передаются процессору отображения;
- процессор отображения (Rendering processor — RP) — микропрограммная машина, которая одновременно оперирует блоками 4×4 пиксела. Для отображения графических примитивов процессор отображения "прогуливается" поверх блоков пикселов, вычисляя различные 16-ти пиксельные блоки, пока не будут обработаны все пикселы, покрывающие примитив. Процессор отображения принимает от VFP результаты преобразования, отсечения и расчета освещенности примитивов, преобразует их в пикселы и запоминает результаты в виртуальной памяти в виде виртуальных пиксельных карт.

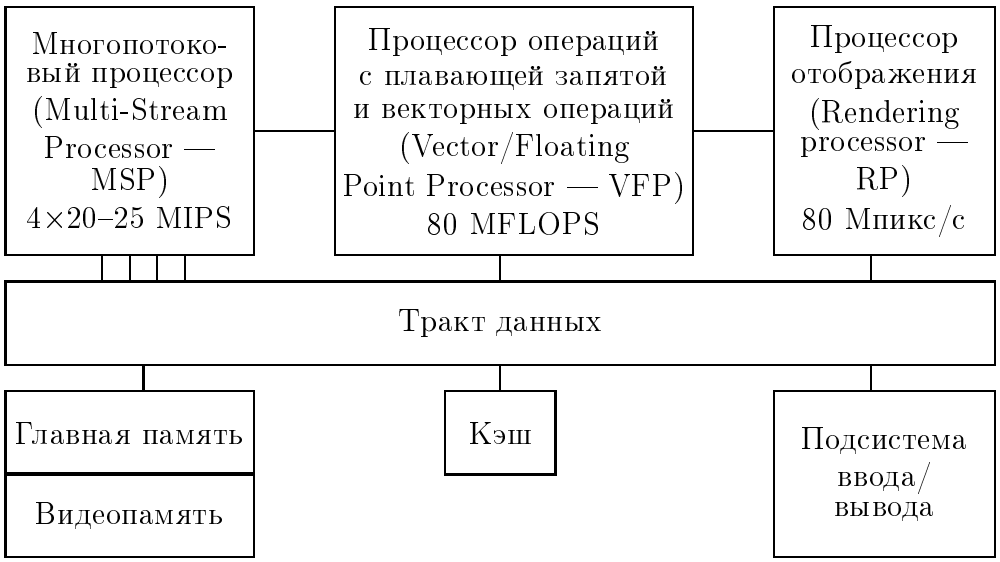

Рис. 0.2.15: Архитектура системы Stardent GS2000

Процессор отображения может читать, модифицировать и записывать изображения со скоростью до 80 Мпикселов/с. Он может вычерчивать 600 K 3D клиппируемых десятипиксельных векторов и 160 K 100 пиксельных треугольников Гуро с Z-буферизацией.

Каждое изображение читаемое и записываемое процессором отображения запоминается в виртуальной памяти как комплект из одной или более виртуальных пиксельных карт (virtual pixel maps — VMAPs) каждая до 64 K $\times$ 64 K пикселов с 32 битами на пиксел (128 Гбайт).

Это освобождает графическую подсистему от ограничений единственного кадрового буфера фиксированного разрешения. VMAP может листоваться на/с дисковой памяти, наподобие

других данных в виртуальной памяти. VMAP отображается при копировании видимой порции в видеопамять.

Видеопамять содержит 1024 строки по 1280 пикселов. Они могут быть глубиной 16 или 32 бита (40 Мбайт). Двойная буферизация делает возможным стереоотображение.

0.2.6 Выводы

Таким образом, с точки зрения разработчиков графических дисплейных систем, на примере продукции ведущих фирм, специализирующихся в области технических средств машинной графики, можно выделить три основных подхода:

- создание функционально завершенного ("закрытого" для разработчика) графического сопроцессора, аппаратно реализующего основные примитивы графических стандартов GKS/CGI (Intel 82786);
- предоставление разработчику средств строить архитектуры растровых графических дисплейных систем (с различными функциональными и скоростными характеристиками) на основе набора специализированных СБИС, выполняющих строго регламентированные функции (National Semiconductor);
- создание универсального (полностью программируемого на языке высокого уровня) процессора, аппаратно реализующего основные растровые операции и предоставляющего разработчику максимальную гибкость в реализации собственных алгоритмов (Texas Instruments, i860).

0.3 СТАНДАРТИЗАЦИЯ В МАШИННОЙ ГРАФИКЕ

Начальный период создания и развития средств машинной графики характеризовался развитием многочисленных, иногда достаточно эффективных, графических систем, ориентированных на то или иное оборудование [1, 42]. Фактически этот период можно охарактеризовать как период основного внимания к техническим средствам [2]. Программное обеспечение рассматривалось, как удачно заметил Гантер [18], чем-то вроде “верхнего слоя краски на аппаратуре”.

В следующий период более актуальной стала проблема создания программного обеспечения. Во-первых, велись разработки алгоритмов машинной графики — генерация примитивных элементов, интерполяция, аппроксимация, формы и методы представления изображений и т.д.; во-вторых, создавались инструментальные средства машинной графики — графические языки, пакеты процедур, языки диалога.

Постепенно сформировалось представление о программном продукте как о промышленном изделии, что выдвинуло проблему стандартизации графического программного обеспечения. Развитие сетей ЭВМ, оснащенных терминальными устройствами различных типов, потребовало обеспечить независимость программного обеспечения от аппаратуры.

0.3.1 NGP (Network graphics protocol)

Первые результаты по стандартизации были получены применительно к сети ARPA в рамках работ по разработке протоколов для аппаратно и машинно-независимого представления графических данных в сети [116].

Модель работы пользователя в сети с применением графического протокола [116] приведена на рис. 0.3.1.

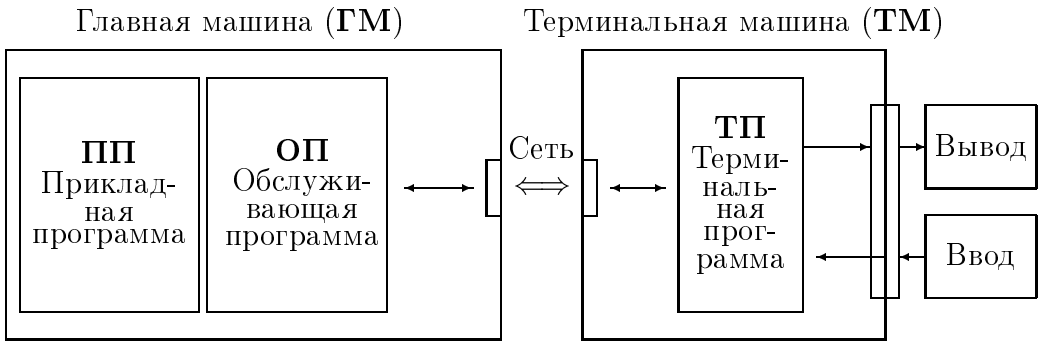

Рис. 0.3.1: Модель организации работы в сети

В начале сеанса работы пользователь располагается перед дисплеем, подключенным к терминальной ЭВМ, инициирует терминальную программу и устанавливает связь с ЭВМ сети, на которой ему будет предоставлено необходимое обслуживание. Прикладная программа главной ЭВМ при необходимости выполнить ввод/вывод использует обслуживающую программу, функцией которой является интерфейс прикладной программы с сетевым графическим протоколом.

Терминальная программа используется для интерпретации протокола в подходящую аппаратно-зависимую форму и для реализации функций ввода и запроса к обстановке.

Данные в сети передаются только в стандартной форме, следовательно, на передающей стороне выполняется кодирование, а на приемной — декодирование информации. Предусмотрено

два уровня протокола вывода: сегментированный и структурированный форматы. В сегментированном формате изображение строится из отдельных сегментов, представляющих собой список графических примитивов (точек, линий, строк текста). ТП выполняет только перекодировку и может быть достаточно простой. В структурированном формате изображение строится из вызовов отдельных подкартин, состоящих из примитивов и вызовов других подкартин. Объем передаваемых данных уменьшается, но на ТМ, как правило, требуется программное выполнение преобразований.

Для работы с виртуальными устройствами ввода, служащими для выполнения позиционирования, ввода скалярного значения, ввода состояния кнопки (вкл/выкл), ввода строки символов, ввода времени, используется два метода: 1) запроса и получения состояния устройства ввода, например, строки символов; 2) разрешения пользователю совершить действия, приводящие к возникновению события ввода и получения на главной ЭВМ "сообщения" события.

Аппаратная независимость обеспечивается средствами опроса, который позволяет выяснить конфигурацию и возможности используемых устройств. Адаптация к возможностям реализуется необходимыми настройками прикладной и обслуживающей программ.

После публикации [116] появился целый ряд работ, посвященных использованию идей протокола в нашей стране [4, 8, 15, 22].

0.3.2 Международная деятельность по стандартизации в машинной графике

Работы по протоколам послужили отправной точкой по развитию стандартизации в машинной графике. В 1974 г. в США был создан комитет по стандартизации машинной графики GSPC в ACM/SIGGRAPH. В 1975 г. в ФРГ в Институте стандартов был создан подкомитет по машинной графике — DIN-NI/UA-5.9. В 1977 г. в международной организации по стандартизации (ISO) была создана рабочая группа TC97/SC5/WG2 "машинная графика" [70].

Важную роль в разработке методологии стандартизации машинной графики сыграла конференция в Сейлаке (Франция), организованная графическим подкомитетом WG 5.2 IFIP в 1976 г. На конференции были сформулированы и обсуждены основные условия и проблемы стандартизации. Было установлено, что основная цель стандартизации — переносимость графических систем, которая достигается стандартизацией интерфейса между графическим ядром системы (базовой графической системой), реализующим собственно графические функции, и моделирующей системой — проблемно-ориентированной прикладной программой, использующей функции графического ядра. Базовая система должна обладать: независимостью от вычислительных систем; независимостью от языков программирования; независимостью от области применения; независимостью от графических устройств.

Структура прикладной графической системы, удовлетворяющей сформулированным требованиям, может быть представлена в виде шестиуровневой модели (рис. 0.3.2).

Процесс преобразования информации при выполнении вывода может быть представлен состоящим из следующих этапов (рис. 0.3.3):

1. Модельные преобразования. Проблемно-ориентированный уровень из геометрических моделей отдельных объектов, задаваемых в собственных локальных системах координат, формирует описание совокупного объекта в некоторой единой (мировой) системе координат. Описание совокупного объекта подается в графическую систему.

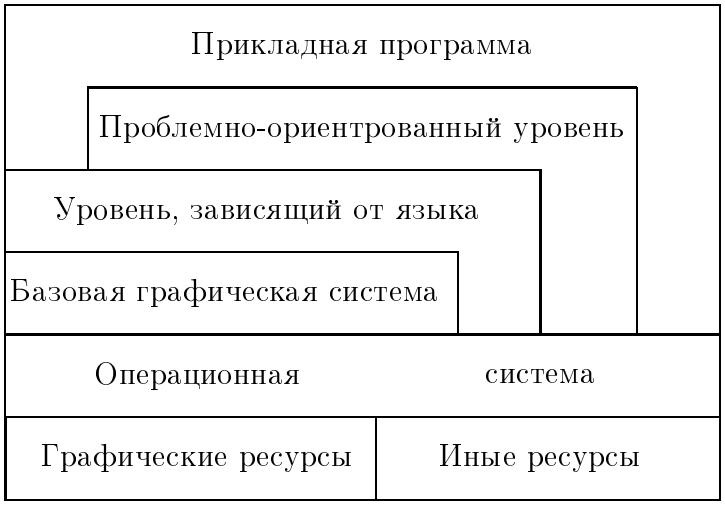

Рис. 0.3.2: Уровневая модель прикладной графической системы

2. Нормализующие преобразования. Графическая система переводит описание из мировой, вообще говоря произвольной, системы координат в т.н. нормализованные координаты устройства, имеющие фиксированные пределы изменения координат, например, от 0.0 до 1.0.

3. Преобразования сегментов. Если графическая система предоставляет средства манипулирования отдельными подкартинами изображения (часто именуемыми сегментами), например, для независимого размещения отдельных самостоятельных частей изображения, то могут потребоваться такие преобразования.

4. Видовые преобразования. В случае 3D описания изображения и 2D устройства вывода необходимо выполнить проецирование изображения на заданную картинную плоскость. Наоборот, при 2D сцене и 3D устройстве вывода необходимо выполнить преобразование, связанное с размещением изображения. При выполнении этих преобразований, естественно, может потребоваться выполнение отсечения частей изображения. После этого этапа по сути дела готово описание изображения в некоторой аппаратно-независимой форме, пригодной для вывода на

любое устройство.

5. Преобразование рабочей станции. Для выполнения вывода на конкретное устройство необходимо преобразование данных из аппаратно-независимой формы в координаты устройства.

Процесс преобразования координатной информации при вводе от координатных устройств обратен вышеописанному преобразованию при выводе.

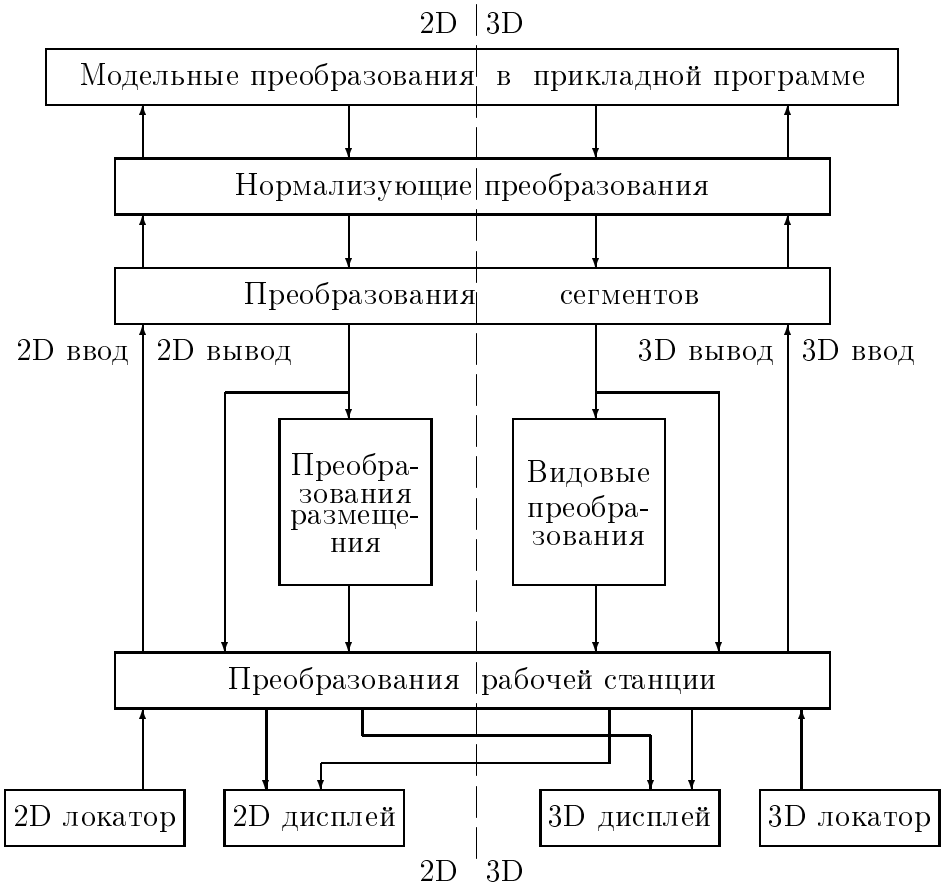

Рис. 0.3.3: Схема преобразований координатной информации в графической системе

Концептуальная модель переносимой графической системы показана на рис. 0.3.4. Штриховые линии на нем обозначают интерфейсы, при стандартизации которых может быть обеспечена переносимость.

Верхний уровень стандартизации — IGES предназначен для обеспечения мобильности компонент САПР.

Средний уровень стандартизации — уровень базового графического пакета (GKS) определяется выбором базовых функций системы. Этот интерфейс делает базовую графическую систему независимой от области применения.

Нижний уровень стандартизации — уровень связи с виртуальным графическим устройством (CGI) зависит от выбора примитивов ввода/вывода, являющихся абстракцией возможностей устройств. Этот интерфейс делает базовую графическую систему аппаратно-независимой.

Независимость от вычислительных систем и языков программирования обеспечивается соответствующей дисциплиной программирования и взаимодействия с системами программирования.

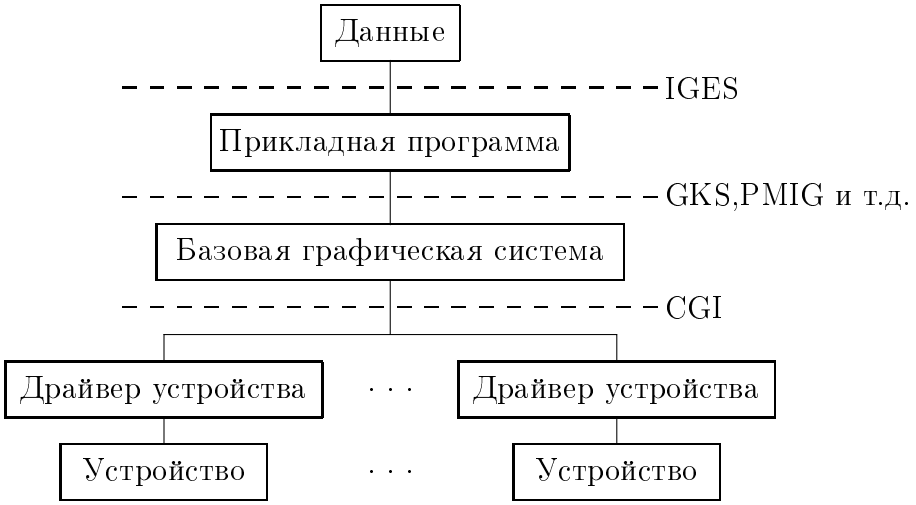

Рис. 0.3.4: Архитектура переносимой графической системы

0.3.3 Деятельность ISO, IEC по стандартизации в машинной графике

Главными организациями формирующими международные стандарты в области информационной технологии являются ISO (International Organization for Standartization) и IEC (International Electrotechnical Commission). В конце 1987 г. был сформирован первый совместный технический комитет (JTC1) ISO/IEC с целью стандартизации в области информационной технологии. Стандартизацией в машинной графике занимается 24-й подкомитет (ISO/IEC JTC1/SC24). В 1988 г. была создана постоянная советская часть этого подкомитета. Основными стандартами являются [68]:

1. GKS (Graphical Kernel System) — набор базовых функций для 2D аппаратно-независимой машинной графики.

2. GKS-3D (Graphical Kernel System for 3 Dimensions) — расширение GKS для поддержки базовых функций в 3D.

3. PHIGS (Programmer's Hierarchical Interactive Graphics System) — набор базовых функций 3D графики аналогичный GKS-3D, но в отличие от GKS-3D, ориентированной на непосредственный вывод графических примитивов, группируемых в сегменты, графическая информация накапливается в иерархической структуре данных. В целом PHIGS ориентирован на приложения, требующие быстрой модификации графических данных, описывающих геометрию объектов.

4. Языковые интерфейсы (Language bindings) — представление функций и типов данных функциональных графических стандартов в стандартизованных языках программирования.

5. CGM (Computer Graphics Metafile) — аппаратно-независимый формат обмена графической информацией. Используется для передачи и запоминания информации, описывающей изображения.

6. CGI (Computer Graphics Interafce) — набор базовых элементов для управления и обмена данными между аппаратно-независимым и аппаратно-зависимым уровнями графической системы.

7. CGRM (Computer Graphics Reference Model) — модель стандартов в машинной графике, которая определяет концепции и взаимосоотношения применительно к будущим стандартам в машинной графике.

8. Регистрация — механизм регистрации стандартизуемых аспектов примитивов вывода, обобщенных примитивов, escape-функций (для доступа к аппаратным возможностям устройств) и других графических элементов.

9. Тестирование реализаций на соответствие графическим стандартам — основные цели этого проекта: специфицирование характеристик стандартизованных тестов, используемых для определения соответствия реализаций графическим стандартам, и выработка предписаний разработчикам функциональных стандартов относительно правил соответствия.

В составе 24-го подкомитета имеется 5 рабочих групп (WG):

WG1: Архитектура. Цель этой группы — развитие CGRM — модели стандартов машинной графики.

WG2: Интерфейсы прикладных программ. Стандартизация функциональных спецификаций для интерфейсов прикладных программ.

WG3: Метафайлы и интерфейсы с устройствами. Стандартизация обмена графической информацией, включая метафайл и интерфейс с устройствами.

WG4: Языковые интерфейсы. Стандартизация языковых интерфейсов для функциональных стандартов машинной графики.

WG5: Верификация, тестирование и регистрация. Разрабатывает методы и процедуры проверки соответствия и тестирования реализаций функциональных стандартов машинной графики и методов и процедур регистрации графических примитивов.

0.3.4 Классификация стандартов

Из рис. 0.3.4 видно, что для обеспечения мобильности программного обеспечения требуется стандартизовать:

- базовую графическую систему, т.е. стандартизовать графический интерфейс (набор базовых графических функций) — Core System, GKS, GKS-3D, PMIG, PHIGS, PHIGS+ и т.д.
- графический протокол (порядок и правила обмена информацией) — IGES, CGM и др.

Далее будут рассмотрены графические интерфейсы, являющиеся международными графическими стандартами, а затем — графические протоколы, среди которых большая часть — стандарты де-факто и только один — CGM — международный стандарт.

0.3.5 Core-System

Существенным этапом в области стандартизации машинной графики явилась публикация проекта стандарта CORE-SYSTEM (GSPC-77) [118], модель которой приведена на рис. 0.3.5. Главные идеи, положенные в основу системы CORE [118, 119]: разделение функций ввода и вывода; минимизация отличий между выводом на графопостроитель и интерактивный дисплей; концепция двух координатных систем — мировой системы координат, в которой конструируется выдаваемое изображение, и приборной системы координат, в которой представляются данные для отображения; концепция дисплейного файла, содержащего приборную координатную информацию; понятие дисплейного файла сегментов, каждый из которых может независимо модифицироваться как элемент; обеспечение функций преобразования данных из мировой системы координат в приборную путем вызова видового преобразования.

В системе выделены следующие группы функций: вывода; сегментирования дисплейного файла; установления и опроса атрибутов примитивов (цвет, яркость, ширина линии и т.д.) и

атрибутов сегментов (тип, видимость, указуемость и т.д.); визуализации; выполнения ввода с виртуальных устройств ввода типа указка, клавиатура, кнопка, локатор, датчик; управления и доступа к специальным аппаратным возможностям.

В 1979 г. был опубликован уточненный проект стандарта CORE-SYSTEM (GSPC-79) [119]. Кроме прочих изменений, в этой версии предусмотрена (весьма ограниченно) поддержка растровых устройств. Всего предлагалось 266 функций, так что охватывался широкий спектр применения машинной графики, начиная от пассивного вывода до интерактивных систем высокого уровня.

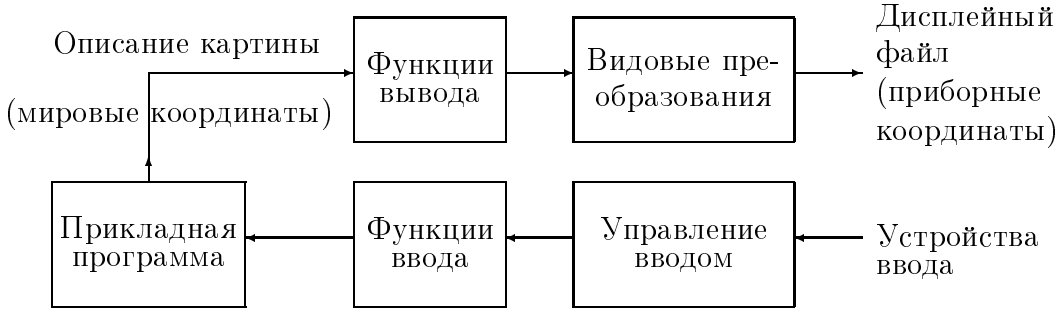

Рис. 0.3.5: Модель графической системы, положенная в основу CORE-SYSTEM

Ясно, что для многих приложений требуется лишь часть возможностей графпакета, все остальные, если будут присутствовать, будут приводить к неэффективности прикладной программы. Для устранения этого противоречия система разбита на три не зависящие друг от друга группы уровней — группа уровней вывода, группа уровней ввода, группа уровней размерности. Уровни внутри группы совместимы снизу-вверх.

Группа уровней вывода включает в себя:

- базовый вывод, поддерживающий полный набор примитивов вывода, их атрибутов и операций визуализации и предназначенный для приложений, не требующих выборочного редактирования изображений;
- буферизованный вывод дополнительно к предыдущему уровню позволяет использовать сохраняемые сегменты без преобразования образа — преобразования графической информации, содержащейся в сегменте в момент выполнения вывода.

Группа уровней ввода включает:

- без ввода, т.е. применяется для пассивных приложений;
- синхронный ввод — ввод производится синхронно с работой прикладной программы, т.е. ее исполнение приостанавливается до завершения ввода;
- асинхронный ввод — ввод производится независимо от работы прикладной программы, а вводимые оператором данные накапливаются в обрабатываемой прикладной программой очереди ввода.

На последних двух уровнях поддерживаются виртуальные устройства ввода классов ЛОКАТОР, ШТРИХ, ДАТЧИК, ВЫБОР, УКАЗКА и КЛАВИАТУРА.

Группа уровней размерности включает:

- 2D — поддерживаются только 2D операции;
- 3D — дополнительно к 2D поддерживаются и 3D операции.

Предложения GSPC получили широкий отклик в виде многочисленных реализаций версий базовой системы [7, 40, 49, 96, 123].

0.3.6 GKS (Graphical Kernel System)

Результатом работ в ФРГ было создание системы GKS. Модель графической системы, положенная в ее основу, приведена на рис. 0.3.6. В 1979 г. GKS была принята в качестве отправной точки международного стандарта. В процессе разработки международного стандарта в исходную версию GKS был внесен целый ряд изменений, приблизивших ее к CORE-SYSTEM, но сохранивших ряд отличительных фундаментальных концепций. Основной из таких отличительных черт является введение понятия рабочей станции, представляющей собой абстракцию совокупности виртуальных устройств ввода/вывода, более полно соответствующей современной тенденции использования интеллектуальных терминалов.

По максимуму рабочая станция обладает следующими свойствами: имеет одну адресуемую поверхность вывода с фиксированным разрешением; допускается только прямоугольное поле отображения, которое не может состоять из нескольких отдельных частей; позволяет задание и использование поля отображения, которое меньше максимально возможного, с гарантией того, что изображение не генерируется вне заданного поля отображения; поддерживает несколько типов линий, шрифтов текста, размеров символов и т.д. для вывода примитивов с различными атрибутами; имеет одно или больше устройств ввода для каждого класса устройств, которые независимо друг от друга могут работать в одном из трех режимов — синхронный ввод, опрос, асинхронный ввод; запоминает сегменты и обеспечивает средства для изменений сегментов и манипуляций с ними.

Ясно, что выделение рабочей станции излишне громоздко, когда приходится иметь дело с одним устройством вывода и ввода. Основная ценность введения понятия рабочей станции состоит в удобной и естественной возможности разделения аппаратно-независимой и аппаратно-зависимой частей.

Следует отметить, что в GKS столь же последовательно проводится разделение функций ввода/вывода, как и в CORE.

Набор примитивов GKS подобен набору примитивов CORE, хотя меньше и несколько более приспособлен для целей растровой графики (ломаная; полимаркер; текст; многоугольник, который может быть “залит” одним цветом, заштрихован, покрыт узором, либо может быть не закрашен за исключением границ; массив прямоугольных ячеек, закрашенных одним цветом; обобщенный примитив черчения, дающий доступ к специальным геометрическим и графическим возможностям устройств).

В отличие от CORE в GKS нет понятия текущей позиции, так что построение примитива не зависит от предыстории вычерчивания. В GKS имеется 2 способа задания атрибутов примитивов. Первый — индивидуальный способ аналогичен используемому в CORE, не зависит от рабочей станции и основан на индивидуальном задании атрибутов, которые сохраняют свое значение пока не будут изменены прикладной программой. Второй — групповой способ зависит от рабочей станции и основан на задании независимых групп значений атрибутов. Для использования требуемой группы атрибутов задается ее номер, который сохраняет свое значение пока не будет переустановлен.

Видовые операции, определяющие переход от входных (мировых) координат к физическим координатам устройства, также похожи, но в GKS, в отличие от CORE, допускается наличие не одной, а нескольких пар окно/поле вывода.

Средства сегментации GKS напоминают средства сегментации CORE, но в GKS добавлены средства манипулирования сегментами на заданной рабочей станции и средства работы с рабочей станцией специального типа — приборно-независимым хранилищем сегментов (ПНХС).

Виртуальные устройства ввода также подобны, хотя названия несколько отличаются и в GKS правила работы с устройствами лучше проработаны.

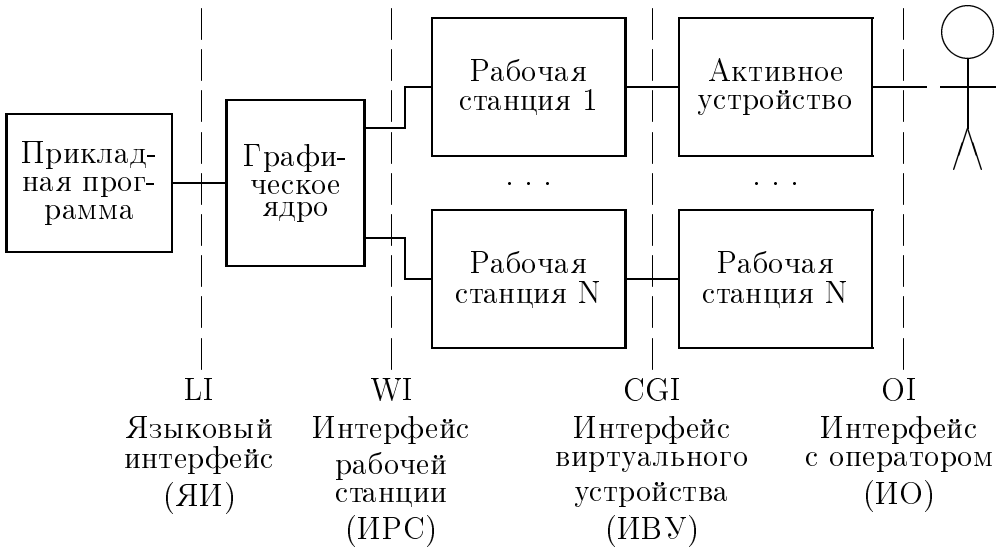

Рис. 0.3.6: Модель графической системы, положенная в основу GKS

Для повышения эффективности конкретных приложений в GKS, аналогично CORE, предусмотрено разбиение на уровни. Предлагается три уровня вывода и три уровня всех остальных функций.

Уровни вывода:

0: Минимальный вывод. Доступны все примитивы. Используются только групповые, немодифицируемые атрибуты примитивов. В каждый момент времени активна только одна рабочая станция вывода. Опросы параметров системы — только базовые. Доступно чтение пикселов — отдельных точек изображения из растровых устройств.

1: Базовая сегментация с полным выводом. Все возможности уровня 0. Полное управление рабочей станцией. Полные возможности вывода. Полные возможности групповых таблиц атрибутов. Одновременно могут быть активными несколько рабочих станций. Есть рабочие станции типа метафайл для сохранения/воспроизведения изображений. Может использоваться требуемое число пар окно/поле вывода. Есть базовая сегментация (без ПНХС). Доступны требуемые опросы.

2: Приборно-независимое хранилище сегментов. Все возможности уровня 1. Имеется ПНХС.

Уровни ввода:

A: Нет ввода.

B: Синхронный ввод. Существуют виртуальные устройства ввода каждого класса, работающие в режиме синхронного ввода (request).

C: Полный ввод. Все возможности ввода уровня B. Предусмотрены режимы асинхронного ввода и ввода по опросу (event и sample).

В 1985 г. GKS была принята в качестве международного стандарта [86]. В 1988 г. для стандартизованных языков программирования были приняты международные стандарты на интерфейс GKS с языками Fortran, Pascal, Ada, C [89].

0.3.7 GKS-3D (Graphical Kernel System for Three Dimensions)

Отличия GKS-3D от GKS заключаются в добавлении 3D функций:
- примитивов 3D вывода;
- установки атрибутов вывода (2 функции);
- поддержки 3D преобразований (9 функций);
- работы с 3D сегментами и преобразований 2D сегментов в 3D и наоборот (4 функции);
- ввода с 3D координатных устройств (10 функций);
- утилит работы с матрицами 3D преобразований (2 функции).

В целом GKS может рассматриваться как подмножество GKS-3D, т.е. 2D приложения, написанные на GKS, гарантированно исполняются в 3D окружении без каких-либо изменений.

В 1988 г. GKS-3D была принята в качестве международного стандарта [90]. В период с 1988 по 1990 гг. прорабатывались проекты стандартов ISO 8806 на интерфейсы GKS-3D с языками программирования. В 1991 г. завершилось их принятие в качестве международных стандартов [91].

0.3.8 PHIGS (Programmer's Hierarchical Interactive Graphics System)

Во многих приложениях возникает необходимость геометрического моделирования трехмерных тел (САПР машиностроительного, строительного, архитектурного и других направлений, системы автоматизации научных исследований, системы визуализации и т.д.).

Использование GKS или GKS-3D для отображения результатов моделирования предполагает, что моделирование целиком должна выполнять прикладная программа, так как эти системы ориентированы на прямой ввод/вывод и в них не предусмотрено иного манипулирования графическими данными кроме накопления в сегментах.

PHIGS [68, 92, 93, 102, 83] же комбинирует графику с техникой моделирования и представляет собой набор функций программирования графики с поддержкой быстрой модификации графических данных, описывающих геометрические соотношения объектов.

Набор примитивов вывода и их атрибутов в PHIGS тот же самый, что и в GKS-3D с добавлением примитива "annotation text relative" для пометки объектов.

Принципиальное отличие PHIGS от GKS состоит в том, что в PHIGS создание и отображение изображения разделены на две независимых фазы — занесения в централизованную структурную память (CSS — Centralized Structure Store) и отображения заданной структуры на требуемую рабочую станцию.

Структура состоит из графических и неграфических элементов. Поддерживаются возможности создания и редактирования структур в CSS. Структура может вызывать другие структуры за счет чего создаются иерархии. Предусмотрено 9 классов структурных элементов:

примитивы вывода;	спецификация атрибута;
локальное преобразование моделирования;	редактирование;
обобщенный структурный элемент;	прикладные данные;
глобальное преобразование моделирования;	управление;
отсечение.	

Графический вывод задается в т.н. модельных координатах. Элемент "локальное преобразование моделирования" задает преобразование, которое применяется к последующим примитивам вывода.

Элемент "редактирование" определяет метку, используемую при редактировании и несущественную при отображении. Предоставлены возможности поиска метки и установки на нее указателя.

"Обобщенный структурный элемент" позволяет определить нестандартные действия.

Элемент "прикладные данные" несущественен при отображении. Обычно представляет собой указатель на прикладную базу данных.

Основное отличие между системами PHIGS и GKS-3D заключается в наличии элемента "Исполнить структуру" (Execute Structure), позволяющего при отображении исполнить структуру как часть другой. Когда такой элемент принимается интерпретатором, то текущее состояние упрятывается и управление переключается на структуру, заданную в элементе. После завершения интерпретации вызванной структуры восстанавливается исходное состояние и интерпретация продолжается с элемента, следующего за элементом "Исполнить структуру".

При вызове структуры она наследует глобальное модельное преобразование вызвавшей структуры. Даже если при исполнении вызванной структуры глобальное модельное преобразование будет изменено с помощью соответствующих функций, тем не менее по возврату исходное глобальное преобразование будет восстановлено.

Локальное модельное преобразование, комбинируясь с глобальным, формирует текущее модельное преобразование.

В PHIGS может быть отредактирован отдельный графический элемент структуры, что обеспечивает большую скорость модификации изображения по сравнению с GKS, в которой, как минимум, требуется перегенерация сегмента, содержащего изменяемый примитив.

Отображение в PHIGS

PHIGS — 3D система, хотя ряд функций позволяет работать в 2D. При этом все функции преобразуются в эквивалентные 3D вызовы.

В отличие от ряда подобных систем, все координатные системы в PHIGS — правые координатные системы с возрастанием координаты по оси Z при движении по направлению к наблюдателю.

При интерпретации структуры из ее элементов генерируются примитивы вывода с использованием текущего модельного преобразования. Координаты сгенерированных примитивов вывода заданы в мировой системе координат (World Coordinates — WC).

Отображение в PHIGS выполняется в две стадии. На первой, называемой View Orientation (ориентация отображения), координатная система изменяется к необходимой для отображения выводимого объекта. На второй стадии, называемой View Mapping (преобразование отображения), объект преобразуется в аппаратно-независимые нормализованные координаты устройства, готовые для вывода на каждой открытой рабочей станции. Проецирование выполняется на этой стадии.

View Orientaion изменяет мировую систему координат к системе координат точки наблюдения (View Reference Coordinate system — VRC). Суть состоит в переносе начала координат в точку фактического наблюдения объекта и с направлением осей согласно требуемой проекции.

View Mapping более сложна и зависит от выбранного типа проекции — "параллельной" или "перспективной". При параллельной проекции объем видимости определяется как парал-

лелепипед преобразуемый в порт проекции, который представляет собой прямоугольный параллелепипед в нормализованных координатах проекции с ребрами параллельными осям. При перспективной проекции объем видимости — конус.

Для каждого типа проекции плоскость проекции — окно, определяемое X-Y размерами параллелепипеда или конуса. Плоскость проекции перпендикулярна Z-оси. Для параллельных проекций угол, который образуют проекторы с плоскостью проекции, определяется точкой центра проекции (Projection Reference Point — PRP). Проекторы параллельны линии из PRP в центр окна на плоскости проекции. При перспективной проекции все проекторы проходят через PRP.

Не требуется, чтобы точка проецирования (PRP) лежала на оси Z. В общем случае все типы проекций могут быть получены путем ориентации объекта относительно VRC осей и посредством размещения PRP относительно VRC-осей и плоскости проекции.

Для многих применений требуется одновременное наличие нескольких видов объекта. В PHIGS возможно определение множества видов одного или различных объектов. Для каждого вида SET VIEW REPRESENTATION (установить представление вида) задает как он будет показан на требуемой рабочей станции. Отсечение производится для создания части пространства нормазизованных координат проекции (Normalized Projection Coordinates — NPC), потенциально видимой на рабочей станции для этого вида.

Преобразования окно/порт рабочей станции близки к GKS в определении расположения пространства нормализованных координат проекции на рабочей станции и в одинаковости коэффициентов масштабирования вдоль направлений X и Y.

0.3.9 PHIGS+

PHIGS+ [94, 74, 81] — расширение PHIGS, имеющее дополнительные функциональные возможности для приложений, требующих учета освещенности, раскраски (интерполяции цветов по поверхности), а также дополнительные возможности по управлению отображением и новые примитивы для поддержки эффективного описания сложных поверхностей. Эти расширения были сформулированы как поправка к существующим частям 1–3 стандарта PHIGS и введением новой части ь 4 стандарта.

Поддержка освещенности и раскраски основана на предоставлении средств задания позиции источника света и наличии примитивов “с данными”, задающими вектора нормалей и цвета вершин.

Предоставлена возможность задания в структуре “ограничивающего бокса”. Когда он обрабатывается, то устанавливаются флаги отсечения (если ограничивающий бокс полностью вне области вывода) и отбраковки (если ограничивающий бокс меньше заданного размера). Этот дополнительный элемент структуры предназначен для управления пропуском исполнения структуры, образ котрой находится вне области вывода, и позволяет повысить скорость отображения.

0.3.10 CGI (Computer Graphics Interface)

Это стандарт ISO [66] на интерфейс между аппаратно-независимой частью графического программного обеспечения (базисной графической системой) и аппаратно-зависимой (драйверами). Этот интерфейс ранее (в рамках ANSI) назывался интерфейсом виртуального устройства.

Для эффективного использования аппаратных возможностей современных графических устройств набор функций CGI перекрывает аппаратно-реализуемые возможности и включает в себя следующие функции:
• управление устройством,
• вывод графических примитивов,
• изменение графических атрибутов,
• сегментация изображений,
• графический ввод,
• растровые операции.

Отличительными особенностями CGI (по сравнению со стандартами на интерфейс базисной графической системы) являются следующие: расширенный набор графических примитивов, одноступенчатое преобразование координат, увеличенное количество логических устройств ввода, наличие растровых операций. В целом набор функций CGI достаточно удобен для создания надстроенного над ним графического программного обеспечения. Последнее позволяет эффективно создавать на основе CGI различные базисные графические системы.

Следует отметить, что стандарт на интерфейс виртуального устройства ориентирован в первую очередь на унификацию способа взаимодействия с различными графическими устройствами и предназначен для системных, но не прикладных программистов. Он был утвержден после появления множества проектов по стандартизации программных интерфейсов базисной графической системы.

0.3.11 Графические протоколы

Анализ применяемых в настоящее время графических протоколов и проектов по их стандартизации позволяет выделить протоколы следующих типов:
• аппаратно-зависимые графические протоколы или команды графических устройств,

• аппаратно-независимые графические протоколы или метафайлы,
• прикладные графические протоколы,
• растровые графические файлы.

Аппаратно-зависимые графические протоколы

Аппаратно-зависимые графические протоколы разрабатываются фирмами, производящими графическое оборудование. Они представляют собой последовательность команд для построения изображений на устройствах выпускаемых данной фирмой. Для интерпретации таких протоколов не требуется дополнительных ресурсов если используется соответствующее устройство. Поэтому, такие протоколы могут успешно применяться в распределенных системах при отсутствии локальной ЭВМ.

Вопрос о поддержке тех или иных аппаратно-зависимых графических протоколов определяется составом используемого оборудования. Целесообразно, чтобы центральная ЭВМ обеспечивала возможность генерации команд для наиболее распространенных графических устройств. В настоящее время значительная часть производящейся в мире графической аппаратуры работает с протоколами TEKTRONIX, REGIS и HPGL. Поддержка этих протоколов обеспечивается также в наиболее распространенных зарубежных программных продуктах.

Протокол TEKTRONIX

Разработан одноименной фирмой, выпускающей графические дисплеи. Ввиду широкой распространенности устройств этой фирмы другие разработчики графической аппаратуры часто обеспечивают режим совместимости с TEKTRONIX'ом в своих устройствах. Поддержка этого протокола производится также в некоторых эмуляторах терминала (например, VTERM, ST240, TEEMTALK) работающих на персональных компьютерах типа IBM PC.

Существуют две разновидности протокола TEKTRONIX, соответствующие дисплеям серии 4010/12/14 и дисплеям серий 41XX, 42XX, 43XX. Дисплеи 4010/12/14 это дисплеи на запоминающей трубке с минимальным набором графических команд. Дисплеи серии 41XX и выше это цветные растровые дисплеи, работающие с более развитым графическим протоколом, который предусматривает следующие основные группы графических команд:

- команды построения векторных примитивов,
- команды работы с растровыми изображениями,
- команды управления сегментацией изображения,
- команды задания цветовых и геометрических атрибутов,
- команды графического ввода,
- команды управления плоскостями вывода,
- команды выполнения видовых преобразований,
- команды определения символов (графических образов).

Кроме перечисленных графических функций протоколы серии TEKTRONIX 41XX и старше включают множество функций управления алфавитно-цифровым режимом, каналом передачи данных и дополнительными внешними устройствами (hardcopy, планшет, диск).

Для представления команд используется символьное кодирование с использованием служебных символов в имени команды (чаще всего — символа Esc), что затрудняет чтение операторов человеком.

Протокол REGIS

Разработан для дисплеев серии VT (240 и выше). С этим протоколом работают также персональные компьютеры фирмы LabTam и ряд графопостроителей различных фирм. Поддержка протокола REGIS обеспечивается в некоторых эмуляторах терминала на IBM PC (VTERM, ST240). По функциональным возможностям протокол REGIS заметно уступает протоколу TEKTRONIXa. В частности, в нем гораздо беднее набор растровых операций, задания атрибутов построений, средств графического ввода и управления плоскостями вывода, полностью отсутствуют возможности сегментации изображения, выполнения видовых преобразований, определения символов.

Протокол HP-GL

Графический протокол HP-GL (язык описания данных Graphic Language) разработан фирмой Hewlett Packard в 1976 г. и поддерживается множеством других фирм, выпускающих графопостроители. В настоящее время используется версия HP-GL/2. Операторы языка содержат символьное имя команды и несколько параметров, также подготовленных в печатном текстовом виде. Всего в языке 88 операторов, разбитых на 9 функциональных групп. Ядро языка содержит 5 групп из 55 операторов, которые должны поддерживаться на всех устройствах.

Оставшиеся 3 группы из 33 операторов являются специфичными для некоторых из устройств. В целом, благодаря использованию явного текстового представления, язык легко читается и интерпретируется.

Языки описания страниц

Любая страница может быть описана как просто пиксельный массив, но это практически неприемлемо. Язык описания страниц должен описывать любой текст и графику на высоком уровне в терминах абстрактных графических элементов.

Выполнение вывода с использованием языка описания страниц идет в две стадии:

1. Приложение генерирует аппаратно-независимое описание на языке описания страниц.

2. Программа, управляющая некоторым растровым устройством вывода, интерпретирует описание и отображение его на устройство.

Эти две стадии могут быть выполнены в разное время и в разных местах.

Примитивы вывода выдаются на растровое устройство вывода процессом, называемым преобразованием сканирования (растеризация).

Язык PostScript

Особое место среди графических языков высокого уровня занимает интерпретируемый язык описания страниц PostScript [109, 108], разработанный фирмой Adobe и используемый не только для описания и построения изображений, но и качестве высокоуровневого аппаратно-независимого протокола обмена между компонующей и отображающей системами.

В первую очередь PostScript — это общий язык программирования с встроенными мощными графическими примитивами. С другой стороны, это язык описания страниц, который включает возможности языка программирования, и используется для связи с электронными печатающими устройствами.

На PostScript'е можно описывать любые графические формы, двухуровневые изображения и печатаемые формы. Для построения изображений (в том числе и всевозможных шрифтов) в языке PostScript предоставляется возможность управления каждой точкой печатающего устройства.

Естественно, что вследствие наглядности PostScript, как и другие языки программирования неоптимален в смысле минимальности кодирования информации. Поэтому его использование в качестве графического протокола представляется нецелесообразным. Однако он становится незаменим при передаче тексто-графических документов, предназначенных для воспроизведения на печатающих устройствах с высоким разрешением (например, лазерных принтерах).

Аппаратная независимость достигается тем, что построение изображения ведется в пользовательской системе координат с помощью обычных графических примитивов и описание изображения не содержит никакой информации об устройстве отображения.

В языке предусмотрен ряд типов данных — числа, строки, одномерные массивы, словари (таблицы, задающие соответствие между ключом и значением). Элементы массивов могут быть различных типов. Примитивы управления включают в себя условия, циклы и процедуры, которые могут вызываться рекурсивно.

Операторы (арифметические, логические, графические и управления) записываются в постфиксной записи и манипулируют со стеком типа LIFO.

В язык встроены следующие изобразительные возможности:

- изображения строятся из отрезков линий, дуг и кубических кривых;
- примитивы могут быть выведены линиями требуемого вида, закрашены любым цветом или использоваться для задания области отсечения для других графических элементов;
- текст полностью интегрирован с графикой, символы как стандартных шрифтов, так и определенных пользователем, рассматриваются как графические образы, которые обрабатываются графическими операторами языка;
- 2D общая координатная система поддерживает обычные линейные преобразования и их комбинации (сдвиг, поворот, масштабирование, отражение и т.д.);
- как построенные графическими операторами, так и естественные изображения, введенные, например, со сканера, могут иметь требуемые разрешение и динамический диапазон.

PostScript стал стандартом "де-факто" и получил чрезвычайно широкое распространение как язык описания страниц для лазерных принтеров и других устройств с высоким разрешением, его интерпретаторы входят в состав контроллеров растровых принтеров многих типов.

Языки описания страниц, близкие по возможностям к PostScript, разработаны также фирмами Xerox (язык Interpress) и Imagen (язык DDL).

На основе расширения языка PostScript фирмой Sun Microsystems разработана система NEWS [105] (the Network extensible Window System) для управления окнами в сети.

Язык PCL

Несколько версий языка описания страниц Printer Communication Language (PCL) было разработано фирмой Hewlett-Packard для вывода на лазерный принтер рисунков и текстов с использованием различных шрифтов. В версии PCL5 имеется 64 оператора, разбитых на 10 функциональных групп. Все операторы начинаются с символа Esc (шестнадцатиричный код 01BH) и содержат один или несколько последующих символов. Символьное кодирование, используемое в PCL, менее приспособлено для чтения человеком, чем явное текстовое кодирование, используемое в языке PostScript, но значительно компактнее последнего.

Аппаратно-независимые графические протоколы

Аппаратно-независимый графический протокол или метафайл представляют собой процедурное описание изображения в функциях виртуального графического устройства. Он обеспечивает возможность запоминать графическую информацию единым образом, передавать ее между различными графическими системами (в том числе работающими на различных ЭВМ) и интерпретировать информацию для вывода на различные графические устройства. Для интерпретации метафайла требуется локальная ЭВМ, выполняющая эмуляцию не реализованных в аппаратуре функций и кодирование в команды конкретных устройств.

В настоящее время в мировой практике наиболее активно поддерживаются стандартизованные аппаратно-независимые протоколы NAPLPS, GKSM, CGM и WMF — стандарт де-факто фирмы Microsoft на метафайл.

NAPLPS — North American Presentation Level Protocol Syntax

Это американский стандарт на представление графических данных в сетях VIDEOTEX (имеется также европейский аналог этого стандарта — SET). Основными требованиями при разработке этого протокола были следующие:

- возможность передачи графической информации в потоке буквенно-цифровых данных,

- минимальность объема передаваемых графических данных,
- минимальность усилий для интерпретации и возможность вывода изображений на простейшие графические устройства.

Обеспечение этих требований привело к тому, что был разработан эффективный способ упаковки графической информации в 7-ми или 8-битные коды ASCII. Эти же требования привели к ограничению функциональных возможностей протокола, что не позволяет получить высокое качество изображений при использовании современных графических устройств.

GKSM — Graphical Kernel System Metafile

Это de-facto стандарт на графический метафайл в рамках базисной графической системы GKS (приложение E к стандарту GKS). По функциональным возможностям этот протокол полностью соответствует системе GKS со всеми ее достоинствами и недостатками. Поэтому, он легко интерпретируется в системах, соответствующих стандарту GKS.

Недостатком GKSM-протокола (равно как и системы GKS) является, например, минимальный набор стандартизованных графических примитивов (их всего 5), что не дает возможности эффективного использования современных интеллектуальных графических устройств и увеличивает объем передаваемой информации. Возможность же использования обобщенного графического примитива или ESCAPE-механизма (предназначенных для нестандартных функций) не обеспечивает однозначность интерпретации графических данных при передаче их между различными системами.

Кодирование в GKSM текстовое, что позволяет читать (просматривать, редактировать) метафайл но требует большего объема чем символьное кодирование применяемое в NAPLPS или CGM.

CGM — Computer Graphics Metafile

Это стандарт ISO [88] на графический метафайл. Функционально этот протокол соответствует стандарту на интерфейс виртуального устройства CGI (Computer Graphics Interface), предназначенного для обеспечения связи различных графических систем с различным графическим оборудованием и являющегося обобщением текущего уровня развития графического программного и аппаратного обеспечения. По этой причине протокол CGM может совмещаться с различными программными системами и позволяет наиболее эффективно использовать возможности имеющихся графических устройств.

В CGM предусмотрены три способа кодирования — символьное, двоичное и текстовое. Символьное кодирование (по идеологии близкое к принятому в NAPLPS) достаточно компактно и предназначено для хранения и транспортировки графической информации. Двоичное кодирование требует минимальных усилий по генерации и интерпретации и предназначено для внутрисистемного использования. Текстовое кодирование наиболее наглядно и обеспечивает возможность визуального просмотра и редактирования графических файлов. При необходимости в рамках графической системы могут быть предусмотрены соответствующие перекодировщики.

DXF

Это формат графических файлов с которыми работает система AutoCad. Его следовало бы отнести к категории аппаратно-зависимых протоколов, но (в настоящее время) нет ни одного устройства понимающего этот формат. Использование этого формата целесообразно в том случае, когда есть необходимость использования графических возможностей системы Autocad.

WMF — Windows Metafile Format

В системе Windows фирмы Microsoft для сохранения и последующего использования цветных изображений используется свой формат метафайла. В WMF используется единственный способ кодирования — двоичный, который, как это отмечалось выше, наиболее компактен и обеспечивает наибольшие скорости упаковки и воспроизведения, но неудобен для просмотра и анализа человеком. Метафайл содержит заголовок и собственно описание изображения в виде записей GDI (Graphical Device Interface) функций. Всего предусмотрено три варианта метафайла — стандартный, размещаемый (placeable) и буферный (clipboard). Отличия вариантов состоят только в разных структурах заголовков. В составе Windows предусмотрены функции для создания и проигрывания метафайлов и манипулирования ими.

* * *

Перечисленные аппаратно-независимые графические протоколы ограничиваются 2-мерными графическими примитивами, что не позволяет использовать ресурсы локальной ЭВМ для сокращения объема передаваемой информации при работе с 3D изображениями. Поэтому, представляется целесообразной разработка двухуровневого графического протокола согласованного с CGM и обеспечивающего возможность передачи 3D объектов. Сокращение нагрузки на линию связи будет происходить в данном случае не только за счет повышения семантической насыщенности передаваемой информации, но и за счет возможности получения на локальной ЭВМ множества изображений с различными параметрами визуализации (например, множество изображений пространственного объекта с различных точек зрения).

Проблемно-ориентированные протоколы

Прикладные графические протоколы это объектно — ориентированные протоколы передачи данных между прикладными системами. Они наиболее компактны (вследствие высокой семантической насыщенности), допускают свободу в выборе различных способов графического представления, но требуют большей мощности локальной ЭВМ для интерпретации. Прикладные протоколы стандартизованы пока только для САПР машиностроения и электроники. В качестве примеров можно привести следующие стандарты:

- IGES (Initial Graphics Exchange Specification),
- STEP (STandard for the Exchange Product Model Data),
- MAP (Manufacturing Automation Protocol),
- VDAFS (Sculptured Surface Interface),
- EDIF (Electronic Design Interchange Format).

Основные трудности, связанные с разработкой протоколов этого уровня, состоят в том, что во многих областях применения до сих пор не унифицированы основные объекты (в том числе графические) и операции над ними. Для работы в этом направлении потребуются объединенные усилия проблемных специалистов, математиков и системных программистов в области баз данных, машинной графики, телекоммуникаций и т.д.

Растровые графические файлы

С появлением и широким распространением персональных ЭВМ, использующих растровые дисплеи и устройства документирования (лазерные и струйные принтеры и т.д.), для целей компактного хранения и транспортировки графической информации стали активно применяться различного рода растровые графические файлы. Используется более десятка различных типов растровых графических файлов. К наиболее известным относятся:
• TIFF (Tag Image File Format),
• GIF (Graphics Interchange Format),
• PIC,
• PCX,
• MAC (MacPaint),
• BMP (Bitmap).

TIFF (Tag Image File Format)

Разработан корпорациями Aldus и Microsoft совместно с некоторыми фирмами, производящими сканеры и принтеры. Этот формат поддерживается целым рядом систем подготовки документации и является наиболее реальным претендентом на стандарт для хранения и транспортировки растровых изображений.

Основной концепцией формата TIFF является цветовая модель изображения. Под этим понимается набор характеристик изображения, определяющих способ представления цвета. Стандартизованы следующие цветовые модели:
• двух-уровневое изображение (bi-level image);
• монохромное изображение (gray-scale image);
• индексированное цветное изображение (paletted color image);
• полное цветное изображение (full RGB image).

TIFF является открытым форматом и позволяет создать любую модель изображения. Естественно, что выбор требуемой модели определяется решаемой задачей. Например, двух-уровневая модель наиболее удобна в системах подготовки документации. Индексированное цветное изображение совместимо с форматом хранения графической информации в наиболее распространенных в настоящее время растровых графических дисплеях.

Помимо информации о модели изображения формат TIFF содержит метрические характеристики, а именно: размеры изображения, плотность (количество пикселов на единицу длины), с которой создавалось изображение. Эти характеристики особенно полезны в системах подготовки документации. TIFF не накладывает практически никаких ограничений на параметры изображения. Так, например размеры изображения могут достигать 4 миллиардов. Количество битов на пиксел ограничено этим же числом.

Формат TIFF позволяет хранить в одном файле любое количество изображений. Кроме того, есть возможность хранить несколько копий одного изображения с различными характеристиками. Так, например, можно иметь несколько вариантов изображения, отличающихся различной плотностью, что полезно опять же в издательских системах для работы с несколькими принтерами.

В формате TIFF имеется возможность упаковывать изображение различными методами. В том числе изображение может храниться и в неупакованном виде, что представляется удобным,

так как, например, при создании изображения важен произвольный доступ к любому элементу изображения за достаточно малое время. Одним из методов кодирования является LZW (Lempel, Ziv & Welch), который дает высокий коэффициент сжатия.

GIF (Graphics Interchange Format)

Разработан в CompuServe Incorporation для хранения и транспортировки растровых изображений. GIF позволяет содержать в одном файле несколько изображений, не связанных между собой.

По сравнению с форматом TIFF он более прост. В GIF используется наиболее распространенная модель изображения — индексированное цветное изображение. Хотя это не мешает хранить в нем изображения двух-уровневой и монохромной моделей. Это реализуется путем соответствующей настройки таблицы цветности. Модель полноцветного изображения с хранением для каждого пиксела компонент R, G и B принципиально не вписывается в формат GIF, так как существует ограничение на количество битов в пикселе — 8.

Достоинством формата GIF является наличие стандартизованного протокола передачи GIF-изображения по линии связи. Формат GIF, в отличие от TIFF, организован по принципу последовательного доступа. Т.е. он не содержит оглавлений или каких-либо ссылок внутри себя. Это делает его удобным при передаче изображений в распределенной графической системе.

GIF использует единственный метод кодирования изображений — LZW. Это свойство следует отнести к недостаткам формата, так как использование одного метода кодирования ограничивает область его применения. Кроме того, следует отметить, что не существует графических редакторов, понимающих формат GIF.

ZSoft (PCX)

Формат распространен на IBM PC и используется в графических редакторах (Paintbrush, EgaPaint) и системах подготовки документации (Ventura Deck Top Publisher, First Publisher).

В PCX используется очень неэффективный метод кодирования, он дает низкий коэффициент сжатия. Однако время, используемое на кодирование/декодирование практически равно времени кодирования без всякой упаковки. Это дает преимущества при использовании этого формата в интерактивных системах с быстрой сменой изображений.

MacPaint (MAC)

Является основным форматом в системах подготовки документации и графических редакторах на персональных компьютерах фирмы Apple (Macintosh, Lisa).

BitMap (BMP)

Является одним из основных форматов представления растровых изображений в системе Windows. Файл имеет достаточно простую структуру и сохраняет единственное изображение с одним, четырьмя, восемью и двадцатью четырьмя битами на пиксел. Одно-, четырех- и восьмибитное представления соответствуют индексированному цветному изображению. Для таких изображений в заголовке BMP-файла хранится таблица цветности. Изображение может быть сжато с использованием кодированием длин серий для четырех- и восьмибитных изображений.

Наряду с перечисленными имеется еще ряд форматов для хранения растровых файлов, используемых в отдельных системах, но не получивших широкого распространения. Их применение может оказаться целесообразным в случае использования соответствующих систем.

0.3.12 X Window System

В отличие от большинства других предложений по стандартизации, таких как, например, GKS, система X спроектирована и реализована небольшой группой разработчиков из Массачусетского технологического института в рамках проекта Athena, спонсорами которого были фирмы DEC и IBM.

Система X — программное окружение для программистов и пользователей прикладного программного обеспечения инженерных рабочих станций и представляет собой целостную систему, в рамках которой синтезированы предложения по управлению окнами, функциональному и языковому интерфейсам, аппаратно-независимому протоколу и программному обеспечению рабочей станции.

Основой X является “базовая оконная система”. Полное X-окружение надстроено над ней в виде уровней (рис. 0.3.7).

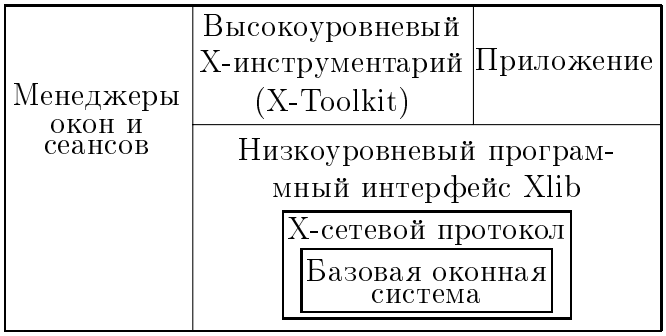

Рис. 0.3.7: Структура X

Базовая оконная система взаимодействует с окружающим миром только через сетевой интерфейс. Какого-либо специального привилегированного доступа к ней не предусмотрено.

Архитектура X-системы базируется на модели клиент-сервер (рис. 0.3.8). Клиент — прикладная программа. Сервер — программа, вызванная на компьютере, к которому физически подключен дисплей. В ней сосредоточена вся аппаратная зависимость.

Основные критерии, которыми руководствовались в разработке X-системы:

- транспортабельность — система должна быть применима к различным дисплеям;
- аппаратная независимость приложений;
- сетевая прозрачность — приложения могут быть вызваны как локально, так и на удаленных ЭВМ без каких-либо их изменений и без необходимости информирования об отличиях. Должны быть поддержаны различные сетевые протоколы;
- конкурирующие приложения — дисплей может делиться между несколькими приложениями. Возможен вывод из нескольких приложений как в единственное, так и в несколько окон;
- интерфейсы с приложениями и управление окнами — X не предписывает какого либо одного способа работы с окнами, например, только перекрывающимися;

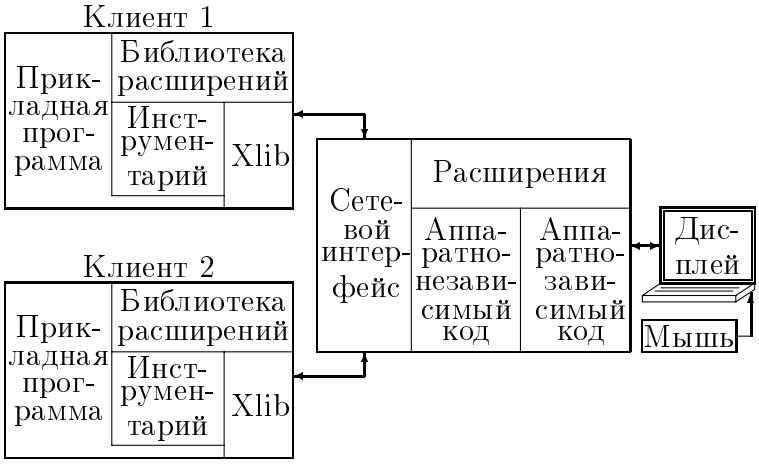

Рис. 0.3.8: Архитектура X-системы

- перекрытия окон — ввод/вывод должны быть возможны для полностью или частично затененных окон.

* * *

Одно из основных отличительных свойств X-системы — работа одного сервера с несколькими клиентами и наоборот одного клиента с несколькими серверами представляется весьма удобным и естественным. Подобный подход использовался, например, в системах ДИГРАФ.К и ДИГРАФ.Р [13].

Другое же основополагающее положение, что за перечерчивание окон, получивших экспозицию на экране, несет ответственность прикладная программа, приводит к необходимости иметь высокоскоростные каналы связи (например, Ethernet). Другими словами, за сравнительно малые результаты приходится платить несуразно высокую цену. Представляется целесообразным иметь управление окнами на терминальной ЭВМ, возможно и с какими-либо прагматическими ограничениями на возможности такой локальной системы управления окнами.

0.3.13 Выводы

В машинной графике широко распространилось понимание необходимости стандартизации, которая позволяет обеспечить переносимость пакетов прикладных программ, унифицировать графические методы работы, углубить их понимание и практического использования, ставить задачи перед разработчиками аппаратуры.

В настоящее время работы по стандартизации, в основном, сосредоточены на узком фронте специфицирования некоторого минимального набора “базисных” функций с одновременным стремлением к многофункциональности пакетов графических подпрограмм. Следует ожидать, что дальнейшее продвижение стандартизации будет идти по пути повышения ее функционального уровня в определенных, сформировавшихся областях приложений.

В целом, текущее состояние работ по стандартизации машинной графики — необходимый, но пока первый шаг в этой части создающейся на наших глазах индустрии программного обеспечения [21].

Наряду с положительными аспектами стандартизации следует отметить и определенные минусы, имеющие общий характер.

1. Стандартизация всегда означает затверждение некоторого определенного уровня достижений и понимания, тем самым в определенной степени тормозится развитие новых, нестандартных технических средств и программного обеспечения. Особенно, если учесть то, что первой строчкой наших стандартов является: “несоблюдение стандарта преследуется по закону”.

2. Стремление покрыть широкий спектр применений, начиная от пассивного вывода до высокоинтерактивных приложений, несмотря на наличие уровней, все-таки приводит к громоздкости и набора средств, и структуры данных и, естественно, программного кода.

3. Стремление к легкой адаптируемости влечет за собой чрезвычайно большое количество средств запроса к обстановке (в GKS — 75 функций из общего числа 185, т.е. более 40%). Такое количество несомненно избыточно для многих конкретных приложений. Не случайно поэтому, например, еще в 1987 г. темой одной из дискуссий на Всесоюзной школе-семинаре по “Информатике и интерактивной компьютерной графике” (Цахкадзор, 16–20 марта 1987 г.) было: “Стандартизация — закон или методология, тормоз или ускорение”.

Важно отметить, что в предложениях по стандартизации наряду со стремлением к многофункциональности пакетов очевидно и стремление к минимизации набора стандартизованных примитивных функций, что, вообще говоря, неверно для конкретной области приложений. В последнем случае повышение функционального уровня стандартизации обеспечит как легкость изучения, так и легкость адаптации, которая важна ведь не вообще, а в каждом случае в некотором конкретном классе приложений. Практика написания прикладных систем показывает, что для повышения эффективности прикладных программ требуется набор различных функциональных возможностей из различных, предлагаемых стандартами уровней.

В этой связи, интересным представляется решение, предложенное в [32], положенное в основу графпакета АТОМ. Система машинной графики представляется в виде совокупности пяти сравнительно слабо связанных подмножеств: средств формирования изображений; средств промежуточного хранения информации; средств ввода; средств преобразований изображений; средств управления графическими устройствами.

Требуемая конфигурация графической системы собирается из отдельных модулей, объединяемых в конвейер [34]. Конвейер собирается либо статически — на этапе проектирования программы, либо динамически, в процессе ее исполнения. Передача данных в конвейере осуществляется через единый межмодульный интерфейс.

0.4 СИСТЕМЫ УПРАВЛЕНИЯ ПОЛЬЗОВАТЕЛЬСКИМ ИНТЕРФЕЙСОМ (UIMS)

Начало интерактивных вычислений и, следовательно, исследование человеко-машинного интерфейса принято отсчитывать с 1959 г., когда на конференции Юнеско по обработке информации Г.Стречи предложил режим разделения времени при решении задач на компьютерах.

По мере роста мощности компьютеров росли и затраты на диалоговую компоненту программного обеспечения. Вопрос эффективности использования машин обострился во время стремительного выхода на рынок рабочих станций, объединивших интерактивность с графикой. Термин эффективность с тех пор изменил свое значение — если раньше он отражал такие характеристики как процессорное время и объем занимаемой памяти, то теперь под ним понимают простоту разработки, легкость сопровождения и удобство работы с программой. Поэтому затраты на исследование и разработку пользовательского интерфейса являются оправданными. В настоящее время большие усилия прикладываются к разработке методов и созданию инструментальных средств в рамках систем, получивших название UIMS — User Interface Management System.

Основные концепции UIMS были выработаны на ряде семинаров:

- 1983 Workshop on “User Interface Management Systems”, Seeheim, FRG;
- 1986 ACM SIGGRAPH Workshoop on “Software Tools for User Interface Management Systems”, Seattle, USA;
- 1987 Glasgow University Workshop on “User Interface Management Systems”;
- 1990 ESPRIT/Eurographics International Workshop on “User Interface Management Systems and Environments”, Lisbon.

Традиционный графический подход к интерфейсу с пользователем связан с работами Сазерленда, Ньюмена и др. [120, 104], в котором взаимодействие базируется на использовании графического дисплея с регенерацией и светового пера. Дальнейшее развитие графического диалога связано с прогрессом в области систем интерактивной машинной графики, который привел к регламентации в виде международных стандартов.

GKS — первый международный графический стандарт. В нем впервые зафиксированы концепции “рабочих станций” и логических устройств ввода (ЛОКАТОР, ШТРИХ, ДАТЧИК, ВЫБОР, УКАЗКА и КЛАВИАТУРА). К сожалению GKS был задуман во время превосходства парадигмы векторного рисования. Отсюда слабость поддержки диалога: отсутствие возможности ввода новых устройств или видоизменения изображения устройства на экране даже из прикладной программы (пользователя графического пакета), что приводит к необходимости использования в основном символьного ввода при организации диалога. Реализация диалога в GKS прерогатива прикладной программы, возможности раздельного проектирования не предполагается.

Второе направление графики — растровая графика оказала чрезвычайно большое влияние на все последующее развитие интерактивных систем. Все основные черты интерфейса с пользователем на современных рабочих станциях суть производные от работ по Xerox Park:

- управление окнами;
- использование графических символов (“икон”) для представления объектов;
- стиль взаимодействия, называемый непосредственным манипулированием;

- популярность “мыши” как устройства позиционирования на экране;
- объектно-ориентированный стиль программирования.

С тех пор система классификации инструментария для создания и управления пользовательским интерфейсом рассматривается на трех уровнях:
- Системы управления окнами (WMS-Window Manager System);
- Специализированный инструментарий;
  - обычный (MacIntosh, SunView ...),
  - объектно-ориентированный (Smalltalk-80, Andrew, InterView).
- Системы управления пользовательским интерфейсом.

В следующих разделах будут даны краткие характеристики, статус и функциональное описание каждого из этих уровней.

0.4.1 Системы управления окнами (WMS)

Многооконная технология обеспечивает пользователя доступом к большему объему информации, чем это возможно при работе с одним экраном. Окна дают доступ ко множеству источников информации. Пользователь может объединять информацию от нескольких источников, исследовать информацию на разных уровнях детализации. В мультипрограммном режиме есть возможность управлять несколькими параллельными задачами. Вход и выход каждой задачи отображается в разных окнах, позволяя пользователю сосредоточиться по необходимости на каждой задаче.

WMS операционная среда связанных с окнами ресурсов управления осуществляет поддержку:
- перекрывающихся окон (прямоугольных областей экрана);
- различных устройств ввода (цифровых и аналоговых);
- курсоров;
- шрифтов.

Интерфейс со стороны оператора и прикладной программы содержит команды заведения/уничтожения окон, изменения их размеров и положения, поднятие наверх, сжатия окна до пиктограммы и восстановления. Содержит графическую библиотеку вывода (только основные примитивы) и обработчик событий. Тем самым есть некие механизмы для реализации пользовательского интерфейса.

Возможны реализации WMS двух типов: базовая система (Kernel System), работающая на одной машине, и сетевая (Network oriented), реализуемая на основе модели клиент-сервер (client-server model).

0.4.2 Инструментарий создания пользовательского интерфейса

По Майеру [103], “Инструментарий создания пользовательского интерфейса есть библиотека технологических интерактивных средств, дающих возможность использовать физические устройства ввода (мышь, клавиатура, планшет...) для ввода значений (таких как команда, число, положение или имя) при наличии обратной связи, отображаемой на экране”. Программист использует этот инструментарий для организации взаимодействия с человеком. Инструментарий содержит набор функций, реализующий компоненты интерфейса нижнего уровня такие как: меню, кнопки, зоны диалога, подокна, зоны прокрутки.

Возможные модели управления, по терминологии конференции 1982 г. в Сиэтле:

1. Внутренняя (прикладная программа вызывает подпрограмму при необходимости ввода/вывода). Все управление диалогом сосредотачивается в прикладной программе, которая должна создаваться с учетом этого факта.

2. Внешняя (интерфейсные процедуры обращаются к прикладной программе в случае наступления требуемого события).

3. Смешанная, включающая модули с управлением по той и другой модели.

Пути реализации:

- механизм обратного вызова (прикладная программа организована как набор процедур вызываемых инструментарием);
- разделяемая память;
- передача сообщений;
- механизм обработки событий.

Дальнейшее развитие инструментария привело к появлению понятия Widget (заготовка) — объекта более сложного, чем перечисленный выше набор простых средств ввода в прикладную программу, хотя и включающих в себя эти средства. Такой инструментарий не стандартизован, различные фирмы (Apple, Sun etc.) предлагают существенно разный набор средств, как по номенклатуре, так и по функциональным возможностям. В качестве примера рассмотрим набор простых, составных и дополнительных заготовок, предоставляемых программным продуктом OSF/Motif.

Основные:

Область рисования	графическое пространство
Разделитель	линии разделяющие области
Метка	статический текст
Шкала	слайдер для получения числа
Зона прокрутки	управление прокруткой
Три типа Кнопок	управляющие кнопки с различным статусом
Каскадные Кнопки	кнопки для каскадных меню
Необязательные поля	отображение перечислимых значений переменной
Текст	ввод и редактирование текста
Команды	клавиатура с описанием

Составные:

Доска объявлений	панель с произвольным размещением объектов
Экранная форма	форма размещения объектов с выравниванием
Список	список строк
Вертикальное подокно	столбец с изменяемой высотой
СтрокаСтолбец	объект с ограничениями по строкам и столбцам
Зона меню	область меню для выпадающего меню
Кадр	контейнер для поддержки 3D обрамления

Дополнительные:

Прокрутка текста	область прокрутки текста
Прокрутка списка	область прокрутки списка
Окно прокрутки	обобщенная область прокрутки
Радио поле	набор радио кнопок
Поле выбора	выбор из списка строк
Поле выбора файла	специализированная область селектирования файлов
Основное окно	прикладное окно верхнего уровня
Поле диалога	транзитное поле диалога
Диалог в экранной форме	транзитное поле диалога для экранных форм "выпадающее"/"выпрыгивающее" меню
Сообщение/ предупреждение	зона диалога для печати сообщений

Несмотря на явное облегчение создания интерфейса пользователя с помощью такого инструментария, сейчас ставится задача создания интегрированной среды разработки и управления диалогом. Основной целью таких систем является отделение процесса конструирования интерфейса от разработки прикладной программы.

0.4.3 Системы управления интерфейсом пользователя

В научной литературе пока нет согласованного взгляда на термин UIMS — точное его значение само является объектом исследования. Одна из версий принадлежит Майерсу: "Система проектирования интерфейса пользователя есть интегрированный набор средств, помогающих программисту в создании и управлении различными интерфейсами пользователя. Эти системы обычно называют системами управления пользовательским интерфейсом (UIMS — User Interface Management Systems), но предпочтительнее называть их системами проектирования (UIDS — User Interface Development Systems), поскольку UIMS ассоциируется только с частью системы, работающей во время исполнения программы (но не с частью, используемой во время разработки), или с системами, включающими явные компоненты управления диалогом. UIDS обеспечивает как разработку, так и реализацию интерфейса и, таким образом, покрывает более широкий класс программ".

Основной концепцией UIDS является идея строгого разделения интерфейса и прикладной программы. В идеале она должна поддерживать все стили диалога и упрощать построение сложных интерфейсов. UIDS должен обеспечивать язык определения интерфейса для представления требуемого диалога и генератор, которой автоматически создает необходимый код из исходного определения в этом языке. Эти функции во многом похожи на функции компилятора или интерпретатора для обычных языков программирования.

Можно выделить три объекта, для каждого из которых ставятся различные цели при разработки UIDS.

1. Интерфейс с пользователем:
   - согласованность;
   - поддержка пользователя разного уровня;
   - обеспечение обработки ошибок и восстановления.
2. Разработчик программного обеспечения:
   - предоставление абстрактного языка для конструирования интерфейса пользователя;

- предоставление согласованных интерфейсов для связанных прикладных задач;
- простота изменения интерфейса на стадии его проектирования (быстрое создание прототипа);
- упрощение разработки повторным использованием программных компонент;
- обеспечение простоты изучения и использования прикладных программ.

3. Конечный пользователь:

- согласованность интерфейса по прикладным программам;
- многоуровневая поддержка сопровождения или функций помощи;
- поддержка процесса обучения;
- поддержка расширяемости прикладных программ.

Эти цели определяют следующие ФУНКЦИОНАЛЬНЫЕ ХАРАКТЕРИСТИКИ UIDS/UIMS:

- работа с входными устройствами;
- проверка допустимости ввода;
- обработка ошибок пользователя;
- реализация обратной связи;
- поддержка обновления/изменения данных прикладной задачи,
- поддержка задач развития интерфейса;
- синтаксическая поддержка.

Наиболее часто используется модель, введенная на конференции в Seeheim, в соответствии с которой UIMS состоит из трех КОМПОНЕНТ:

- система представления, обеспечивающая низкоуровневый ввод и вывод;
- система управления диалогом, обрабатывающая лексические единицы, получаемые в системе представления, в соответствии с синтаксисом диалога;
- модель интерфейса прикладной программы, представляющая семантику диалога и управляющая функциональностью прикладной программы.

Структурная схема компонент UIMS/UIDS представлена на рис. 0.4.1.

Конкретные реализации моделей основываются на различных СПОСОБАХ СПЕЦИФИКАЦИИ интерфейса, среди которых можно выделить следующие типы:

- языковая;
- графическая;
- автогенерация по спецификации прикладной задачи;
- объектно-ориентированный подход.

Каждая из этих спецификаций имеет свои особенности. В языковой используется спецязык для спецификации синтаксиса интерфейса. Такими языками могут служить:

- сети меню;
- диаграммы состояний и переходов;
- контекстно-свободные грамматики;
- языки событий;
- декларативные языки;
- обычные языки программирования;
- объектно-ориентированные языки.

Этот подход приложим к широкому кругу прикладных задач. Его недостатком является то, что разработчик диалога должен обладать профессиональной программисткой подготовкой.

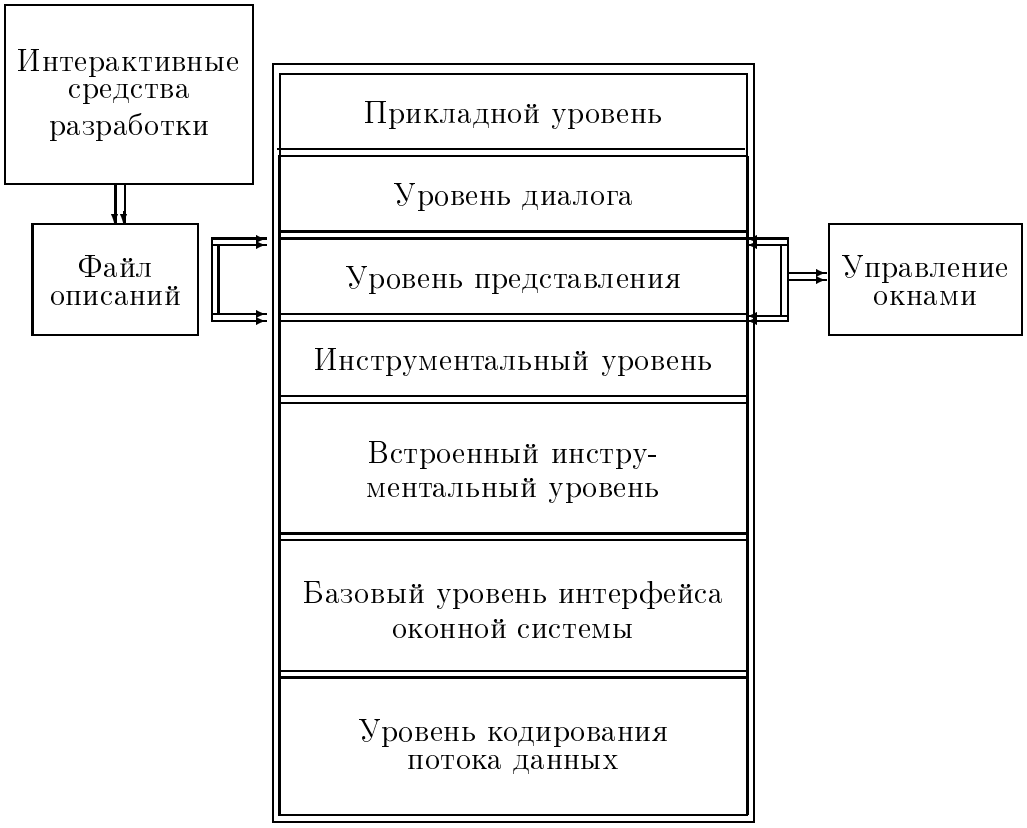

Рис. 0.4.1: Уровни в системах разработки пользовательского интерфейса

Графическая спецификация связана с определением интерфейса с помощью размещения объектов на экране (визуальное программирование). Она проста для использования не программистами, НО:

- трудна в реализации;
- поддерживает лишь ограниченные интерфейсы.

Здесь также существуют разные реализации:

- размещение на экране интерактивных средств (меню, кнопки и т.п.) и их привязка к фрагментам, написанным разработчиком интерфейса;
- сеть статичных страниц (кадров), содержащих тексты, графики, интерактивные средства;
- спецификация по демонстрации.

В третьем подходе создают интерфейс автоматически по спецификации семантики прикладных задач. Этим, в сущности, предпринимается попытка преодолеть сложности использования других технологий, однако ввиду сложности адекватного описания интерфейса, трудно ожидать скорого появления систем, реализующих такой подход в полной мере.

Возможные реализации:

- создание интерфейса на основе списка процедур прикладной программы (Mike);
- создание интерфейса по типам параметров процедур (Control Panel Interface);
- создание интерфейса на основе определения семантики прикладной задачи, описываемой на специальном языке (IDL).

Четвертый подход связан с принципом, называемом 'Direct Manipulation' — DM, рассматриваемым в следующем разделе. Основное свойство этого подхода состоит в том, что пользователь взаимодействует с индивидуальными объектами, а не со всей системой как единым целым.

0.4.4 Непосредственное манипулирование

Во многих отношениях технология непосредственного манипулирования (DM — Direct Manipulation) рассматривается как новая генерация методов программирования в области проектирования интерфейса с пользователем, имеющих такое же значение как разработка языка четвертого поколения для разработки баз данных. Начало этому подходу положили исследования, проводимые в центре Palo Alto корпорацией Xerox.

Что же такое непосредственность? Можно выделить четыре аспекта этого понятия:

Семантическая непосредственность. Определяется через “расстояние” между пользовательскими намерениями и операциями предоставляемыми системой. Пользователю важно, что: (1) в любое время он может передать свои намерения системе и (2) он может их выразить простым и кратким способом.

Для достижения этих целей необходимо, чтобы система предоставляла соответствующую функциональность, наряду с концептуальными объектами и операциями на уровне абстракции, удовлетворяющей пользователя. Эта проблема хорошо известна из области языков программирования высокого уровня — конструкции языка должны соответствовать проблемной области. Другими словами, семантическая непосредственность заключается в предоставлении пользователю возможности определять собственные классы графических объектов, соответствующих его прикладной задаче, а не заставлять его использовать базовые графические примитивы системы.

Операционная непосредственность. На уровне диалога можно рассматривать временной аспект непосредственности. Диалоговая последовательность не обладает нужной непосредственностью, если пользователь хочет воспользоваться последовательностью действий, не предоставляемых системой. Например, выбор пиктограммы с помощью мыши и получение возможности тут же передвинуть ее по экрану является реализацией непосредственности в операционном смысле, поскольку действие не подразделяется на дополнительные команды ввода (не надо нажимать клавишу “двигать”).

Но не просто определить последовательность действий в намерениях пользователя. Большинство DM систем пытаются сделать все видимые объекты доступными способами инициируемыми пользователем и поддержать развитие последовательности действий непосредственной обратной связью на каждом шаге.

Формальная непосредственность. Этот аспект относится к естественности восприятия системного вывода, простоте и эффективности ввода (клавиатура, кнопки, работа с мышью и т.п). Увеличению степени формальной непосредственности может способствовать использование представления в виде пиктограмм при выборе объектов и функций вместо символических имен команд, хорошо структурированный экран и понятное обозначение функциональных клавиш. Еще одним важным требованием является использование принципа “То что вы видите — то и получите” (WYSIWYG — What You See Is What You Get), благодаря которому на экране формируется именно то изображение, которое будет получено при распечатке.

Компоненты DM интерфейса. На верхнем уровне DM систем обычно находится одна из метафор графического представления (типа метафоры письменного стола, конкретный объект).

Через этот верхний уровень пользователю доступны прикладные программы, работающие на окнах. В окнах, подокнах и компонентах экрана доступны различные средства выбора объектов, функций инициирования и управления. Типичными компонентами используемыми для манипуляций с объектами и управляющими функциями являются:

- Обработчики (Handlers): управление непосредственно связанное с объектами, определяемыми прикладной программой. Обычно проявляются после выбора объекта и могут быть "захвачены" с помощью мыши для выполнения самых разнообразных манипуляций типа перемещения, изменения размеров, вращения и т.п.
- Управления (Controls): элементарные средства инициации функций или ввода параметров. Например, Motif предоставляет следующие типы управления:
  - кнопки различного вида (простые, радио, контрольные);
  - зоны (boxes) вывода, ввода, форматного ввода;
  - датчики (шкалы, ...).
- Меню: рассматриваемые как совокупность элементарных управлений с типовой организацией (в Open Look меню есть просто набор кнопок). Наиболее часто используемые типы меню: "выпадающие" (pull-down), "выпрыгивающие" (pop-up) и каскадные.
- Зоны диалога (Dialog boxes): для выдачи сообщений или ввода подтверждения.

Синтаксические режимы и обратная связь

DM часто называют интерактивной технологией с отсутствием режимов. Это конечно не так, но верно, что пользователь получает доступ к множеству функциональностей работая в нормальном контексте, команды и режимы ввода, присущие командно-ориентированным редакторам в этой технологии по возможности не используются. Обычной синтаксической композицией взаимодействия в DM является "выбор объекта" за которым следует "активация функции". Переход в режим "объект выбран" отражается изменением яркости объекта, после активации функции возможен переход в новый режим (например ожидания ввода параметра) отображаемый изменением формы курсора.

Требования к конструированию DM интерфейсов:

- поддержка разных классов интерактивной технологии;
- возможность создания новых типов технологий;
- использования подходящего, простого в изучении языка программирования для описания частей системы, которую трудно специфицировать графически;
- предоставление разной интерактивной технологии одновременно;
- обеспечение семантической обратной связи, проверки семантики и использования в семантике установок по умолчанию;
- гибкость компоненты представления;
- выражение синтаксиса в терминах индивидуальных объектов.

0.4.5 Пример реализации UIDS/UIMS

В заключении приведем в качестве примера описание системы UIMS XFaceMaker2 фирмы Non Standard Logics, созданной на базе OSF/Motif и X Window System.

1. Проектирование интерфейса. Пользователь создает интерфейс в интерактивном режиме, используя предопределенные элементы — заготовки (Widgets).

2. Спецификация ресурсов. Реализует простую установку параметров (ресурсов) для заготовок. Многообразие ресурсов заготовок и их взаимодействия делает задачу установки параметров чрезвычайно сложной. XFM2 везде, где это возможно, предоставляет предопределенную установку соответствующего выбора, в частности в зонах диалога.
3. Спецификация поведения интерфейса. Описывается на C-подобном командном языке (Face). Динамика поведения интерфейса трактуется XFM2 как целостная часть вместе с геометрическим представлением.
4. Простая и естественная связь между интерфейсом и прикладной задачей. Реализована двумя способами: вызовом функции прикладной задачи из описания или с помощью разделяемых переменных (активных значений). Разделяемые переменные могут быть любого типа. Таким образом возможна связь с прикладной задачей через указатель на заготовку в интерфейсе.
5. Непосредственное и полное тестирование интерфейса и его поведения (так называемый режим попытки (try mode). В этом режиме интерпретируется описание связанное с какими либо событиями, но без вызова функций прикладной задачи.
6. Эффективность конечного приложения. Результат проектирования реализуется двумя способами: либо интерпретацией описания, аналогично режиму TRY, либо компиляцией интерфейса вместе со всеми описаниями в C код.

0.4.6 Выводы

Ситуация в области разработки систем построения и управления интерфейсом пользователя в наши дни напоминает ситуацию с графикой до выработки международных стандартов: нет единой терминологии, есть разные подходы к построению моделей (лингвистический, графический), которые часто пересекаются.

Анализ разработок в области UIMS позволяет представить соотношение между различными компонентами в виде слойной структуры, напоминающей структуру OSI [104]. При реализации конкретной прикладной задачи необходимый уровень UIMS и набор компонент может определяться из соображений требуемой эффективности и доступных ресурсов.

Открытые проблемы в разработки UIMS можно определить как:
- эргономика взаимодействия;
- управление диалогом;
- отделение интерфейса пользователя;
- сопровождение, мобильность и эффективность.

Пока не существует единственной стратегии конструирования UIMS. Нужны эксперименты и опыт. Сфера быстро развивающаяся, сулящая многочисленные выгоды. Появившиеся стандарты (пока де-факто), по крайней мере на нижних уровнях систем (X-Windows и, в какой-то степени OSF/Motif), и активность разработчиков многих фирм дают надежду что ситуация в ближайшее время может измениться к лучшему.

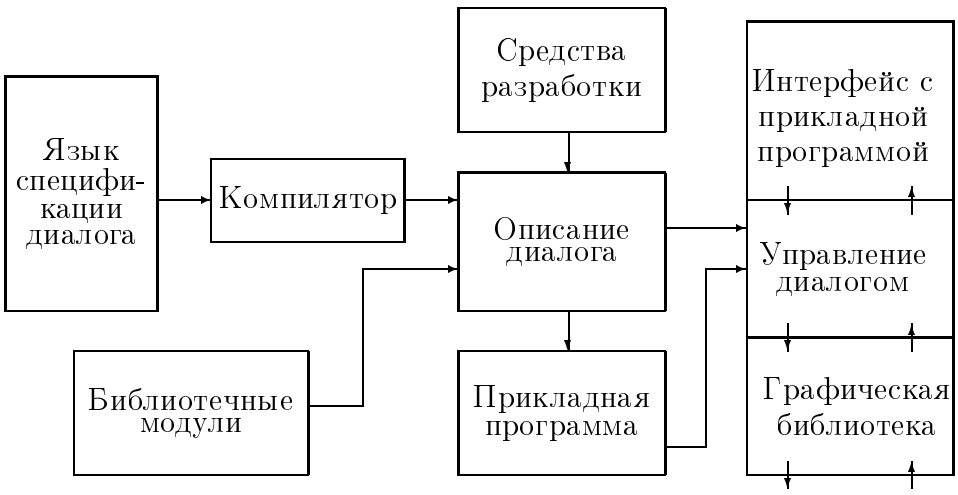

Рис. 0.4.2: Рекомендованная модель UIMS

0.5 VISC — ИНИЦИАТИВА

С расширением использования суперкомпьютеров заметно обострилась известная и ранее проблема — проблема представления результатов в форме удобной для воспринятия и анализа ее человеком. Здесь сложилась ситуация, когда, по образному выражению Апсона, "исследователь может вычислить больше, чем запомнить, а запомнить — больше чем понять". Поэтому чрезвычайно важна разработка средств, непосредственно ведущих от вычислений к пониманию.

Следует отличать визуализацию (visualization) от представления данных (prezentation) и отображения (rendering). Под визуализацией следует понимать средство, "делающее видимым невидимое", т.е. средство выражения мысли в графической форме [82]. Под представлением данных следует понимать ту или иную графическую интерпретацию необязательно графических данных. Под отображением следует понимать представление данных графического или геометрического характера на те или иные устройства вывода.

Повышение интереса к визуализации породило в США инициативу ViSC (Visualization in Scientific Computing) — "Визуализация в научных исследованиях" [55]. На ViSC — инициативу предполагается финансирование в объеме 1% от всех затрат на машинную графику (в 1990 г. это составило около 170 млн. долларов).

ViSC — инициатива охватывает (интегрирует) машинную графику, обработку изображений, компьютерное зрение, САПР (дизайн), обработку сигналов, пользовательский интерфейс.

Серьезной проблемой при построении системы интерактивной визуализации является совмещение графики с многооконным интерфейсом [58]. Важным моментом является использование современных стандартов машинной графики [80] и построение эталонной модели для системы визуализации [62].

Технической базой для разработки систем визуализации являются суперстанции, конкретный пример такой системы рассмотрен ниже.

0.5.1 AVS — Прикладная система научной визуализации

Одна из лучших систем по научной визуализации сделана на фирме STELLAR Computer при участии Крейга Апсона и Эндриса ван Дама [121].

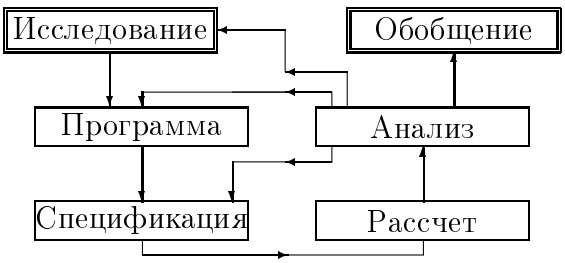

Рис. 0.5.1: Цикл вычислений

Эта система получила название AVS — Application Visualization System — и была задумана как инструмент для быстрой и эффективной визуализации данных при минимальных затрат на программирование для ученых и инженеров. Цель системы — объединить интерактивную графику и мощные вычислительные средства, которые легко можно было бы использовать непрограммистами, а с другой стороны, программистам эта бы система обеспечивала мощный и гибкий инструмент для развития. Система построена по модульному принципу и предоставляет пользователю интерфейс прямого манипулирования для каждого модуля. Система ориентирована для применения на супер графических станциях с мощным вычислителем и аппаратной поддержкой 3D-графики.

В проект системы были заложены следующие цели:

- простота в использовании;
- низкая стоимость;
- завершенность для применения;
- расширяемость функций;
- мобильность.

По мнению авторов системы такие стандарты, как PHIGS+ [106], GL, GKS [75], Core System [119] и др. имеют очень низкий уровень применения и не могут удовлетворить пользователя, занимающегося визуализацией. Аналогичное мнение было высказано В.Н.Кочиным [98]. С другой стороны, такие коммерческие системы, как MOVIE.BYU и продукция Wavefront и Alias, являются слишком замкнутыми, и их трудно интегрировать с прикладными задачами пользователей. Они больше используются как графические постпроцессоры после получения пользователем файла с данными. Типичный недостаток таких систем — хвост априорной ориентации на конкретную прикладную область (вычислительная химия, гидродинамика и др.) с вытекающими последствиями на привычные формы данных, используемые графические примитивы и алгоритмы. Такие системы трудно интегрировать с другими приложениями. Авторы AVS, используя новейшие достижения в области информационных технологий, такие как объектно-ориентированный подход, визуальное программирование и распределенные вычисления, смогли построить унифицированную систему, которую легко интегрировать с прикладной программой произвольной предметной области. Успех AVS во многом определился тем, что она разработана построена на общих принципах построения модели вычислений и анализа данных [77, ?] (рис. 0.5.1 и 0.5.2).

Основные процедуры переработки данных можно определить следующим образом:

- фильтрация данных — из одной формы в другую, возможно более информативную и менее объемную (фильтрация данных в данные),

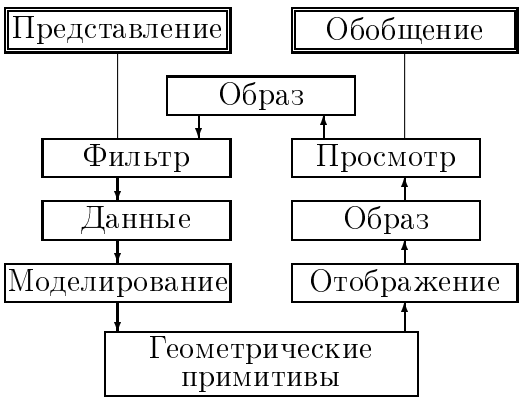

Рис. 0.5.2: Цикл анализа

- мэппинг (моделирование) — преобразование данных в геометрические примитивы (сопоставление чисел и геометрии),
- отображение геометрических данных в картинки (отображение геометрии в образы).

Процедура фильтрации включает в себя вычисление производных величин, таких как градиенты скалярных полей, огибающих поле скоростей и т.п. Эти новые, производные данные исследователь может связать с широким набором геометрических примитивов, традиционно используемых в машинной графике: точки, линии, сплайны, полигоны, поверхности, сферы и т.п. Один набор данных, таких как, например, трехмерное скалярное поле, может быть преобразован в геометрические примитивы многими способами, как показано на рис. 0.5.3.

При разработке AVS основным принципом было удовлетворение сформулированных выше требований.

Отдавая должное специфике научных исследований основное внимание уделялось расширяемости системы самими пользователями. При этом принималось во внимание наличие трех категорий разработчиков новых модулей.

В первую категорию входят пользователи, которые хотят создать новые модули для своих конкретных специфических приложений.

Вторую категорию составляют разработчики модулей, в некотором смысле универсальных для целой научной дисциплины, такой как газовая динамика или молекулярное моделирование.

Третью категорию составляют разработчики коммерческого прикладного программного обеспечения. Это программное обеспечение наиболее трудно интегрируется с другими платформами приложений.

Таким образом, AVS обеспечивает обобщенную функциональную применимость визуализации и специфицирует интерфейс между компонентами программного обеспечения для различных потребителей — пользователей и разработчиков, предоставляя им за небольшую стоимость больше гибкости и легкости к интеграции.

Архитектура системы прикладной визуализации

Основная идея архитектуры заключается в объединении усилий исследователей (пользователей) и разработчиков (программистов) для создания визуализации, независимо от конкретных приложений.

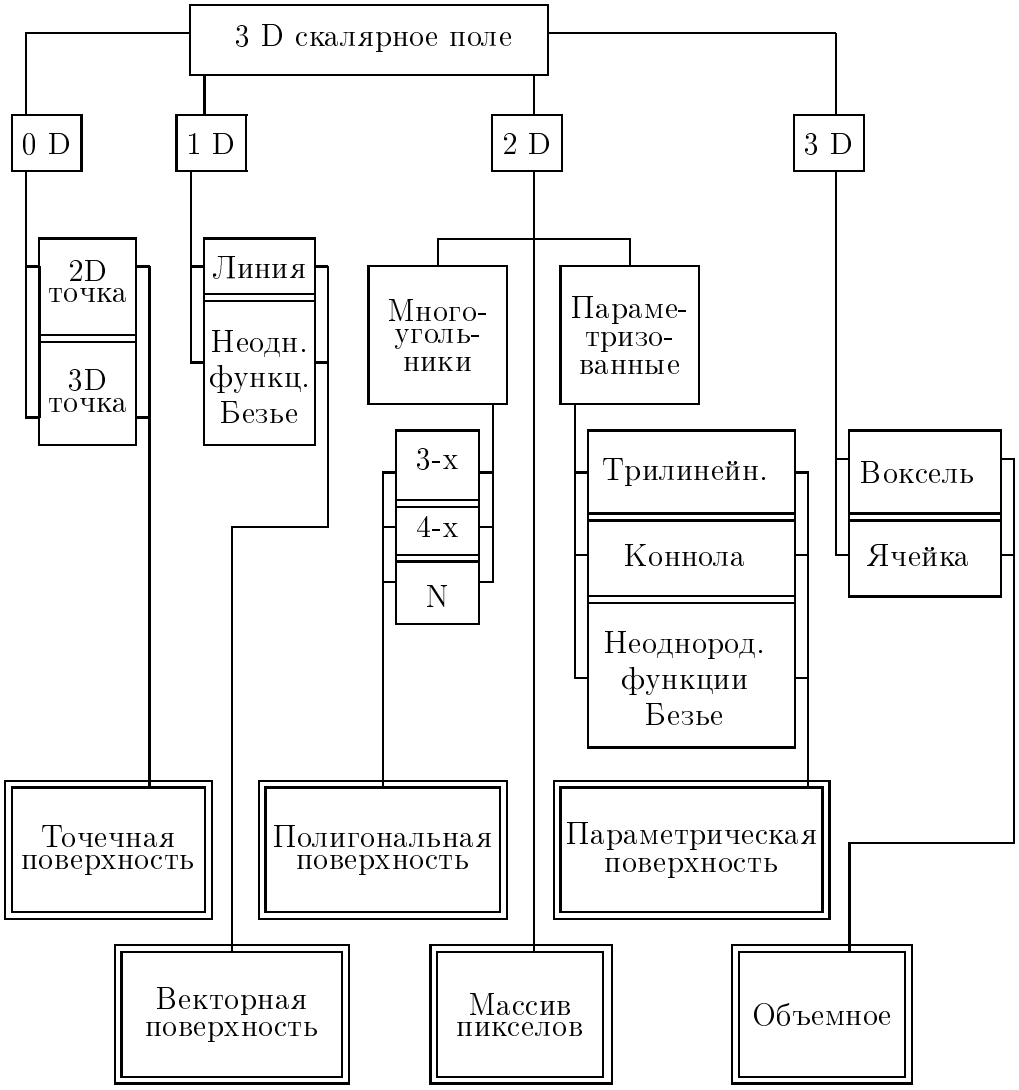

Рис. 0.5.3: Возможности моделирования трехмерного скалярного поля

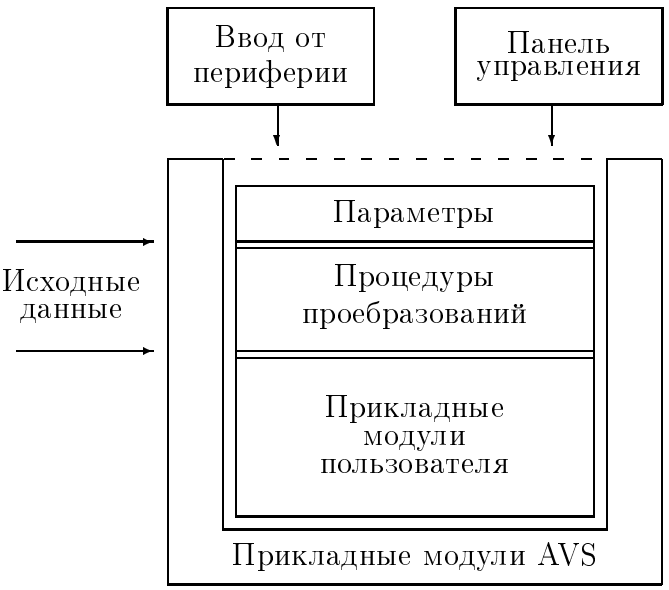

Рис. 0.5.4: Концептуальная модель модуля

Архитектура построения на элементах более высокого уровня абстракции, чем процедуры, но ниже, чем прикладные программы.

Основная концепция образована из объектно-ориентированного подхода [67] и заказного (customised) программного обеспечения. Обработка данных построена на основе конвейера с возможностью параллельной и распределенной реализации. Модули конвейера могут быть определены следующим образом:

- Исходные модули — генераторы данных (не имеющие исходных данных), вырабатывающие несколько потоков данных, в зависимости от параметров генерации.
- Модули преобразователи (или фильтры в терминологии ATOM) — процедуры, выполняющие фильтрацию данных, моделирование геометрических характеристик и отображение геометрии в изображение.
- Терминальные (оконечные) модули — осуществляющие вывод на конкретные устройства: дисплеи, принтеры, видеозапись.

Структура модулей (рис. 0.5.4) очень похожа на конвейерный подход, реализованный в графическом пакете ATOM [98]. При реализации программного обеспечения активно использовалась методика измерения программных алгоритмов и модулей и улучшение их характеристик на основе измеренных данных.

При проектировании пользовательского интерфейса основное внимание уделялось характеристике приспособляемости к различным группам пользователей, таких как “новички” и “эксперты”. За основу при реализации здесь брали методику визуального программирования [113].

В результате были созданы два интерфейса — пользовательский (для непрограммиста) и программистский (для профессионала-программиста). Эти интерфейсы существенно различаются как по использованию аппаратуры ввода (манипулирования), так и по программированию процедур заказчика.

В системе визуализации используются следующие методики [111]:

1. Отображение объемов, основанное на технике вокселей.

2. Отображение объемов построенное на технике реалистичности и технике текстур.
3. Мозаичное построение (закрашивание) изоповерхностей.
4. Техника направленных частиц.

Схема вычислительного конвейера показана на рис. 0.5.5. Причем, пользователь может управлять конкретной цепочкой вычислений из этого конвейера (как выбор пути на сетевом графике).

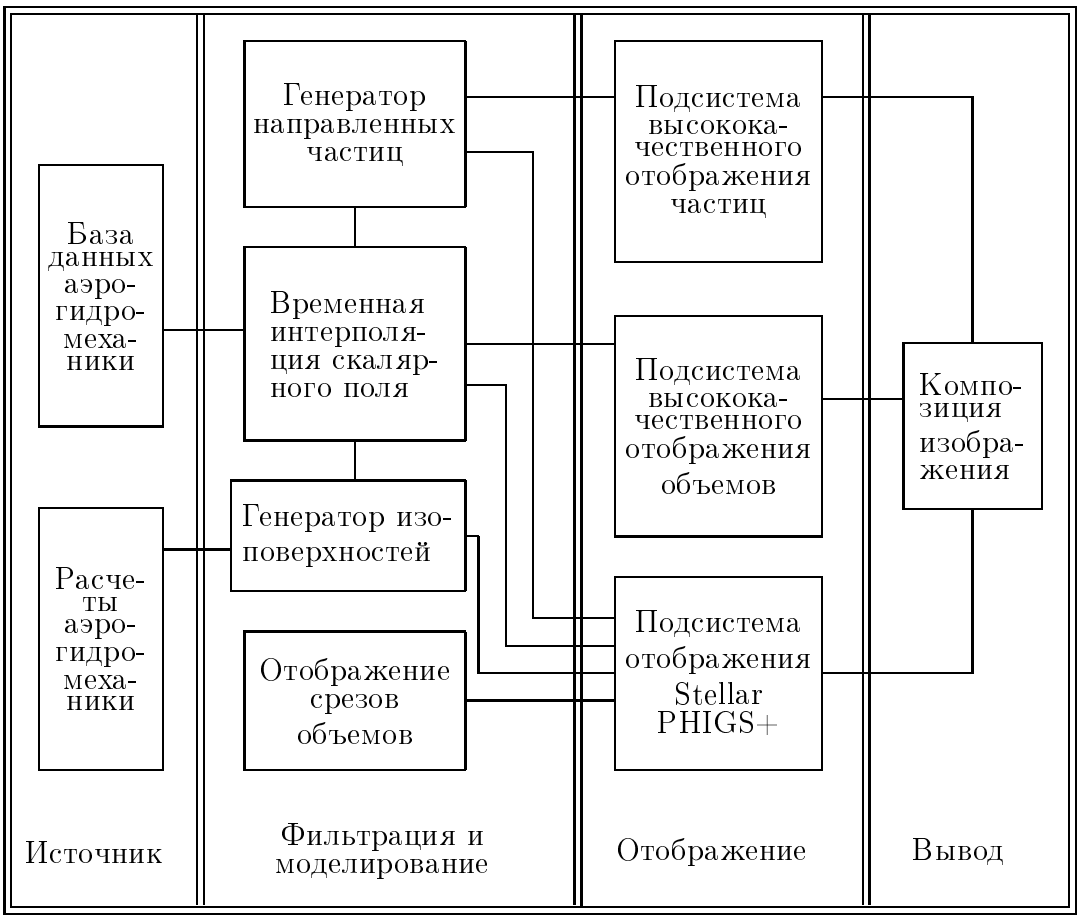

Рис. 0.5.5: Пример потока обработки

В заключение можно утверждать, что пользователи AVS получили в свое расположение систему нового типа для проведения исследований и разработок (R & D) в своих конкретных предметных областях с помощью методов визуализации. AVS дает возможность пользователям интегрировать свои прикладные программы с самыми современными алгоритмами и методами визуализации.

Другой целью создания AVS было построение такой законченной системы визуализации, которая походила бы к большинству прикладных вычислительных наук.

AVS по своему принципу построения не является застывшей системой, а способна к развитию и совершенствованию как самими пользователями, так и на основе обратной связи с ними.

0.6 ОЦЕНКА ПРОИЗВОДИТЕЛЬНОСТИ

В данном разделе:

- формулируются подходы к оценке производительности рабочих станций;
- кратко описываются популярные тестовые программы;
- приводятся результаты тестирования некоторых (более 20) рабочих станций.

Факторами, определяющие производительность рабочей станции, приведены на рис. 0.6.1.

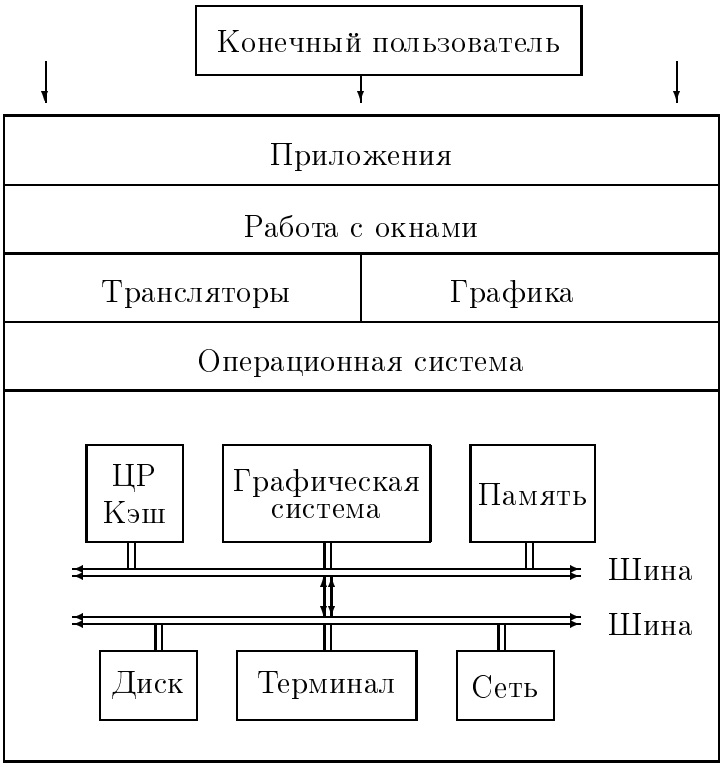

Рис. 0.6.1: Факторы, определяющие производительность рабочей станции

Для оценки производительности компьютеров могут быть предложены несколько подходов:

- функциональный: вычисления, работа с файлами . . .; используется на стадии проектирования "железа" (логическое моделирование);
- операционный: поведение системы в реальных ситуациях; используется при реализации "железа";
- языковой: качество компиляторов, оптимизация, библиотеки; используется при реализации компиляторов (программы на C, F77, ADA . . .);
- ОС: управление виртуальной памятью, обращения к системе, организация файловой системы; используется при реализации ОС.
- прикладной: автоматизация офисов, DBMS, CAD . . .; используется при реализации прикладного программного обеспечения.

Оценки производительности, основанные на сложных математических расчетах, например теории очередей, подходят разработчикам компьютеров и их производителям. Конечного же пользователя интересует в первую очередь производительность его собственного труда.

0.6.1 Тестовые программы (Benchmarks)

Решение, полностью устраивающее конечного пользователя, состоит в использовании его прикладной задачи в качестве теста на интересующем его ряде компьютеров. К сожалению, это очень часто оказывается невозможным.

Тестовая программа — это стандартная программа, используемая для сравнения производительности компьютеров. Тестовые программы создаются как для оценки общих характеристик: управление файлами, класс математических вычислений, так и для оценки более специфических возможностей данного компьютера. В результате появляется

возможность сравнить производительность различных компьютерных систем в интересующей Вас области. Кроме того, наличие нескольких тестовых программ (или одной, измеряющей несколько параметров) позволяет рассматривать компьютер "объемно", а не сводить эту сложную систему к одной характеристике, например, количеству mips.

В настоящее время на западе существует около 200 признанных тестов, которые могут быть разбиты на три группы:

- т.н. "стандартные" тесты: Dhrystone, Whetstone, Linpack, Doduc, Byte, Spice, Euug, Stanford, Musbus, Livermore, Los Alamos, и др., — опубликованные в журналах или распространяемые основными пользователями. Коды этих программ, часто измененные предыдущими пользователями, широко разошлись и теперь имеется значительное количество различных вариантов, что затрудняет их интерпретацию и использование;
- т.н. "коммерческие" тесты: AIM, Neal Nelson, Uniprobe, Workstation Laboratories, ... — хорошо документированные, предлагающие хороший сервис, но достаточно дорогие и предоставляющие ту же информацию, что и предыдущая группа тестов;
- т.н. "внутренние" тесты, используемые основными производителями (IBM, DEC, HP, ATT, Olivetty, NCR, Texas) для моделирования загрузки и калибровки своих компьютерных систем.

0.6.2 Результаты тестов

Для сравнения суперстанций в [63] в первую очередь используется группа тестов SSBA (The Synthetic Suite of Benchmarks from the French Association of Unix Users), предлагаемая Французской Ассоциацией Пользователей Unix'а и позволяющая оценить общие возможности компьютера. Результаты тестов, где это возможно, даны для вычислений с двойной точностью и с использованием возможностей оптимизации. В табл. приведены результаты следующих тестов:

- Mips/Joy: измеряет скорость ЦПУ. Количество MIPS, даваемое тестом, близко к коммерческому (или VAX 11/780). Использовано сокращение M/J;
- Dhrystone (Dhry/s): оценивает эффективность разработки ПО, особенно на C в среде Unix. Использовано сокращение Dhrys;
- Whetstone (KWhet/s): оценивает скорость работы для прикладных научных и инженерных задач. Использовано обозначение Whet;
- Linpack (Kflops): как и предыдущий, только для задач допускающих векторизацию вычислений (решает большую систему линейных уравнений). Использовано сокращение Lin;
- Saxer (Ko/s): измеряет скорость работы файловой системы;
- Test C (sec): измеряет скорость компиляции для языка C (пользователь + система). Использовано сокращение TC.

Графические возможности машин можно сравнить, опираясь на данные теста XBENCH — одного из наиболее популярных тестов для X Window серверов. Тест легко переносится, запускается и интерпретируется. Он был разработан фирмой Simens и оценивает эффективность в единицах — xstone, получаемых в результате взвешенного учета более элементарных измерений. Тест включает в себя измерения в 7 основных областях:

- Линии: solid, dashed, wide;
- Многоугольники: unfilled, filled, tiled, stippled, invertined;
- Окружности: unfilled, filled;
- Полигоны: filled;

Основные типы и параметры суперстанций. Таблица 6.1

Рабочая станция	Процессор	Частота	Диск	Шина
DN100X0 VS (APOLLO)	от 1 до 4 RISC PRISM	18 Mhz	ESDI striped	X-bus 150 Mb/c
Decstation 5000-200 PGX Turbo (DEC)	RISC MIPS R3000+R3010	25 Mhz	SCSI	turbochannel 100 Mb/c
ESV 50 (Evans&Sutherland)	RISC MIPS R3000+R3010	25 Mhz	SCSI	
HP 9000/433s Turbo VRX T3 (HP)	CISC Motorola 68040	33 Mhz	SCSI	EISA
RS 6000 Powerstation 730 (IBM)	RISC POWER	25 Mhz	SCSI	MCA 40 Mb/c
Interpro 6280 (INTERGRAPH)	RISC Clipper C300	50 Mhz	SCSI	
Powerstation 4D/3X0 VGX (Silicon Graphics)	от 1 до 8 RISC MIPS R3000+R3010	33 Mhz	IPI2X	
Stardent 30X0 (STARDENT)	от 1 до 8 RISC MIPS R3000 + vector unit	33 Mhz	SCSI striped	256 Mb/c
Sparc Station 470 TAAC-1 (SUN)	RISC SPARC Cypress	33 Mhz	IPI	Sbus
XD 88/34 (TEKTRONIX)	RISC Motorolla 88100 + 4 88200	20 Mhz	SCSI	Futurebus 100 Mb/c

- Битовые операции: screencopy, scroll, bitmapcopy;
- Текст: fixed fonts;
- Сложные операции (с окнами): create, draw, destroy.

Результаты теста приведены в табл. с использованием следующих сокращений:

П — количество процессоров,

Ком — комментарий,

Лин — линия,

Зап — заполненная область,

Бит — битовые операции,

Тек — текст,

Окр — окружность,

Слж — сложные операции,

XST — xstone.

Скорости основных построений на суперстанциях. Таблица 6.2

Рабочая станция	ОС	Окна	3D векторов/с	Полигонов/с
DN100X0 VS (APOLLO)	Domain/OS 10.2	MOTIF	1 100 000	155 000
Decstation 5000-200 PGX Turbo (DEC)	Ultrix 4.0	MOTIF	400 000	100 000
ESV 50 (Evans&Sutherland)	Unix ES	MOTIF + PEX	1 080 000	100 000
HP 9000/433s Turbo VRX T3 (HP)	HP-UX 7.05 или Domain /OS 10.3	MOTIF	1 000 000	300 000
RS 6000 Powerstation 730 (IBM)	AIX 3.1	MOTIF	990 000	120 000
Interpro 6280 (INTERGRAPH)	CLIX	MOTIF	400 000	30 000
Powerstation 4D/3X0 VGX (Silicon Graphics)	IRIX 3.2	4D Sight	1 000 000	1 000 000
Stardent 30X0 (STARDENT)	Titan OS	MOTIF	600 000	200 000
Sparc Station 470 TAAC-1 (SUN)	SunOS	Open Look	1 000 000	300 000
XD 88/34 (TEKTRONIX)	Utek V 3.2	MOTIF	1 000 000	65 000

Проверка общих возможностей компьютеров станций. Таблица 6.3

Машина	M/J	Dhrys	Whet.	Lin.	Saxer	TC
Solbourne Series 5/502	10,88	39370	11029	2624	714,3	40
BULL DPX/2 320	4,43	10141	1987	214	542,3	83
Tektronix XD88–30	9,84	32362	6424	1329	480,0	
MIPS rs 2030	10,51	26260	10823	1434	299,1	82
Sil.Gr. 4D/25	11,92	29481	11876	1467	513,6	64
Sil.Gr. IRIS 4D/80GT	10,33	24888	8562	1267	541,1	70
Sil.Gr. IRIS 4D/240S	14,09	39215	13774	3433	2087,3	53
Control Data 4360	15,90	46772	16393	3224	893,6	40
HP 9000/375	10,00	19230	8772	429	616,5	160
HP 9000/835	6,74	23518	7321	1627	391,9	44
SONY NEWS–1850	4,66	8726	1719	203	476,2	202
SONY NEWS–3860	14,42	34246	11111	1329	2658,0	43
SUN 3/260	3,54	7030	1461	115	384,6	179
SUN SPARC Station	6,67	20174	4573	1030	625,0	83
SUN 4/370	8,61	25295	7075	1622	192,3	69
DEC Station 5000/200	16,76	45248	16393	2624	3332,4	43
Data General AV310C	9,25	37119	8681	1228	448,0	79
Compaq 486/33L	13,78	21034	7599	1550	384,2	75
IBM RS/6000–320	14,76	52137	21739	7007	1760,3	6
IBM RS/6000–930	19,38	65530	27322	10729	2734,8	0,34
IBM RS/6000–540	22,14	78369	32468	11839	2415,0	0,31

Проверка станций тестом XBENCH Табл. 6.4

Машина	П	Ком	Лин	Зап	Бит	Тек	Окр	Слж	XST
SUN 4/260 (Xndx/unix)	1	unix-socket	54418	36648	56793	65312	229975	44640	53118
SUN 4/260 (Xndx/tcp)	1	10 MB ether	48769	33677	53372	66687	203667	30392	48220
HP9000/835 CHX	8	local	58085	23938	41182	63250	707026	29803	41684
D.G. AViiiON 300	8	10 MB ether	61027	28843	19757	56375	218463	22045	34648
HP9000/835 CHX	8	10 MB ether	49395	20705	34338	60156	470665	15555	33982
HP9000/340	8	local 10 MB	28422	18297	28680	55000	207900	8954	25003
GIPSI–tXC8 fpu	8	ether	18621	22717	37643	32187	115573	12614	24845
9733–203 (R2)	8	unix-soket	61216	14878	19142	24635	566337	18836	23218
GIPSI-tXC8	8	10 MB ether	17044	19243	37246	32500	12710	12418	21961
NCD–17C	8	10 MB ether	35676	13672	21341	33000	200829	15294	21292
SONY NEWS 1850	8	unix socket	16514	14477	25208	21656	598290	37124	20871

Проверка станций тестом XBENCH (продолжение) Таблица 6.4

Машина	П	Ком	Лин	Зап	Бит	Тек	Окр	Слж	XST
2HP9000/370 SRX	8	local	20478	11544	12655	56375	421195	25359	20074
HP9000/370 SRX	8	10 MB ether	19913	11439	12431	53968	351757	20130	19384
SUN 4/260 (Xndx/bs2)	1	10 MB ether	38776	23393	10302	58750	66543	4967	18989
DEC Station 3100(dec)	8	unix-socket	78115	6913	7292	79193	231587	16274	15735
DEC Station 3100 (mit)	8	unix-socket	76446	6723	7144	69375	224755	16143	15261
DEC Station 3100	8	10 MB ether	67956	6881	6698	72531	190992	12156	14634
SUN 3/50 (R3) purdue	1	unix-socket	14644	11489	16639	15312	43379	9745	14160
SUN 3/50 (R3) (4.0)	1	unix-socket	13843	11905	14552	13437	43130	10522	13447
SUN 3/50 (Torch)	1	unix-socket	10891	11361	17237	10312	20761	13071	11989
SUN 3/50 (R3) (3.4)	1	unix-socket	10228	10170	14337	10625	17941	10326	11106
SUN 3/160 (R2)	1	unix-socket	11536	10681	9021	11250	233545	10784	11035

Проверка станций тестом XBENCH (продолжение) Табл. 6.4

Машина	П	Ком	Лин	Зап	Бит	Тек	Окр	Слж	XST
Acorn R140 4.3 bsd	1	unix-socket	13266	8008	8865	13125	72954	10392	10759
SUN 3/50 (R3)	1	unix-socket	10000	10000	10000	10000	10000	10000	10000
NCD - 16 (R3)	1	10 MB ether	6366	5612	11771	15312	10455	7516	8469
DEC gpx (R2)	8	unix-socket	4835	7892	5710	30937	390212	5490	8250
HP 700/X	4	10 MB ehter	12351	7471	3302	17187	50051	5169	7582
Visual 640	1	10 MB ether	5532	3554	4893	8020	28662	3032	5124
SPARC Station 1	8	unix-socket	4877	1227	3006	9739	189817	12287	3424
GraphOn OptimaX 200	1	38,4 kBit	3493	2958	2141	3110	35294	15320	3115
SUN 386i-250 (R3)	8	unix-socket	2021	452	1334	3191	67722	5032	1314
Acorn R140, 4.3 bsd	4	unix-socket	2141	484	1344	2062	46674	4640	1279

СПИСОК ЛИТЕРАТУРЫ

Литература

[1] Бабат Е.Г. Графическое программирование для автономной дисплейной станции “Дельта”. Новосибирск, 1978. (Препринт/ ИАиЭ СО АН СССР; ь 80).

[2] Баяковский Ю.М., Галактионов В.А. Графические протоколы// Автометрия. 1978. ь 5. С. 3–11.

[3] Баяковский Ю.М., Галактионов В.А., Михайлова Т.Н. Графор. Графическое расширение Фортрана. М.: Наука, 1985.

[4] Бобков В.А., Белов С.Б. Сетевой графический протокол. Владивосток, 1980. (Препринт/ ИАПУ ДВНЦ СССР; ь 23).

[5] Бобков В.А., Белов С.Б. Универсальная графическая система// Проблемы вычислительной техники. М.: МЦНТИ, 1981. С. 145–155.

[6] Бобков В.А., Белов С.Б., Говор В.И. Интерактивная графическая система общего назначения// Материалы Второго Всесоюз. совещ. “Диалоговые вычислительные комплексы”. Серпухов, 1979. С. 279–284.

[7] Бучнев А.А., Вельтмандер П.В. Аппаратно-независимые графпакеты// Тез. докл. Всесоюз. конф. “Диалог человек-ЭВМ”. Ленинград, 1982. Ч. 2. С. 189–192.

[8] Бучнев А.А., Вельтмандер П.В. Сетевой графический протокол общего назначения// Тез. докл. Седьмой всесоюз. школы-семинара по вычислительным сетям. Москва—Ереван, 1983. Ч. 3. С. 20–25.

[9] Бучнев А.А., Вельтмандер П.В., Кудряков В.Ф. Базовое программное обеспечение обработки изображений// Тез. докл. Всесоюз. конф. “Диалог человек-ЭВМ”. Ленинград: ЛИАП, 1982. Ч. 2. С. 118.

[10] Вальский В.Б., Гартман О.В., Демиденко Т.М. и др. Диалоговый графический пакет для МВК “Эльбрус”// Тез. докл. Всесоюз. конф. “Диалог человек-ЭВМ”. Ленинград: ЛИАП, 1982. Ч. 2. С. 109–110.

[11] Венда В.Ф. Видеотерминалы в информационном взаимодействии (инженерно-психологические аспекты). М.: Энергия, 1980. 200 с.

[12] Венда В.Ф., Чачко С.А. Психологические факторы сложности диалога “человек-ЭВМ” в автоматизированных системах// Материалы семинара “Диалог в автоматизированных системах”. М.: Общество “Знание” РСФСР, Московский Дом научно-технической пропаганды. 1981. С. 111–118.

[13] Вельтмандер П.В. Аппаратно-ориентированные графпакеты// Проблемы машинной графики /Под ред. А.М.Мацокина. Новосибирск: ВЦ СО АН СССР, 1982. С. 32–38.

[14] Вельтмандер П.В. Интерактивная машинная графика в пакетах прикладных программ// Материалы Третьей всесоюз. конф. “Диалог человек-ЭВМ”. Протвино, 5–7 июля 1983. Серпухов: ИФВЭ. 1984. С. 23–38.

[15] Вельтмандер П.В. Распределенная система интерактивной машинной графики локальной вычислительной сети// Материалы пятой школы-семинара “Интерактивные системы”. Тбилиси: Мецниереба, 1983. С. 158–160.

[16] Вельтмандер П.В., Кривошеев В.А., Прошкин А.А., Сизых В.Г. Построение стереоизображений// Тез. докл. Всесоюз. конф. “Диалог человек-ЭВМ”. Ленинград: ЛИАП, 1982. Ч. 2. С. 187–189.

[17] Вуль В.А. Оперативные графические диалоговые системы и их применение// Зарубежная радиоэлектроника. 1980. ь 1. С. 57–85.

[18] Гантер Р. Методы управления проектированием программного обеспечения/ Пер. с англ. М.: Мир, 1981. 392 с.

[19] Гартман О.В., Комаровский В.Д., Макаров К.К. и др. Аппаратно-независимый пакет для диалоговой машинной графики// Материалы Второго Всесоюз. совещ. “Диалоговые вычислительные комплексы”. Серпухов, 1979. С. 285–287.

[20] Гилой В. Интерактивная машинная графика: структуры данных, алгоритмы, языки/ Пер. с англ. М.: Мир, 1981. 384 с.

[21] Громов Г.Р. Национальные информационные ресурсы: проблемы промышленной эксплуатации. М.: Наука, 1985. 240 с.

[22] Дебелов В.А. Диалоговый графический канал// Машинная графика и ее применение/ Под ред. А.М.Мацокина. Новосибирск: ВЦ СО АН СССР, 1979. С. 66–91.

[23] Зозулевич Д.М. Машинная графика в автоматизированном проектировании. М.: Машиностроение, 1976. 240 с.

[24] Каминский Л.Г., Клименко С.В., Лебедев А.А. О программном обеспечении графического дисплея с запоминанием.Серпухов, 1975. 12 с. (Препринт/ИФВЭ; ь 75–17).

[25] Каминский Л.К., Клименко С.В., Кочин В.Н., Лебедев А.А. О программном обеспечении машинной графики для физического эксперимента. Серпухов, 1978. 16 с. (Препринт/ИФВЭ; ь 78–6).

[26] Каминский Л.Г., Клименко С.В., Кочин В.Н., Самарин А.В. О программном обеспечении машинной графики для физического эксперимента// Материалы Второго Всесоюз. совещ. “Диалоговые вычислительные комплексы”. Серпухов, 1979. С. 228–293.

[27] Каминский Л.Г., Клименко С.В., Кочин В.Н., Самарин А.В. Графический пакет “АТОМ”. Структура и основные принципы. Серпухов, 1981. 25 с. (Препринт/ИФВЭ; ь 81–156).

[28] Катков В.Л., Янчук Т.С. Автокод для работы с графическим дисплеем ЕС-7064// Программное обеспечение машинной графики для решения научно-технических задач. Новосибирск: Институт математики СО АН СССР, 1977. С. 14–35.

[29] Клименко С.В., Кочин В.Н., Самарин А.В. Графический пакет “АТОМ”. Подпрограммы общего назначения. Серпухов, 1981. 35 с. (Препринт/ ИФВЭ; ь 81–172).

[30] Клименко С.В., Кочин В.Н. Графический пакет “АТОМ”. Построение графиков. Серпухов, 1982. 22 с. (Препринт/ИФВЭ; ь 82–9).

[31] Клименко С.В., Кочин В.Н., Самарин А.В. Графический пакет “АТОМ”. Построение пространственных объектов. Серпухов, 1982. 31 с. (Препринт/ИФВЭ; ь 82–71).

[32] Каминский Л.Г., Клименко С.В., Кочин В.Н., Самарин А.В. Графический пакет “АТОМ”. Структура и основные принципы. Серпухов, 1981. 25 с. (Препринт/ИФВЭ; ь 81–156).

[33] Клименко С.В., Кочин В.Н., Самарин А.В. Графический пакет для анализа данных. Средства формирования изображений// УС и М. 1983. ь 3. С. 104–109.

[34] Клименко С.В., Кочин В.Н., Самарин А.В. Конвейерная модель программного обеспечения машинной графики// Тез. докл. Всесоюз. конф. по проблемам машинной графики и цифровой обработки изображений. Владивосток, 24–26 сентября 1985. Владивосток: ИАиПУ ДВНЦ АН СССР, 1985. С. 19–20.

[35] Климов В.Е., Клишин В.В. Принципы построения системы машинного конструирования деталей “СИМАК-Д”// Материалы семинара “Автоматизация проектирования”. М.: Общество “Знание” РСФСР, Московский Дом научно-технической пропаганды. 1978. С. 130–137.

[36] Климов В.Е., Мороз С.В. Повышение производительности технических средств машинной графики // Электронная вычислительная техника. Сборник статей. М.: Радио и связь. 1989.

[37] Кольцов Ю.В., Манако В.В., Чичкань И.В. Графический пакет ГРАС. Базовые алгоритмы и структуры графических данных// Методические материалы и документация по пакетам прикладных программ. Вып.26: Машинная графика баз данных// Под ред. В.В.Пилюгина и Л.Н.Сумарокова. М.: МЦНТИ, 1984, с. 98–108.

[38] Курилов М.А., Манако В.В., Никитин А.И., Чичкань И.В. Стандартный графический пакет ГРАС. Средства отображения, хранения и передачи графической информации// Автоматизация научных исследований на основе применения ЭВМ: Тез. докл. VI Всесоюз. конф. Новосибирск, 1981. С. 116–117.

[39] Липкин И. Еще раз о RISC// Компьютер пресс. Обозрение зарубежной прессы. 1991. ь 6, с. 43–47.

[40] Лященко А.А., Кириевский Л.А., Демченко В.В. Методические рекомендации по применению пакета прикладных программ машинной графики для автоматизации выпуска проектно-графической документации на АРМ-С. Киев: НИИАСС, 1982. 46 с.

[41] Манако В.В., Никитин А.И. Об унификации программного обеспечения машинной графики// Машинная графика и обработка документации в управлении, планировании и проектировании. Тез. докл. Первой Всесоюз. школы-семинара. Цахкадзор, 1983. С. 36–41.

[42] Марин А.В., Щедрова М.Г., Сильдник И.Э. Технологические средства для разработки диалоговых программ// Тез. докл. Всесоюз. конф. "Диалог человек-ЭВМ". Ленинград: ЛИАП, 1982. Ч. 1. С. 145–147.

[43] Математическое обеспечение графопостроителей. I уровень. СМОГ. Инструкция по программированию/ Под ред. Ю.А.Кузнецова. Новосибирск: ВЦ СО АН СССР, 1976. 118 с.

[44] Математическое обеспечение графопостроителей. II уровень. СМОГ. Инструкция по программированию/ Под ред. Ю.А.Кузнецова. Новосибирск: ВЦ СО АН СССР, 1976. 78 с.

[45] Математическое обеспечение графопостроителей (I уровень)/ Под ред. А.Я.Куртукова. Новосибирск: ВЦ СО АН СССР, 1971.

[46] Ньюмен У., Спрулл Р. Основы интерактивной машинной графики/ Пер. с англ. М.: Мир, 1976. 573 с.

[47] Панкеев Г.А. Пакет для создания аппаратно-независимого программного обеспечения машинной графики// УС и М, 1983. ь 2. С. 115–117.

[48] Панфилов А.В. Графические аппаратные средства персональных компьютеров семейства IBM PC // Микропроцессорные средства и системы. 1990. ь 3.

[49] Пиковский А.С. Реализация распределенной графической системы на основе протокола виртуального графического терминала// Машинная графика и обработка документации в управлении, планировании и проектировании: Тез. докл. Первой Всесоюз. школы-семинара. Цахкадзор, 1983. С. 95–99.

[50] Пилюгин В.В. Структура математического обеспечения графического пакета САГРАФ// Материалы семинара "Автоматизация проектирования". М.: Общество "Знание" РСФСР, Московский Дом научно-технической пропаганды. 1978. С. 11–15.

[51] Пилюгин В.В., Аджиев В.Д., Поляков М.Д. Решение на ЭВМ задач геометрического и графического моделирования// Материалы семинара "Автоматизация проектирования". М.: Общество "Знание" РСФСР, Московский Дом научно-технической пропаганды. 1981. С. 74–77.

[52] Роуз В. Разработка интерфейсов человек-машина для интерактивных систем работающих в реальном времени// ТИИЭР. 1975. Т. 63, ь 6. С. 17–30.

[53] Фоли Д.У. Искусство организации естественного графического диалога человек-машина// ТИИЭР. 1974. ь 4. С. 54–67.

[54] Фоли Дж., вэн Дэм А. Основы интерактивной машинной графики. В 2 т./ Пер. с англ. М.: Мир, 1985.

[55] ACM SIGGRAPH Computer Graphics, v. 21, ь 6, November 1987.

[56] Allari S. et al. Achievements Derived from the Adoption of UIMS with Graphic Interaction Techniques in Vitamin Project// Proceeding of the Graphics and Interaction in ESPRIT Session. Eurographics'89.

[57] APT. Part programming manual. IIT Researh Institute, Chicago, 1964.

[58] Arnold D.B. CGI Versus X-11. State of the Art Reports// Eurographics' 89, Hamburg, FRG, 4–8 September 1989. 42 p.

[59] Asal M., Short G., Preston T. et all. The Texas Instruments 34010 Graphics System Processor// IEEE CC&A, October 1986.

[60] Belch F.R. An Economik High Performance Pixel Memory Witch 3D Solids Capability// Computer Graphics Forum 6 (1987).

[61] Bergernn R.D. Picture Primitive in device Independent Graphics System // Computers and Graphics. 1976. V. 10, ь 1. P. 57–60.

[62] Bergeron R.D., Grinstein G.G. A Reference Model for the Visualization of Multi-Dimensional Data// Eurographics' 89, Hamburg, FRG, 4–8 September 1989.

[63] Binot Ch. Architecture and Evaluation of Graphics Superworkstation// Eurographics' 90. Tutorial Note 9. Montreux, 3.–7. September 1990. Switzerland, Montreux.

[64] Carinalli C., Blair J. National's Advanced Graphics Chip Set for High-Performance Graphics// IEEE CC&A, October 1986.

[65] Cockton G. Interaction Ergonomics, Control and Separation: Open Problems in User Interface Systems., AMU8811/03H, Scotish HCI Centre, 1988.

[66] Computer Graphics Interface Techniques for Dialoques with Graphical Devices (CGI), ISO DP 9636, 1986.

[67] Cox B.J. Object-Oriented Programming, an Evolutionary Approach, Addison-Wesley, Reading, Mass., 1986.

[68] Duce David A., Hopgood F.R.A. Essential PHIGS and View of PHIGS PLUS// Eurographics' 91. Tutorial Note 4. Vienna, 2.–6. September 1991. Austria, Vienna. 104 p.

[69] mbib69 User Interface Toolkits and User Interface Management Systems // ZGDV e.V. Darmstadt, FRG.

[70] Enderle G., Kansy K., Pfaff G. Computer Graphics programming. GKS — the Graphics Standard. Berlin, Heidelberg, New York, Tokyo: Springer Verlag, 1984. 542 p.

[71] England N.A. A Graphics System Architecture for Interactive Application-Specific Display Functions// IEEE CC&A, January 1986.

[72] Fountanier G., Gross P. Architectures of Graphics Processors for Interactive 2D Graphics//Computer Graphics Forum 7 (1988)

[73] Gloebel M., Kroemker D. A Multi-Microprocessor GKS Workstation// IEEE CC&A, July 1986.

[74] Gourand H., Stapleton M. PHIGS PLUS// Eugographics UK Conference. Tutorial Note. Eugographics UK Chapter, PO Box 38, Abingdon OXON OX14 1PX, UK April, 1991.

[75] Graphical Kernel System. ANSI X3H3/83–25r3, special issue// Computer Graphics, Feb. 1984.

[76] Guttag K., J. van Aken, Asal M. Requirements for VLSI Processor//IEEE CC&A, January 1986.

[77] Haber R. Visualization in Engineering Mechanics// SIGGRAPH'88 Course Notes, R.Wolff, ed. ACM, N.Y., 1988.

[78] Hans Joseph, Max Mehl. Computer Graphics Hardware: Introduction and State of the Art// Eurographics' 91. Tutorial Note 9. Vienna, 2.-6. September 1991. Austria, Vienna. 29 p.

[79] Henry. M. Levy. VAXstations: A General Purpose Raster Graphics Architecture//ASM Transactions on Graphics, V. 3, ь 1, January 1984.

[80] Hewitt W.T. PHIGS BR. State of the Art Reports// Eurographics'89, Hamburg, FRG, 4–8 September 1989. 64 p.

[81] Howard T.L.J., Hewitt W.T., Hubbold R.J., Wyrwas K.M. A Practical Introduction to PHIGS and PHIGS PLUS, Addison-Wesley, 1991.

[82] Hubbold R.J. Interactive Visualization: Challanges and Prospects. State of the Art Reports// Eurographics' 89, Hamburg, FRG, 4–8 September 1989. 39 p.

[83] Hubbold R.J., Hewitt W.T. GKS-3D and PHIGS — Theory and Practice // Eurographics' 88. Tutorial Note 1. Nice, 12.–16. September 1988. France, Nice. 60 p.

[84] Hurwith A., Citron J.P., Jeaton J.B. GRAF: Graphic Addition to Fortran. Proc. AFIPS Spring Joint Computer Conference, 1967, pp. 553–557.

[85] International standard ISO 7942 (Draft). Information proceedings systems — Computer Graphics — Graphical Kernel System (GKS) Functional description. В надзаг.: 1984 September 28, ISO TC97/SC5/WG2.

[86] International standard ISO 7942 Information Proceedings Systems — Computer Graphics, Graphical Kernel System (GKS) Functional Description. International Standards Organization (ISO), 1985.

[87] International standard ISO 7942. Information proceedings systems — Computer Graphics — Graphical Kernel System (GKS) Functional description. В надзаг.: 1985 August 15, ISO TC97/SC5/ WG2.

[88] International standard ISO 8632 Information Proceedings Systems — Computer Graphics, Metafile for the storage and transfer of picture description information (CGM). International Standards Organization (ISO), 1987.

[89] International standard ISO 8651 Information Proceedings Systems — Computer Graphics, Graphical Kernel System (GKS) language bindings – Part 1: Fortran ISO 8651–1, Part 2: Pascal ISO 8651–2, Part 3: Ada ISO 8651–3, Part 4: C ISO 8651–4, International Standards Organization (ISO), 1988.

[90] International standard ISO 8805 Information Proceedings Systems — Computer Graphics, Graphical Kernel System for Three Dimensions (GKS-3D) Functional Description. International Standards Organization (ISO), 1988.

[91] International standard ISO 8806 Information Proceedings Systems — Computer Graphics, Graphical Kernel System for Three Dimensions (GKS-3D) language bindings – Part 1: Fortran, ISO 8806–1, Part 2: Pascal, ISO 8806–2, Part 3: Ada, ISO 8806–3, Part 4: C ISO 8806–4, International Standards Organization (ISO), 1991.

[92] International standard ISO 9592 Information Proceedings Systems — Computer Graphics, Programmer's Hierarchical Interactive Graphics System (PHIGS). Functional Description. International Standards Organization (ISO), 1988.

[93] International standard ISO 9593 Information Proceedings Systems — Computer Graphics, Programmer's Hierarchical Interactive Graphics System (PHIGS). Language bindings – Part 1: Fortran ISO 9593–1 (1990), Part 2: Pascal ISO 9593–2 (1990), Part 3: Ada ISO 9593–3 (1990), Part 4: C ISO 9594–4 (to be publisched) , International Standards Organization (ISO).

[94] ISO/IEC JTC 1, N 3, PHIGS+ Functional Description Rev 3.0. International Standards Organization (ISO), 1988.

[95] Jacked D. The Graphics PARCUM System: A 3D Memory System Based Computer Architecture for Processing and Display of Solid Modeles// Computer Graphics Forum 4 (1985).

[96] Kellner R.G., Reed T.H., Solem A.V. An Implementation of the ACM/SIGGRAPH Proposed Graphics Standard in a multisystem environment // Computer Graphics. 1978. V. 12, ь 3. pp. 308–312.

[97] Kilgour A. Theory and practice in user interface management systems// SYSTEMS vol 29, ь 4, 1987.

[98] Klimenko S.V., Kochin V.N., Samarin A.V. Pipeline Approach for Building the Presentation Graphics Systems// Eurographics'85, / Под ред. C.E. Vandoni. Elsevier Sci. Pub., 1985, pp.427–438.

[99] Kopke F. GALA — Eine Maschinenunabhangige Interaktive Graphische Programmiersprache// Angewandte Informatik. 1976. V. 18, ь 8. S. 343–347.

[100] Dr.Ing. Kurth J. COMPAC Ein System zur rechner-orientierten Werkstuckbeschreibung// Zeitschrift fur wirtschaftliche Fertigung. 1973. V. 68, ь 2. S. 61–67. ь 3. S. 127–132.

[101] Ludwig M., Weber R., Richter K. DIGOS — Ein Digital-geometrie-orientiertes System fur die Konstruction und Darstellung geometrischer Objekte// Zeitschrift Rechnentechnik/Dateverarbeitung. 1976. ь 1. S. 40–45.

[102] Maxine D. Brown. Understanding PHIGS. Template, San Diego, 1985.

[103] Myers B. Creating dynamics interaction techniques by demonstration// ACM CHI 87–GI Conference, 1987.

[104] Newman W.M. A system for interactive graphical programming// Proc. Spring Joint Comput. Conf. Spartan Books, Baltimore, USA, 1968.

[105] NEWS Manual, Version 1.1. Mountain View, California, 1987.

[106] PHiGS+ Function Description Revision 3.0// Computer Graphics (Proc. SIGGRAPH), Vol. 22, N 3, July 1988, pp. 125–218.

[107] PostScript language description/ Adobe System Inc. Addison-Wesley Publishing Company, Inc., Reading, Massachusets, 1985.

[108] PostScript language reference manual/ Adobe System Inc. Addison-Wesley Publishing Company, Inc., Reading, Massachusets, 1985.

[109] PostScript language tutorial and cookbook/ Adobe System Inc. Addison-Wesley Publishing Company, Inc., Reading, Massachusets, 1985.

[110] Prime M. User Interface Managment Systems — A Current product Rewiew// Computer Graphics Forum 9, 1990.

[111] Proc. Volume Visualization Workshoop, Oniv. of North Carolina, Chapel Hill, 1989.

[112] Shires G.A. New VLSI Graphics Coprocessor — The Intel 82786// IEEE CC&A, October 1986.

[113] Shu N.C. Visual Programming, IBM Los Angeles Scientific Center, Van Nostrand Reinhold, N.Y., 1988.

[114] Smith D.N. GPL/1 — A PL/1 Extension for Computer Graphics// Proc. AFIPS Spring Joint Computer Conference, 1971, pp. 511–538.

[115] Sprool R.F., Sutherland I.E. et all. The 8 by 8 Display// ASM Transactions on Graphics, V. 2, ь 1, January 1983.

[116] Sproull R.F., Thomas E.L. A Network graphics protocol: in Techн. Report, ARPA. Network, NIC 24308, August, 16, 1974.

[117] Spur G., Schliep W. Integration of Mechanical calculation programs in CAD Systems// Proc. of 5th International Conference and exhibition on Computers in Desing Engineering. Brighton Metropole Sussex, UK, 30 March — 1 April 1982, p. 615–631.

[118] Status Report of the Graphics Standards Planning Committee of ASM/SIGGRAPH// Computer Graphics. 1977. V. 11, ь 3. 130 p.

[119] Status Report of the Graphics Standards Planning Committee of ASM/SIGGRAPH// Computer Graphics, Vol. 13, ь 3, special issue, Aug. 1979.

[120] Sutherland I.E. Sketchpad: man-mashine graphical communication system// PhD Thesis Massachussets Institute of Technology.

[121] Upson C. et al. The Application Visualization System: A Computation Environment for Scientific Visualization// IEEE Computer Graphics & Applications, July 1989, pp. 30–42.

[122] Upson C. Scientific Visualization Euvironments for the Computational Sciences// Proc. Compcon 89, CS Press, Los Alamitos, California, pp. 322–327.

[123] Warner J.W. e.a. DIGRAF — a Fortran Implementation of the Proposed GSPC Standard// Computer Graphics. 1978. V. 12, ь 3. pp. 301–307.

[124] Woodsford P.A. The Desing and Implementation of the GINO-3D Graphics Software Package// Software practice and experience. 1971. V. 1. pp. 335–365.

[125] mbib125Maker: An Interface generator for OSF/Motif

[126] Ziegler J.E Direct Manipulation Techniques for Human Computer Interfaces// Eurographics–90, Technical Report Series.