



улахаамаа,
минхэнэ дэсрөөр
наа гэдгүч наама
он автор
16 нугаа 2008.

2008



Серия основана в 2000 году

РЕДАКЦИОННАЯ КОЛЛЕГИЯ:

чл.-кор. РАН *И.Б. Федоров* — главный редактор
д-р техн. наук *И.П. Норенков* — зам. главного редактора
д-р техн. наук *В.В. Девятков*
канд. техн. наук *И.П. Иванов*
д-р техн. наук *В.А. Матвеев*
канд. техн. наук *Н.В. Медведев*
д-р техн. наук *В.В. Сюзев*
д-р техн. наук *Б.Г. Трусов*
д-р техн. наук *В.М. Черненький*
д-р техн. наук *В.А. Шахнов*

А.Н. Божко, Д.М. Жук, В.Б. Маничев

Компьютерная графика

Допущено Учебно-методическим объединением вузов
по университетскому политехническому образованию
в качестве учебного пособия для студентов
высших учебных заведений,
обучающихся по направлению
«Информатика и вычислительная техника»

Москва
Издательство МГТУ имени Н.Э.Баумана
2007

УДК 004.92(075.8)

ББК 32.973

Б72

Рецензенты:

д-р техн. наук, проф. А.Н. Данчул;
кафедра «Цифровые информационные системы»
Московского университета печати
(зав. кафедрой д-р пед. наук, проф. М.Ф. Меняев)

Божко А.Н., Жук Д.М., Маничев В.Б.

Б72 Компьютерная графика: Учеб. пособие для вузов. — М.: Изд-во МГТУ им. Н.Э. Баумана, 2007. — 392 с.: ил. — (Информатика в техническом университете.)

ISBN 978-5-7038-3015-4

Рассмотрены физические основы цветовосприятия и особенности самых распространенных цветовых моделей. Дано представление о принципах измерения цвета и калибровки устройств ввода-вывода. Подробно описаны технические средства компьютерной графики и особенности графических устройств современных вычислительных систем. Изложены алгоритмы растровой графики и форматы графических файлов. Отдельная глава посвящена математическим моделям кривых, поверхностей и тел.

Содержание книги соответствует курсу лекций, который авторы читают в МГТУ им. Н.Э. Баумана.

Для студентов, обучающихся по различным направлениям и специализациям современных информационных технологий. Может быть полезна специалистам по системам автоматизированного проектирования, разработчикам игр, программистам, пользователям современных графических пакетов.

УДК 004.92(075.8)

ББК 32.973

ISBN 978-5-7038-3015-4

© А.Н. Божко, Д.М. Жук,
В.Б. Маничев, 2007

© Оформление. Издательство
МГТУ им. Н.Э. Баумана, 2007

ОГЛАВЛЕНИЕ

ОГЛАВЛЕНИЕ	5
ПРЕДИСЛОВИЕ.....	8
I. ВВЕДЕНИЕ В КОМПЬЮТЕРНУЮ ГРАФИКУ	11
1.1. Направления и области использования компьютерной графики.....	12
1.1.1. Изобразительная компьютерная графика.....	12
1.1.2. Обработка и анализ изображений.....	13
1.1.3. Анализ сцен и распознавание образов.....	13
1.1.4. Когнитивная компьютерная графика.....	13
1.1.5. Области использования компьютерной графики	14
1.2. Растровая и векторная графика.....	14
1.2.1. Векторный формат	16
1.2.2. Растровый формат	17
1.3. Свет и физические основы цветовосприятия.....	18
1.3.1. Светотехнические величины	19
1.3.2. Зрительный аппарат человека	20
1.3.3. Чувствительность глаза.....	22
1.3.4. Дефекты цветового восприятия	24
1.3.5. Цветовые иллюзии	25
1.3.6. Хроматическая адаптация.....	27
1.4. Формирования цветных изображений.....	28
1.4.1. Базовые принципы описания цвета.....	28
1.4.2. Модель RGB.....	30
1.4.3. Модели CMYи CMYK	31
1.4.4. Модели YUV и YIQ.....	33
1.4.5. Модель CIE XYZ	34
1.4.6. Модели CIE Luv и CIE Lab	36
1.4.7. Интуитивные цветовые модели.....	37
1.4.8. Цветовой круг	39
1.4.9. Стандартные источники света CIE	39
1.4.10. Формирование цветов на экране монитора.....	40
1.4.11. Интерполяция цветов	43
1.5. Измерение цвета и калибровка технических средств	44
1.5.1. Системы управления цветом	45
1.5.2. Профили ICC.....	47
1.5.3. Инструменты для измерения цвета.....	48
1.5.4. Создание профиля монитора	50
1.5.5. Создание профиля сканера	51
1.5.6. Создание профиля печатающего устройства	53
1.5.7. Передача цветовых значений	54

1.6. Теоретические основы оцифровки	56
1.6.1. Типы сканирующих устройств	57
1.6.2. Разрешение	59
1.6.3. Глубина цвета устройства оцифровки	61
1.6.4. Диапазон оптических плотностей	62
1.6.5. Размеры изображений	64
1.6.6. Масштабирование	65
1.6.7. Дискретизация	67
Вопросы для самоконтроля	69
2. ТЕХНИЧЕСКИЕ СРЕДСТВА КОМПЬЮТЕРНОЙ ГРАФИКИ	71
2.1. Общие сведения об ЭВМ, используемых для обработки графической информации	71
2.1.1. Основные технические параметры ЭВМ	72
2.1.2. Классификация ЭВМ	73
2.1.3. Аппаратные средства ЭВМ	83
2.2. Графическая подсистема ЭВМ	105
2.2.1. Принципы работы графического адаптера	109
2.2.2. Технологии 3D-графики	128
2.2.3. Последовательность работы графического конвейера	149
2.2.4. Поколения графических процессоров	154
2.2.5. Мониторы	155
2.2.6. Проекторы	166
Вопросы для самоконтроля	170
3. МАТЕМАТИЧЕСКИЕ МОДЕЛИ ГЕОМЕТРИЧЕСКИХ ОБЪЕКТОВ	172
3.1. Представление кривых и поверхностей	172
3.1.1. Полигональные сетки	174
3.1.2. Представление полигональных сеток	174
3.1.3. Согласованность полигональных сеток	177
3.1.4. Уравнения плоскости	177
3.2. Параметрические кубические кривые	179
3.2.1. Основные положения	179
3.2.2. Кривые Эрмита	184
3.2.3. Кривые Безье	189
3.2.4. Однородные нерациональные B-сплайны	194
3.2.5. Неоднородные нерациональные B-сплайны	199
3.2.6. Рациональная форма кривых и сплайнов	205
3.2.7. Разбиение кривых	206
3.2.8. Преобразование представлений	209
3.2.9. Рисование кривых	210
3.2.10. Сравнение кубических кривых	214
3.3. Параметрические бикубические поверхности	216
3.3.1. Поверхности Эрмита	217
3.3.2. Поверхности Безье	222
3.3.3. Бисплайновые поверхности	223
3.3.4. Нормали поверхностей	223
3.3.5. Визуализация бикубических поверхностей	224

3.3.6. Поверхности второго порядка	228
3.4. Твердотельное моделирование	229
3.4.1. Основные положения	229
3.4.2. Регулярные булевы операции	232
3.4.3. Параметрическое моделирование геометрии	236
3.4.4. Заметание	238
3.4.5. Граничное представление	240
3.4.6. Модели пространственного разбиения	255
3.4.7. Конструктивная твердотельная геометрия	264
3.4.8. Сравнение представлений	267
Вопросы для самоконтроля	268
4. МЕТОДЫ, АЛГОРИТМЫ И ФОРМАТЫ ФАЙЛОВ КОМПЬЮТЕРНОЙ	
ГРАФИКИ	270
4.1. Методы и алгоритмы двумерной компьютерной графики	270
4.1.1. Входные и выходные данные растровые	271
4.1.2. Входные данные растровые, выходные данные векторные	291
4.1.3. Входные данные векторные, выходные данные растровые	303
4.1.4. Входные и выходные данные векторные	317
4.2. Методы и алгоритмы трехмерной графики и геометрии	346
4.2.1. Алгоритмы визуализации трехмерных моделей	346
4.2.2. Алгоритмы закрашивания видимых поверхностей	357
4.2.3. Детальное отображение поверхностей	364
4.2.4. Алгоритмы анимации трехмерных моделей	367
4.3. Форматы графических и геометрических файлов	376
4.3.1. Форматы векторных данных	376
4.3.2. Форматы растровых данных	385
4.3.3. Язык Adobe PostScript	388
Вопросы для самоконтроля	390
СПИСОК ЛИТЕРАТУРЫ	391

ПРЕДИСЛОВИЕ

Информационные технологии являются основными инструментами в решении сложных проблем, поставленных перед человечеством в XXI в. При этом одним из главных факторов, определяющих эффективность использования информационных технологий, служит уровень развития компьютерной графики. Оснащение компьютерной графики включает в себя технические средства, математические методы, модели и алгоритмы, программное обеспечение. Компьютерная графика занимает важнейшее место в решении одной из главных проблем практического использования информационных технологий – проблемы взаимодействия человека и вычислительной системы (проблемы HCI – Human Computer Interaction).

Возможности компьютерной графики в целом определяются уровнем развития технических средств и в особенности средств визуализации графических изображений. Математические основы дисциплины «Компьютерная графика» отличаются разнообразием используемых методов вычислительной математики, математического моделирования, статистики, математического программирования, дискретной математики и искусственного интеллекта.

Различные подсистемы компьютерной графики входят в состав практически всех современных программных продуктов — от простейших текстовых редакторов до сложнейших компьютерных игр с большим числом участников. Особое место они занимают в составе систем автоматизированного проектирования и пакетов, предназначенных для создания художественных и анимационных фильмов. Без преувеличения можно сказать, что знание основ компьютерной графики и умение работать с ее средствами необходимо практически каждому специалисту. Трудно себе представить плодотворную работу офиса, издательства, проектной организации, рекламного агентства, образовательного учреждения без использования компьютерных графических технологий.

Компьютерная графика как научное направление берет свое начало в 60-х годах прошлого столетия с защиты диссертации Айвена Сазерленда, в которой были рассмотрены новые возможности графического интерфейса с ЭВМ в интерактивном режиме. Графический интерфейс был реализован в первой интерактивной графической системе — Sketchpad, — разработанной Сазерлендом и продемонстрированной в Массачусеттском технологическом институте (МТИ) в 1963 г. В этой системе можно было с помощью светового пера рисовать на экране векторного дисплея простые геометрические фигуры и осуществлять с ними различные манипуляции. Однако это была скорее демонстрация возможностей техники и особенностей графического интерфейса, поскольку данная программа

не могла решать какие-либо практические задачи. Система была очень дорогой, поскольку почти вся вычислительная мощность высокопроизводительного (по меркам того времени) компьютера использовалась для управления дисплеем.

Менее чем за полвека своего развития компьютерная графика проделала громадный путь. Сейчас это самая динамично развивающаяся отрасль информационной технологии. В подтверждение этой мысли упомянем лишь о динамичном рынке графических ускорителей, где почти каждые полгода происходит смена поколений графических процессоров, а мощность современных графических вычислителей превысила потенциал центральных процессоров.

Предлагаемое учебное пособие предназначено для широкой аудитории пользователей, связанных с компьютерной графикой. В нем рассмотрены проблемы, интересующие специалистов по системам автоматизированного проектирования, разработчиков игр, программистов, пользователей современных графических пакетов и др. Она может оказаться полезной студентам, обучающимся по различным направлениям и специализациям современной информационной технологии.

В главе 1 рассмотрены основные направления и области применения компьютерной графики, общая структура процесса создания и визуализации графической информации. В цепочке преобразований, которые претерпевает компьютерная геометрическая модель, особое значение имеет зрительная система человека. Обсуждаются особенности зрительного аппарата человека, дефекты восприятия цвета, основные цветовые модели. Растровая графика — одно из самых востребованных направлений компьютерной графики. Здесь приведены основные сведения о преобразовании изображений в растровую форму и о базовых технических характеристиках средств оцифровки.

В главе 2 описано техническое обеспечение компьютерной графики. Основное внимание уделено центральным устройствам компьютеров, используемым при обработке графической информации, а также созданию математических моделей и фотореалистичной визуализации сложных изображений. Подробно описаны видеоподсистемы компьютеров, современные графические процессоры и аппаратная реализация алгоритмов фотореалистичной визуализации, периферийные устройства, используемые для визуализации и документирования графических изображений.

В главе 3 даны теоретические основы компьютерной графики, рассмотрены математические модели кривых, поверхностей и тел, широко применяемых в современных системах геометрического моделирования. Компактно изложены ключевые вопросы математического моделирования графики: сплайны, кривые Безье, параметрические поверхности, булевы операции, твердотельные модели трехмерных объектов и др.

Глава 4 посвящена алгоритмическому обеспечению компьютерной графики и описанию графических форматов. В ней приведены базовые алгоритмы растровой и векторной графики, к которым в первую очередь относятся универсальные способы машинного геометрического преобразования — перемещение, по-

ворот и масштабирование. В растровой графике фундаментальными являются алгоритмы заполнения и сглаживания. В основе функционирования любых систем трехмерного геометрического моделирования лежат алгоритмы визуализации трехмерных сцен. Любая сцена независимо от состава и технологии обработки хранится в файловой форме. Специалистами в области компьютерной графики разработано множество форматов хранения геометрических данных, но только немногие из них применяются в современных системах и пакетах и лишь единицы могут именоваться отраслевыми стандартами. В главе 4 также рассмотрены основные файловые форматы, предназначенные для записи геометрической и сопутствующей информации на машинные носители, и техника сжатия избыточных данных файловой формы хранения.

Авторы выражают глубокую признательность рецензентам — профессору А.Н. Данчулу и коллективу кафедры «Цифровые информационные системы» (зав. кафедрой профессор М.Ф. Меняев) — за полезные замечания и советы, которые были учтены.

Авторы будут признательны всем читателям за замечания по содержанию книги, которые можно направлять по электронным адресам: zhuk@bmstu.ru, manichev@bmstu.ru.

1. ВВЕДЕНИЕ В КОМПЬЮТЕРНУЮ ГРАФИКУ

Глава 1 представляет собой расширенное введение в компьютерную графику (КГ). В ней обсуждаются базовые понятия и соглашения этого плодотворного направления современной информационной технологии. Приводится классификация самых заметных направлений и областей приложения КГ. Дается описание основных подходов к представлению геометрических моделей и графических образов, а также сравнительный анализ векторных растровых геометрических моделей. Подробно рассматриваются основные световые модели и особенности восприятия цвета человеком.

Мы получаем информацию об окружающем мире с помощью органов чувств, которые можно рассматривать как информационные каналы. Основную часть необходимых данных человек получает с помощью органов зрения. Информация, поступающая через органы зрения, обрабатывается головным мозгом и на ее основе принимаются решения о дальнейших действиях. При обработке поступающей информации головной мозг использует два механизма мышления:

- осознанный (логико-вербальный) манипулирует абстрактными последовательностями символов (объектов) с учетом семантики символов и практических знаний, связанных с символами;
- неосознанный (интуитивный, образный) работает с чувственными образами и представлениями о них.

Первый механизм отличается невысокой скоростью обработки информации, поскольку связан с распознаванием и идентификацией символов, осознанием семантики последовательности символов и логической обработкой данной информации. Результат содержит точную (часто количественную) оценку полученной информации. Второй механизм обеспечивает максимальную скорость обработки информации, воспринимаемой в форме графических изображений. При этом имеет место качественная интегральная оценка информации. С точки зрения максимизации объема и скорости получения информации сочетание обоих механизмов обеспечивает наилучшие результаты. Таким образом, в процессе взаимодействия человека с компьютером необходимо сбалансированное использование как символьной, так и образной информации.

Рассмотрим условия, в которых наблюдатель получает полезную зрительную информацию. В общем случае группа источников освещает объекты сцены естественного или искусственного происхождения. Световой поток, падающий на объект, отражается от его поверхностей, частично проходит через него и, возможно, дополняется светом, идущим от самого объекта. Затем часть светово-

го потока, содержащего информацию об объекте, распространяясь в окружающей среде, попадает в органы зрения человека. Иногда на эту составляющую светового потока воздействуют фильтр или оптическая система, находящиеся между объектом и наблюдателем. Органы зрения воспринимают видимую часть светового потока, пришедшую от объекта, и передают ее для дальнейшей обработки. Информация, заключенная в этом сигнале, преобразуется головным мозгом в осознанное изображение. Человек наделяет это представление всеми характеристиками, которые отличают структурированную информацию от хаоса и превращают его в сцену, обладающую семантикой и прагматикой. Итак, человек получает всю зрительную информацию в графическом виде, которая после дополнительной обработки головным мозгом преобразуется в другие виды информации.

Средства компьютерной графики (КГ) используют для моделирования описанных процессов получения графической информации и создания изображений, по своим изобразительным свойствам максимально приближенных к объектам реального мира.

1.1. Направления и области использования компьютерной графики

Выделяют следующие направления КГ:

- изобразительная КГ;
- обработка и анализ изображений;
- анализ сцен и распознавание образов;
- КГ для визуализации синтезированных абстрактных изображений (когнитивная КГ — графика, способствующая познанию).

1.1.1. Изобразительная компьютерная графика

В этом направлении КГ изображение является не промежуточным результатом, не исходными данными для выполнения расчетов или реализации алгоритмов, а основным продуктом. Источником графических данных служат образы искусственного происхождения, созданные средствами геометрического моделирования.

К основным задачам изобразительной графики относятся:

- построение модели сцены по реальному или идеальному прообразу;
- преобразование модели сцены для генерации требуемых технических параметров или получения визуального эффекта;
- визуализация и вывод сцены на устройство печати или экран монитора.

Изобразительная КГ нашла широкое применение в различных отраслях народного хозяйства: компьютерные презентации, графические приложения в полиграфии, анимационные ролики, компьютерные игры и многое другое.

1.1.2. Обработка и анализ изображений

В этом направлении КГ основным источником графических данных является реальная плоская или трехмерная сцена. Процедура оцифровки позволяет представить сцену в цифровом виде и передать для обработки в вычислительную систему. Машинное описание, полученное таким способом, представляет собой образ, который фиксирует только видимую страту оригинала, поэтому модель обладает ограниченными прогностическими и расчетными возможностями.

В данном разделе КГ решаются следующие типовые задачи:

- повышение качества изображения;
- оценка изображения, определение его формы, местоположения, размеров и других параметров;
- преобразование изображения для получения требуемых декоративных свойств или технических характеристик.

Самым известным примером этого направления можно считать растровую изобразительную графику, где объектом преобразования является оцифрованный оригинал, например фотография или рисунок, который подвергается преобразованиям, решающим поставленную художественную или техническую задачу.

1.1.3. Анализ сцен и распознавание образов

В этом направлении КГ объектами анализа и обработки служат их графические модели, размещенные в некоторой естественной или искусственной среде. Основными задачами при распознавании объектов являются локализация объекта, определение его типа, а также расчет геометрических и структурных характеристик. Задачи такого вида решаются в системах машинного зрения (роботы), при анализе рентгеновских снимков в медицине и технике, в процессе расшифровки результатов аэрофото- и космической съемки.

1.1.4. Когнитивная компьютерная графика

Когнитивная КГ занимается синтезом и визуализацией абстрактных изображений, не имеющих реальных физических аналогов или первообразов. Простейшим примером такого использования машинной графики является построение диаграмм, гистограмм, графиков распределений, которые в наглядной графической форме представляют абстрактные сущности, возникающие в результате созидательной деятельности человека.

Основная проблема когнитивной КГ — создание моделей представления знаний, допускающих компактное и единообразное описание логического и образного знания. Для этого требуется решить три важнейшие задачи:

- разработка техники визуализации данных и знаний;
- представление знаний в компактной форме, доступной для восприятия и обработки наблюдателем;
- синтез правил, гипотез, механизмов объяснения и предсказания.

1.1.5. Области использования компьютерной графики

Современный пользователь общается с компьютером посредством графического интерфейса. Графический интерфейс как посредник между человеком и машиной косвенно участвует во всех реализациях информационной технологии. Кроме этого существует множество направлений, где достижения компьютерной графики и геометрии играют принципиальную созидательную роль:

- системы автоматизации научных исследований, системы автоматизированного проектирования, системы автоматизации технологической подготовки производства, автоматизированные системы организации и управления производственными процессами, геоинформационные системы и др.;
- системы организации и управления бизнес-процессами;
- геометрическое моделирование объектов реального мира;
- компьютерная живопись и графика для печати и онлайн-выходов изданий;
- создание статичных и анимированных изображений для Интернета;
- компьютерные издательские и полиграфические системы и другие приложения в области средств массовой информации;
- моделирование и изготовление одежды;
- архитектурное проектирование и строительство;
- системы виртуальной реальности: тренажеры и симуляторы, которые используются для подготовки специалистов в различных отраслях народного хозяйства;
- компьютерные игры и другие приложения в области развлечений.

Области применения средств компьютерной графики в книге не рассматриваются, поскольку они детально освещаются в специальной и научно-популярной литературе.

1.2. Растровая и векторная графика

Компьютер может обрабатывать данные только если они представлены в цифровом виде. Описание графических данных выполняется в соответствии с определенными правилами, совокупность которых часто называется форматом. Важнейшей частью любого графического формата является пространство, в котором расположены объекты сцены или элементы изображения.

В КГ используются два основных варианта описания графических данных, отличающихся количеством независимых координат, необходимых для определения положения графических объектов и их элементов:

- плоское, или двухмерное (2D), которое оперирует только двумя независимыми координатами;
- объемное, или трехмерное (3D), требующее задать три независимые координаты.

Двухмерная сцена может формироваться из объектов, расположенных как в координатной плоскости, так и из проекций всех видимых объектов на плоскость изображения. Компьютерная графика изобилует примерами плоских сцен: цифровые фотографии; сканированные рисунки и картины; изображения, созданные растровыми и двухмерными векторными редакторами.

Трехмерная КГ — одно из наиболее динамично развивающихся направлений информационных технологий. Трехмерное описание геометрии намного сложнее двухмерного, поэтому массовое применение достижений этой отрасли долгое время сдерживалось вычислительными возможностями персональных компьютеров. Основными заказчиками и потребителями трехмерной графики являются компьютерные игры, киноиндустрия, компьютерные тренажеры и симуляторы.

Множество способов представления графических объектов в памяти вычислительной системы можно разбить на два больших класса, существенно отличающихся базовыми принципами моделирования геометрии:

- геометрический объект представляется посредством координат характеристических точек и математических моделей кривых и поверхностей;
- геометрический универсум разбивается на множество элементарных областей (точек или объемов), а каждый объект представляется как множество забираемых им областей дискретного пространства.

Наиболее известная модель первого класса называется векторной. В ней характеристические точки объекта соединяют отрезки прямых линий — векторы. Векторная модель может быть использована для представления плоских и пространственных объектов. Она лежит в основе многих популярных векторных графических форматов. Большинство современных пакетов (системы автоматизированного проектирования, пакеты изобразительной графики, автоматизированные системы научных исследований) используют для обработки графической информации векторное описание.

Известно много вариантов описания геометрии как набора элементов некоторого дискретного универсума. Большинство из них имеют либо теоретический интерес, либо очень узкую область применения. Наибольшее распространение в современных системах КГ получила так называемая растровая модель. В ней описываемое пространство покрывается регулярной сетью одинаковых двухмерных или трехмерных элементов. Элементы сетки имеют форму квадрата или прямоугольника (куба или параллелепипеда в трехмерном случае). Их принято называть пикселями (вокселями — для трехмерного пространства), а саму сеть — растром.

Совокупность элементов растра задает изображение, заполняя область пространства, которое этот объект занимает. Элементы растра — независимые об-

разования, которые могут принимать любые значения цвета и тона из допустимого диапазона. Если размеры пикселей невелики, то наблюдатель воспринимает дискретную растровую картинку как целостный образ — фотографию или рисунок.

Многие популярные графические форматы предназначены для хранения растровых данных в самой простой редакции — в виде плоского массива однородных точек. В геоинформационных системах нашли применение другие модели пространственного разбиения, например регулярно-ячеистые, квадратомигические и др.

Помимо растровых и векторных моделей иногда используют их сочетание — так называемые гибридные модели. Все эти модели при описании одной и той же области пространства могут быть взаимно преобразованы. Как правило, осуществить переход от векторного способа хранения изображения к растровому несложно. При таком преобразовании необходимо задать масштаб для пересчета координат векторов в пиксели. Обратный переход в общем случае является достаточно сложным и неоднозначным, поскольку связан с интеллектуальной интерпретацией точечных образов.

Выбор растрового или векторного формата зависит от поставленных целей и решаемых задач. Практически все современные устройства вывода графических изображений являются растровыми и отображают любое изображение в виде совокупности точек. Это, в частности, относится к самым распространенным представителям компьютерной периферии: мониторам и принтерам. На экране монитора или принтерном оттиске отображается растровый образ независимо от его исходной формы хранения, которая может быть векторной или растровой.

1.2.1. Векторный формат

Векторное представление графических данных предполагает наличие совокупности базовых геометрических элементов, называемых примитивами. Обычно это фигуры простой формы. Так, для создания плоских объектов используются прямоугольники, треугольники, овалы, прямые; для трехмерных фигур — кубы, параллелепипеды, сферы, эллипсоиды, конусы и др. Для всех перечисленных объектов известны математические модели, задающие их форму, положение и габаритные размеры. Кроме того, в векторном файле хранятся оформительские атрибуты примитивов: цвет, тип заливки, толщина граничного контура, окончание линии и др. Кривые сложной формы представляются в виде ломаной со звеньями, размеры которых не распознаются наблюдателем как прямые. Звенья ломаных называются векторами. Векторы задаются координатами начальной и конечной точек или координатами начала, длиной и направлением.

Объекты сложной геометрии образуются из базовых примитивов при помощи допустимого множества операций. В двухмерных векторных редакторах это множество ограничивается группированием, наложением и простейшими бу-

левскими операциями. В трехмерных редакторах набор допустимых преобразований намного шире. В него могут входить регулярные булевские операции, замещение, смещение и вращение контура, а также множество других операций, зависящих от вида примитивов и системы моделирования.

Векторное описание отличается несколькими значимыми достоинствами. Оно позволяет задавать достаточно сложные изображения с высокой точностью. Благодаря этому векторные форматы широко используются в САД-системах для геометрического моделирования технических объектов, когда необходимо детально представить форму и положение реальной машины или прибора.

Объекты, заданные в векторной форме, легко масштабируются, а точность их отображения определяется только параметрами устройства вывода. Как правило, такое описание намного компактнее, чем растровое, когда форма объекта задается множеством пикселей, заполняющих его периметр или объем.

Векторные объекты разрешается масштабировать в очень широких пределах без потерь визуального качества и точности моделирования.

К недостаткам векторных изображений относятся ограниченные возможности создания полных моделей для реальных сцен с учетом всего множества деталей обстановки. Кроме того, в векторной графике существуют серьезные проблемы с фотореалистичной визуализацией созданных моделей. Обычно недостатки векторных моделей компенсируются использованием технологий растровых моделей на этапе визуализации полученных изображений.

1.2.2. Растровый формат

Растровая модель представляет собой отображение непрерывного оригинала во множество элементов абстрактного дискретного пространства, называемого растром. Самый распространенный вариант этой парадигмы — это двухмерный растр, когда плоское изображение задается в виде множества точек регулярной ортогональной сеткой, состоящей из одинаковых элементов.

Двухмерная растровая графика — одно из самых популярных направлений КГ. Она поддерживает способ работы с цифровыми оригиналами, которые близки классическому рисованию или живописи. Простота растровой формы хранения данных обусловила широкое распространение растровых картинок в сети Интернет. Основным массивом графики, циркулирующим в сети, являются двухмерные растровые образы. Существуют и другие причины высокой популярности этого подхода — растровый принцип действия основной части устройств вывода (принтеров и мониторов).

Большая часть особенностей этой формы хранения графических данных связана с ее дискретностью. Только в растровой графике можно получить образ такого качества, который соответствует категории фотореалистичного. Причина этого заключается не в особых креативных свойствах растровых редакторов. Компьютерные образы высокой достоверности получаются оцифровкой реальных физических прототипов, а эта процедура создает только растровые изображения.

Размер растрового файла не зависит от содержимого отображаемой области. Уменьшить его можно с помощью алгоритмов сжатия. Если один оригинал представить в растровой и векторной формах, то в большинстве случаев первая форма потребует заметно больше дискового пространства, чем вторая.

Растровые изображения могут потерять качество в процессе геометрических преобразований, что особенно проявляется при масштабировании и повороте. Проведем простой мысленный эксперимент. Пусть растровое пространство представляет собой плоскость, обладающую способностью к бесконечному растяжению. Если значительно увеличить эту плоскость, то ее точки приобретут заметные размеры, а изображение — зернистую структуру.

В классическом растровом формате не существует понятия объекта. Для обрабатывающей программы нет овалов, линий, треугольников, а существуют только точки, которые для редактора совершенно независимы друг от друга. По этой причине для точечных изображений задачи идентификации и распознавания решаются с большим трудом, а часто и вовсе не имеют решения.

Несмотря на видимую простоту, в этом разделе компьютерной графики есть немало трудных для понимания тем.

1.3. Свет и физические основы цветовосприятия

Понятия света и цвета в КГ являются основополагающими. Известно, что свет имеет дуальную природу и его можно рассматривать либо как поток частиц различной энергии (тогда его цвет определяет энергия частиц), либо как поток электромагнитных волн (в этом случае цвет определяется длиной волны или частотой). Далее свет будет рассматриваться как поток электромагнитных волн. Электромагнитная волна характеризуется своей амплитудой A , длиной волны λ или частотой колебаний, фазой и поляризацией. Видимый свет — узкий участок спектра электромагнитных волн с длиной волны $\lambda = 400 \dots 700$ нм. Амплитуда определяет энергию волны, которая пропорциональна квадрату амплитуды. Фаза и поляризация электромагнитных волн в дальнейшем учитываться не будут.

Энергия, переносимая электромагнитной волной, зависит от ее длины — увеличивается с уменьшением длины волны. По этой причине коротковолновые ультрафиолетовые лучи по своей энергии значительно превосходят длинноволновые инфракрасные лучи. Какое влияние все это оказывает на характеристики света как переносчика информации? Несколько упрощая ситуацию, можно сказать, что общее число всех световых волн в световом луче, которое эквивалентно его общей энергии, обуславливает интенсивность, или яркость, света. Пропорции, в которых представлены различные световые волны, влияют на его хроматические характеристики, поэтому доминирующие длины волн светового потока определяют его цветность. Свет как носитель информации содержит только два основных вида данных — информацию о яркости и цвете.



Рис. 1.1. Пример спектрального состава пучка света

На практике редко приходится сталкиваться с монохроматическим светом, т. е. с электромагнитными колебаниями какой-то одной определенной длины волны (примером такого света является излучение лазера). Обычно свет представляет собой непрерывный поток волн с различными длинами волн и различными амплитудами. Такой поток можно характеризовать так называемой

энергетической спектральной кривой $F(\lambda)$, где само значение функции $F(\lambda)$ представляет собой энергетический вклад колебаний с длиной волны λ в суммарный световой поток.

Общая мощность видимого светового потока равна интегралу от спектральной функции по всему видимому диапазону длин волн. Типичная спектральная кривая приведена на рис. 1.1.

1.3.1. Светотехнические величины

Область физической оптики, посвященная измерению энергии, переносимой световыми волнами и связанными с ней величинами, называется фотометрией. Рассмотрим основные светотехнические величины и единицы их измерения.

Световой поток (F). Световой поток через некоторую поверхность S равен суммарной энергии, переносимой световыми волнами сквозь эту поверхность. Световой поток измеряется в люменах (лм).

Сила света (I). Источник в общем случае может иметь неравномерное излучение по разным направлениям. Плотность светового потока в телесном угле выбранного направления называется силой света и определяется по формуле

$$I = dF/d\omega,$$

где F — световой поток, проходящий через площадку dS ; ω — телесный угол.

Единица силы света называется канделлой (кд). Если в телесном угле, равном одному стерadianу (ср), проходит, равномерно распределяясь, световой поток в 1 лм, то сила света в этом направлении равна одной канделле: 1 кд = 1 лм/1 ср.

Освещенность (E). Плотность светового потока по поверхности S , на которую он падает, называется освещенностью. Освещенность можно рассчитать по следующей формуле:

$$E = F/S.$$

Единицей освещенности является люкс (лк). Освещенность в 1 лк создается световым потоком в 1 лм на площади в 1 м²; 1 лк = 1 лм/1 м². Освещенность экрана в кинотеатре составляет приблизительно 200 лк. Освещенность объекта передачи в телевизионной студии достигает 2000 лк.

Яркость (B). Яркостью называется плотность силы света. Эту величину можно определить по формуле

$$B = I/S,$$

где I — сила света, кд; S — площадь излучения, м^2 .

Единицей яркости является канделла на квадратный метр: 1 кд/м^2 . Яркость экрана кинескопа на белых участках изображения составляет от 40 до 80 кд/м^2 .

Само понятие цвета тесно связано с тем, как человек (органы его зрения) воспринимает свет; можно сказать, что оттенок цвета создается в глазу и головном мозге человека. Поскольку светоприемником в КГ являются органы зрения человека, при создании средств КГ необходимо учитывать физические и психофизиологические особенности зрения.

1.3.2. Зрительный аппарат человека

Рассмотрим, каким именно образом происходит восприятие света человеческим глазом. На рис. 1.2 показано строение глазного яблока человека. Глаз расположен в глазнице черепа. Из глазного яблока выходит глазной нерв, содержащий около 800 000 волокон и соединяющий его с головным мозгом. Глазное яблоко состоит из внутреннего ядра и окружающих его трех оболочек — наружной, средней и внутренней. Наружная оболочка — белочная оболочка — представляет собой жесткую непрозрачную капсулу, переходящую спереди в прозрачную роговицу, через которую в глаз проникает свет. Под ней находится сосудистая оболочка, переходящая спереди в радужную оболочку, в центре которой расположен зрачок. Под действием мышц зрачок способен сужаться и

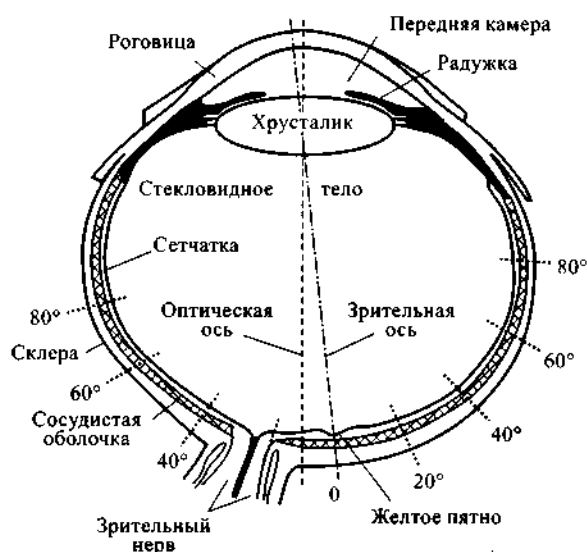


Рис. 1.2. Строение глазного яблока человека

расширяться. В сосудистой оболочке находится ресничная мышца, которая регулирует кривизну хрусталика. Во внутренней оболочке глаза — сетчатке — имеются светочувствительные рецепторы — палочки и колбочки. В них энергия света преобразуется в сигнал, который передается по зрительному нерву в мозг. Колбочки сосредоточены в центре сетчатки, в желтом теле напротив зрачка. Они обеспечивают дневное зрение, воспринимая цвет, форму и детали предметов. На периферии сетчатки расположены только палочки, которые раздражаются слабым сумеречным светом, но не чувствительны к цвету.

Колбочки и палочки содержат зрительные пигменты, которые очень похожи на любые другие пигменты в том, что они поглощают свет и степень поглощения зависит от длины волны. Важное свойство зрительных пигментов состоит в том, что когда зрительный пигмент поглощает фотон света, то изменяется форма молекулы и в то же время происходит переизлучение света. Пигмент при этом изменяется, преобразованная молекула поглощает свет хуже, чем прежде, т. е. как часто говорят, «отбеливается». Изменение формы молекулы и переизлучение энергии некоторым образом иницируют светочувствительную клетку к выдаче сигнала.

Информация от светочувствительных рецепторов (колбочек и палочек) поступает к другим типам клеток, соединенным между собой. Специальные клетки передают информацию в зрительный нерв. Волокно зрительного нерва обслуживает несколько светочувствительных рецепторов, т. е. некоторая предварительная обработка изображения выполняется непосредственно в глазу.

Область сетчатки, в которой волокна зрительного нерва соединены, лишена светочувствительных рецепторов и называется слепым пятном.

Внутреннее ядро глазного яблока вместе с роговицей образует оптическую систему глаза, которая состоит из хрусталика, стекловидного тела и водянистой влаги камер глаза. Прозрачный и эластичный хрусталик расположен за зрачком и имеет форму двояковыпуклой линзы. Он вместе с роговицей и внутриглазными жидкостями преломляет входящие в глаз лучи света и фокусирует их на сетчатке.

Рефлекторный механизм, с помощью которого лучи света, исходящие от объекта, фокусируются на сетчатке, называется аккомодацией. Он выполняет две функции:

- рефлекторное изменение диаметра зрачка. При ярком свете кольцевая мускулатура радужки сокращается, а радиальная расслабляется; в результате происходит сужение зрачка и количество света, попадающего на сетчатку, уменьшается, что предотвращает ее повреждение. При слабом свете, наоборот, радиальная мускулатура сокращается, а кольцевая расслабляется. При сужении зрачка увеличивается глубина резкости и поэтому различия в расстоянии от объекта до глаза меньше сказываются на изображении;

- преломление света. От объекта, удаленного на расстояние больше 6 см, в глаз поступают практически параллельные лучи света, тогда как лучи, идущие от более близких предметов, заметно расходятся. В обоих случаях для того что-

бы свет сфокусировался на сетчатке, он должен быть преломлен, и для близких предметов преломление должно быть более сильным. Глаз человека способен точно фокусировать свет от объектов, находящихся на расстоянии от 25 см до бесконечности. Форма роговицы не может изменяться, поэтому рефракция здесь зависит только от угла падения света на роговицу, который, в свою очередь, зависит от удаленности предмета. В роговице происходит наиболее сильное преломление света, а функция хрусталика состоит в окончательной «наводке на резкость». При сокращении ресничной мышцы хрусталик меняет кривизну, приспособляясь для восприятия дальних или ближних предметов. Преломившиеся лучи света от рассматриваемого предмета, падая на сетчатку, образуют на ней уменьшенное обратное изображение предмета. Однако мы видим предметы в прямом виде благодаря повседневной тренировке зрительного анализатора, что достигается формированием условных рефлексов, показаниями других анализаторов, их взаимодействиями, постоянной проверкой зрительных ощущений, повседневной практикой.

Двигательный аппарат каждого глаза состоит из шести мышц, сокращения которых позволяют изменять направление взгляда. У людей с нормальным зрением на сетчатке возникает четкое изображение предметов, так как оно сфокусировано на центре сетчатки.

1.3.3. Чувствительность глаза

Исследования показали, что многие биологические и технические системы воспринимают цветовую и яркостную информацию отдельно. Это утверждение справедливо и для зрения человека. За цветовое и яркостное восприятие глаза отвечают два различных вида нервных клеток — колбочки регистрируют цветовую компоненту светового потока, а палочки воспринимают его яркостную составляющую.

Палочки образуют однородную популяцию нервных клеток, они демонстрируют высокую чувствительность к световому потоку, но при этом способны реагировать только на суммарную энергию света. Палочки позволяют человеку распознавать предметы в условиях плохой освещенности. Они регистрируют окружающий мир как ахроматическую, лишенную цветовых нюансов среду, наполненную серыми предметами.

Чувствительность палочек к свету неоднородна, при ярком свете она мала, но при низких уровнях освещенности возрастает и обеспечивает способность человека видеть в условиях плохой видимости. Палочки содержат пигмент с максимальной чувствительностью на длине волны около 510 нм (рис. 1.3, точечная линия) в зеленой части спектра. Пигмент палочек из-за его цвета часто называется зрительным пурпуром.

Кривая общей (палочек и колбочек) спектральной чувствительности глаза для яркого освещения показана на рис. 1.3 сплошной линией. Очевидно, что ре-

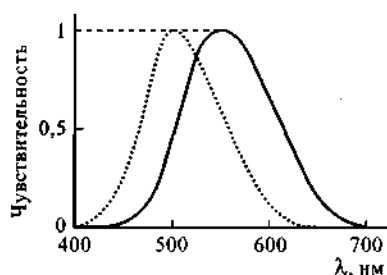


Рис. 1.3. Общая спектральная чувствительность глаза:

..... — чувствительность палочек;
 — суммарная чувствительность палочек и колбочек

зультатирующая чувствительность колбочек и чувствительность палочек имеет максимум на длине волны около 550 нм, что соответствует желто-зеленому свету, но при этом чувствительность палочек почти в 1000 раз выше, чем у колбочек. Таким образом, максимум чувствительности зрения человека лежит в желто-зеленой области спектра.

Графики, показанные на рис. 1.3, подтверждают, что в условиях слабой освещенности цветное зрение практически отсутствует. Например, отклик на красный цвет ($\lambda = 700$ нм) при низких уровнях освещения (см. точечную кривую на рис. 1.3) практически равен нулю.

Поэтому красный цвет ночью будет выглядеть черным.

Физиологами и оптиками давно установлен факт избирательной чувствительности зрения человека к волнам различной длины. Человек хорошо видит зеленый цвет, несколько хуже — красный, и хуже всего — синий цвет. Как объяснить этот цветовой феномен? Существует три типа колбочек, отличающихся фоточувствительным пигментом. Колбочки обычно называют синими, зелеными и красными в соответствии с наименованием цвета, для которого они имеют максимальную чувствительность. Упомянутые три пигмента имеют максимум поглощения приблизительно на 430, 530 и 560 нм.

На рис. 1.4 представлены графики функций чувствительности для всех трех типов колбочек. Точнее, этим длинам волн соответствует не синий, зеленый и красный цвета, а фиолетовый, сине-зеленый и желто-зеленый. Поэтому более корректным будет использование названий коротко-, средне- и длинноволновые колбочки.

Как видно на рис. 1.4, области чувствительности колбочек значительно перекрываются. Поэтому, как правило, в процессе цветового восприятия возбуждаются все три вида колбочек. Почему, несмотря на это, мы так хорошо различаем цвета? Это проблема пока не получила исчерпывающего решения в физиологии зрения.

Итак, эффективности поглощения световых волн существенно различаются. Особенно хорошо человек воспринимает зеленый свет, красный свет — уже несколько хуже, а синий — плохо. Это приводит к тому, что цветовые составляющие цветного изображения вносят разные вклады в ощущение яркости. Наименьший вклад в общую яркость вносит синяя составляющая.

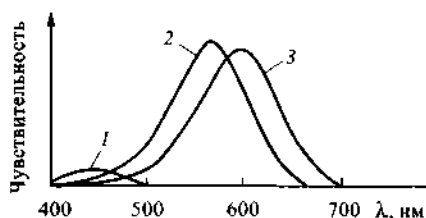


Рис. 1.4. Функции чувствительности трех типов колбочек:

1 — синих; 2 — зеленых; 3 — красных

Низкая чувствительность зрения человека к синим цветовым тонам является также причиной того, что синяя окраска фона хорошо подходит для цветных диапозитивов. Если черный шрифт напечатан на белом фоне, то шрифт и фон воспринимаются одинаково четко. Если же, например, белый шрифт напечатан на синем фоне, то значение фона как бы теряется, и в ощущении изображения доминирует шрифт или остальные элементы изображения с другой окраской.

Избирательная чувствительность зрения человека — экспериментально подтвержденный факт. Более того, исследователи в области психологии зрительно-го восприятия провели многочисленные тесты для получения количественных оценок вкладов отдельных цветовых составляющих. Установлено, что для большинства людей доли цветовых координат распределяются следующим образом: 59 % — зеленый, 30 % — красный и 11 % — синий цвет. Иными словами, если известны яркости зеленой, красной и синей составляющих, то суммарную яркость нельзя получить простым суммированием. Ее следует вычислять по примерной формуле:

$$\text{Яркость} = 0,59 \times \text{зеленый} + 0,3 \times \text{красный} + 0,11 \times \text{синий}.$$

В силу того, что коэффициент преломления в радужке и хрусталике глаза растет с увеличением частоты света, глаз не избавлен от хроматической аберрации, т. е. если изображение сфокусировано для одной из длин волн, то для других длин волн изображение будет расфокусировано. Хрусталик оптимально фокусирует на сетчатке свет с длиной волны около 560 нм. Поскольку пики чувствительности средне- и длинноволновых колбочек (530 и 560 нм соответственно) близки друг к другу, изображения для этих колбочек могут быть сфокусированы одновременно. Изображение же для коротковолновых палочек будет размытым. Поскольку степень фокусировки разная для различных длин волн, то не требуется одинаковой разрешающей способности глаза для разных типов колбочек. В глазу человека на одну коротковолновую колбочку приходится 20 средне- и 40 длинноволновых колбочек. В этой связи понятно, почему ширина полосы пропускания для холодных, коротковолновых цветов в телевидении может быть выбрана существенно меньшей без субъективно заметной потери верности воспроизведения.

1.3.4. Дефекты цветового восприятия

Всем известны такие дефекты зрения, как близорукость, дальнозоркость, астигматизм. Существуют правовые нормы, которые ограничивают дееспособность людей с этими дефектами зрения. Недостатки цветового восприятия встречаются гораздо реже. Исключением является дальтонизм, про который, видимо, знают все. Известность этого явления в немалой степени объясняется его высокой цитируемостью в кроссвордах и сканвордах, но объективные исследования показали, что число людей, страдающих дальтонизмом, оказалось не-

ожиданно велико. Почти каждый десятый человек на планете не способен различать цветовые оттенки в большей или меньшей степени! Дальтонизм — не единственная аномалия цветового зрения человека. Можно привести обширную выборку медицинских диагнозов, относящихся к нарушениям цветового аппарата. Трудно оспорить тот факт, что должны существовать профессиональные ограничения для людей с аномалиями цветового восприятия. Индивидуум с большими диоптриями не может работать часовщиком или претендовать на профессию нейрохирурга. Наличие дихроматопсии или протаномалии у дизайнера делает сомнительными все его заключения и рекомендации в области допечатной подготовки цветных публикаций высокого качества.

Физиологи и специалисты в области психологии разработали несколько тестовых испытаний, предназначенных для проверки зрения человека на наличие дефектов цветовосприятия. Наиболее распространенный в сфере графического дизайна тест Farnsworth-Munsell 100 позволяет обнаружить отклонения испытуемого от нормы цветовосприятия средствами простых проверок. Известны различные реализации этого теста, чаще всего он предлагается испытуемому в форме некоторого набора, напоминающего настольную игру. Тест состоит из цветowych образцов, которые разбиты на четыре группы, в каждую из них входит по 21 цветовому экземпляру. Цветовые образцы представляют собой цилиндры, которые по своим размерам близки к обычным шашкам. Верхние части цилиндров окрашены различными цветовыми оттенками, нижние части пронумерованы. Задача испытуемого — расставить цилиндрики по критерию цветового подобия. После того, как все фишки отсортированы, требуется сравнить номера серий с образцами, которые хранятся в таблицах, или ввести данные в обрабатывающую программу. По расхождениям с эталонной последовательностью можно судить о наличии аномалий цветовосприятия испытуемого.

Тест Ishihara состоит из набора специально разработанных изображений, в которых на хаотичном фоне нарисована простая фигура или нанесен некоторый регулярный узор. Эти изображения предлагаются для просмотра испытуемому, который должен распознать узор или рисунок. По результатам испытания можно диагностировать большую часть хроматических аномалий человека.

1.3.5. Цветовые иллюзии

Цветовые иллюзии — частный случай более общего явления, известного под названием оптические иллюзии. Академические словари объясняют этот феномен обманом зрения или ошибками в оценке геометрических характеристик и физических параметров объектов окружающей среды. Ошибки эти весьма многочисленны, разнообразны и с трудом поддаются объяснению.

Трудно поверить, глядя на рис. 1.5, что все горизонтальные линии являются параллельными прямыми. Этот поразительный пример производит впечатление ловко проделанного фокуса.

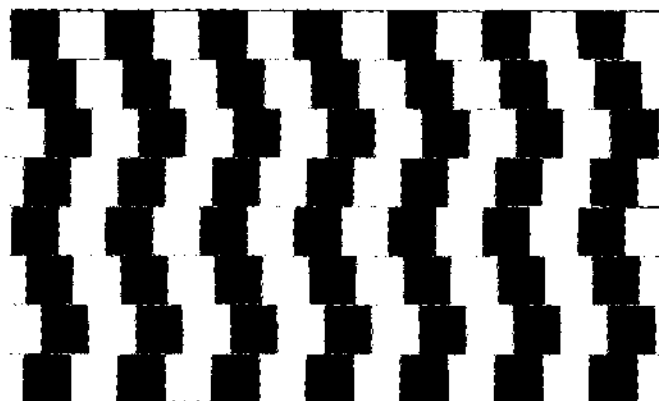


Рис. 1.5. Пример оптической иллюзии

На рис. 1.6 показана регулярная сетка, в узлах которой расположены круги небольшого размера. Это классический пример, демонстрирующий оптическую иллюзию в самой простой форме. Испытуемому предлагается подсчитать количество кружков разного цвета. Обычно уже после обработки первого ряда начинают путаться люди с самой устойчивой психикой и идеальным зрением.

На рис. 1 цветной вклейки приведен пример иллюзии цветового восприятия. Прямоугольное поле заполнено в шахматном порядке клетками светло-зеленого цвета, на диагоналях этой фигуры размещены прямоугольники отличного цвета. Требуется оценить сходство цветов клеточек, заполняющих диагонали. Кажется, что диагонали этой фигуры закрашены разным цветом, но инструментальная проверка или осмотр изображения при большом увеличении свидетельствуют о полной хроматической тождественности. Можно выиграть любое пари, что

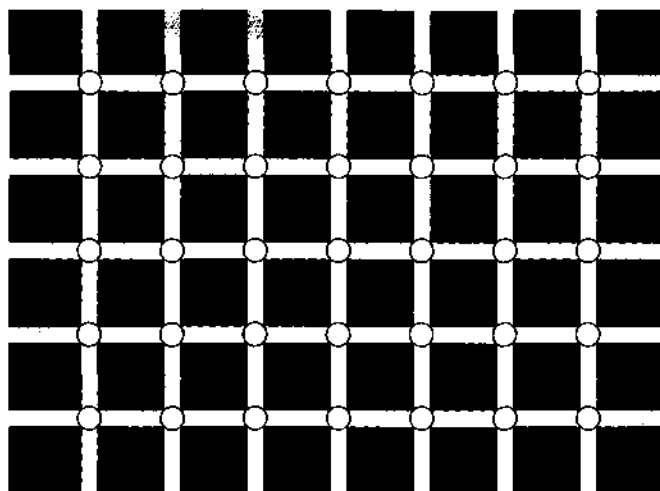


Рис. 1.6. Пример иллюзии восприятия тонов

десять из десяти опрошенных, не знакомых с этим тестом, зафиксировали отличие цветов прямоугольников, расположенных на диагоналях. Объективная инструментальная проверка дает совершенно иной результат — все клеточки диагоналей имеют одинаковый цвет (R234, G35, B120).

Можно продолжать ряд примеров, но даже на основе приведенных данных позволительно сделать вывод о релятивизме цветового восприятия человека. Мозг человека обрабатывает цветовые данные не как объективный феномен, достоверность которого никак не связана с внешними условиями, а относительно в тесной связи с окружающими объектами, средой и предисторией.

1.3.6. Хроматическая адаптация

Проведем мысленный эксперимент, последствия которого, впрочем, легко предсказать заранее. Пусть имеется некоторая сцена или цветное изображение, которые наблюдаются при фиксированных условиях освещения, например при дневном свете. Что произойдет, если резко изменить условия освещения, например заменить естественный свет искусственным флуоресцентным источником с заметным сдвигом в сторону зеленой части спектра. Спустя некоторое время наблюдатель будет видеть сцену в тех же красках, что и прежде. Голубое платье останется голубым, а шляпка красного цвета сохранит свой первоначальный цвет, несмотря на явное противоречие объективным физическим законам. Явление, которое иллюстрирует этот мысленный эксперимент, называется хроматической адаптацией — способностью системы зрения человека приспосабливаться к изменившимся условиям освещения и сохранять неизменными исходные цвета объектов сцены или элементов изображения. Дарвинисты утверждают, что эволюция — это накопление полезных признаков, наверное, этим качеством обладает и свойство хроматической адаптации. Но в области предпечатной подготовки цветных изданий она создает немало проблем для дизайнеров и полиграфистов.

Человек не в силах приказать своему мозгу остановить процесс адаптации, но можно, по крайней мере, исключить основные источники ошибок при принятии ответственных решений по управлению цветом. Вся предпечатная подготовка цветных публикаций должна проходить в условиях освещения, основные параметры которого соответствуют стандартам. Неслучайно многие препресс-бюро и художественные мастерские имеют специально созданные условия освещения. Это значит, что работа ведется на высококачественных и тщательно откалиброванных мониторах в специально подготовленных помещениях с равномерным неярким освещением и без доступа прямых солнечных лучей. Стены и потолок таких помещений должны быть светло-серые. По тем же причинам следует избегать использования картин, репродукций или любых ярких декоративных элементов, которые могут быть источником искажений цветового восприятия.

Если по каким-то причинам невозможно обеспечить нейтральный фон во всем рабочем помещении, то цветные оттиски исследуют и оценивают в специально подготовленных кабинках, на ограниченных площадях которых намного проще обеспечить стандартные условия освещения. Получили распространение и портативные установки, позволяющие выполнить всю необходимую работу по оценке цветового пространства изображения или печатной страницы.

Сходными с хроматической адаптацией причинами объясняется еще один цветовой феномен — метамеризм, который проявляется в том, что два цвета выглядят одинаково при одних условиях освещения и представляются наблюдателю совершенно различными в иной световой среде. Сила проявления этого эффекта зависит от множества причин: технологии производства страницы, преобладающего цвета, типа композиции и пр. Известно, что в наибольшей степени метамеризму подвержены полноцветные полиграфические оттиски, в меньшей степени — фотографии. Изображения, отпечатанные на современных струйных принтерах пигментными чернилами, обладают самой большой устойчивостью в этом отношении.

1.4. Формирование цветных изображений

Описания цвета в терминах выбранного цветового пространства называют цветовой моделью. С помощью модели каждый хроматический тон и оттенок можно представить в виде совокупности чисел — цветовых координат, это позволяет обмениваться цветовой информацией между компьютерами, программами и периферийными устройствами.

1.4.1. Базовые принципы описания цвета

Все распространенные цветовые модели в зависимости от их особенностей и области применения можно разделить на три группы:

- *аппаратно-зависимые* — модели, используемые в технических средствах ввода-вывода графической информации;
- *аппаратно-независимые* — модели, не связанные с конкретным воспроизводящим устройством, они описывают цвет в абстрактных колориметрических терминах. Эти модели предложены международной комиссией по освещению CIE (Commission Internationale d'Eclairage), их широко применяют в теоретических исследованиях и системах управления цветом;
- *интуитивные* — модели, построенные на основе субъективного восприятия цвета человеком.

В технических средствах КГ используются два типа цветных объектов — самосветящиеся, излучающие объекты (экраны электронно-лучевых трубок, плазменные панели, матрицы светодиодов) и несамосветящиеся объекты, отра-

жающие или преломляющие падающий на них свет (бумажные оттиски, светофильтры и т. п.).

Для излучающих объектов используется аддитивное формирование оттенков, когда требуемый цвет формируется за счет смешения трех основных оттенков цветов. В этом случае целесообразно использовать модель, основанную на принципе сложения цветов. Самой известной моделью такого типа является модель RGB. Ее название образовано по первым буквам базовых цветовых координат Red (красный), Green (зеленый), Blue (синий).

Цвет несамосветящихся объектов получается в результате отражения светового потока некоторого внешнего источника или источников. Он формируется как сумма непоглощенных составляющих потока. Корректное описание этого цветового феномена дает так называемая субтрактивная цветовая модель CMY. Ее название образовано по первым буквам цветовых координат Cyan (голубой), Magenta (пурпурный), Yellow (желтый).

На рис. 2, а цветной вклейки показаны результаты смешения цветов в аддитивной модели для трех самосветящихся площадок чистых цветов (красного, зеленого и синего). На рис. 2, б представлены результаты смешения цветов в субтрактивной модели для трех несамосветящихся площадок чистых цветов (голубого, пурпурного и желтого).

Приведем основные правила сложения цветовых координат, которые описывается восемью простыми формулами:

- 1) желтый = красный + зеленый;
- 2) белый = красный + зеленый + синий;
- 3) пурпурный = красный + синий;
- 4) голубой = синий + зеленый;
- 5) красный = желтый + пурпурный;
- 6) зеленый = желтый + голубой;
- 7) черный = желтый + пурпурный + голубой;
- 8) синий = пурпурный + голубой.

Цвета одной модели являются дополнительными к цветам другой модели. Дополнительным называется цвет, который в сумме с данным дает чистый белый. Дополнительным для красного служит голубой, поскольку голубой получается смешением зеленого и синего. Дополнительным для зеленого является пурпурный (пурпурный = красный + синий), для синего — желтый (желтый = красный + зеленый).

Примеры формирования основных оттенков в субтрактивной модели показаны на рис. 3 цветной вклейки. Так, при освещении падающим белым светом в слое голубой краски из спектра белого цвета поглощается красная часть, затем из оставшегося света в слое пурпурной краски поглощается зеленая часть спектра. Отраженный от поверхности бумаги свет еще раз подвергается поглощению, и в результате мы видим синий цвет.

1.4.2. Модель RGB

Модель RGB (Red, Green, Blue — красный, зеленый, синий) — аддитивная аппаратно-зависимая модель, применяемая для описания цвета компьютерных мониторов, телевизионных экранов, дисплеев портативных цифровых устройств и телефонов.

Модель RGB — одна из первых цветовых моделей, ее появление датируется началом прошлого века — временем рождения первых устройств цветного телевидения. Можно привести множество аргументов в защиту ее состоятельности. Достаточно упомянуть классические опыты физиков XVIII в., которые доказали разложимость любого видимого света на три цветовые составляющие.

Цветовое пространство модели RGB можно представить в виде куба в декартовой системе координат (рис. 1.7). Каждый цвет в этом кубе задается точкой и определяется как сумма трех цветовых координат (основных цветов) красного, зеленого и синего. Главная диагональ куба с равными количествами каждого основного цвета представляет ахроматические (серые) цвета: черному цвету соответствует точка $(0, 0, 0)$, а белому — точка $(1, 1, 1)$. Нулевое значение соответствует отсутствию светимости цветовой координаты, единичное значение описывает ее максимальную интенсивность. Так, белый цвет дает смещение трех цветовых координат предельной силы.

Модель RGB — простая цветовая система, правила синтеза которой полностью согласуются с интуицией человека. Она отличается широким цветовым охватом и корректно описывает механизмы генерации цвета в приборах, излучающих свет. Эта модель имеет ряд недостатков, ограничивающих ее применение. Она не способна воспроизводить весь видимый человеком спектр. Сочетание цветов синий + зеленый дает только приближение для чистого голубого, а комбинация зеленый + красный продуцирует приблизительный аналог чистого желтого. На рис. 1.8 представлены зависимости интенсивностей цветовых координат R, G, B от длины волны света видимой части спектра. Как видно на приведенных графиках, некоторые из цветовых координат могут быть меньше нуля. Это означает, что в модели представлены не все цвета, которые способен воспринять глаз человека. Однако во многих случаях цветовая модель RGB дает



Рис. 1.7. Цветовой куб модели RGB

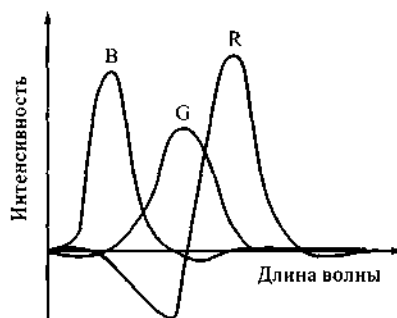


Рис. 1.8. Спектральные кривые для модели RGB

вполне удовлетворительные объяснения цветовых феноменов, что и объясняет ее широкое распространение в системах автоматизированного проектирования и графических пакетах.

1.4.3. Модели CMY и CMYK

При обсуждении модели RGB речь шла в основном об источниках света. Большинство окружающих нас объектов источниками не являются. Они не излучают, а поглощают и отражают падающий свет в разных пропорциях.

Как возникает цветность подобных объектов? Все пассивные объекты мы видим в отраженном цвете. Если яблоко имеет красный цвет, то это значит, что оно отражает длинные волны и поглощает короткие. Почему некоторый предмет окрашен в синий цвет? Это происходит потому, что он поглощает красную и зеленую составляющие и отражает только синюю. Как при отражении получается голубой цвет? Голубой представляет собой смешение синего и зеленого цветов. Следовательно, поверхность голубого цвета отражает синий и зеленый цвета, а значит, поглощает красную составляющую. Пурпурный краситель поглощает зеленый и отражает красный и синий. Если смешать голубой краситель и пурпурный, то цвет такой краски уже можно предсказать. Пурпурная составляющая поглотит зеленую, голубая — красную, остается только синяя компонента, поэтому результирующий цвет будет синим.

Для описания таких явлений используется цветовая модель, которая объясняет порождение цветов не как результат сложения, а как результат вычитания базовых цветов. Схемы, показанные на рис. 3 цветной вклейки, демонстрируют поведение световых волн различной длины на примере бумажного листа с нанесенными красителями. Смешивая попарно пурпурный, желтый и голубой красители, можно получить в отраженном свете оттенки основных цветов красного, зеленого и синего. Сочетания основных цветов позволяют синтезировать множество производных цветов, поэтому пурпурный, желтый и голубой могут быть приняты в качестве базиса субтрактивной (вычитательной) цветовой модели.

Модель CMY (Cyan, Magenta, Yellow — голубой, пурпурный, желтый) — аппаратно-ориентированная модель, используемая в полиграфии для субтрактивного формирования оттенков, основанного на поглощении слоем краски части падающего светового потока. Цветовые координаты модели CMY являются дополнительными к координатам RGB, т. е. сложение каждой дополнительной пары дает чистый белый. Поэтому модель CMY можно представить в виде куба, начало отсчета которого находится в точке с координатами RGB, равными (1, 1, 1). Эта точка соответствует белому цвету. Цветовой куб модели CMY показан на рис. 1.9.

Координаты одного цвета в моделях RGB и CMY — взаимно зависящие величины, соотношение между ними задается простой формулой

$$[R, G, B] = [1, 1, 1] - [C, M, Y],$$

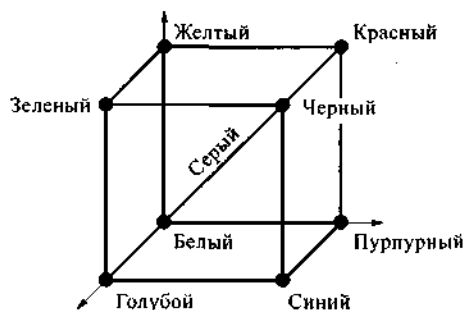


Рис. 1.9. Цветовой куб модели CMY

где единичный вектор $[1, 1, 1]$ в модели RGB — это представление белого цвета, а в модели CMY — черного.

Описанное соответствие координат справедливо только для нормализованных пространств, где значения RGB и CMY задаются в безразмерных величинах. В графических пакетах и CAD-системах эти значения могут рассчитываться по разным шкалам. Например, во многих программах растровой графики координаты CMY задаются в процентах, а RGB — в числах (от 0 до 255). В подобных случаях формулы пересчета принимают более сложный вид.

Таким образом, модель CMY представляет собой некоторое дополнение модели RGB, при нулевых значениях составляющих (отсутствии краски) образуется белый цвет. Смешение равных значений трех компонентов дает оттенки серого. При смешении максимальных значений трех составляющих (Cyan + Magenta + Yellow) должен получиться черный цвет.

Если нанести на белый лист бумаги красители пурпурного, желтого и голубого цветов, то они поглотят все три составляющие падающего света и такой лист должен выглядеть черным. В это теоретически правильное заключение практика вносит свои поправки. Существующие красители по своим химическим свойствам далеки от идеала и часто содержат примеси. Смешение таких красителей дает не черный цвет, а грязно-коричневый темного оттенка. Свой вклад вносит и бумага, поверхность и цвет которой никогда не бывают идеальными. Для повышения качества печати применяется специальный черный краситель, позволяющий получить ровный и глубокий черный цвет. Большинство современных репродуцирующих устройств (принтеров и типографских машин) печатают в четыре краски, и только самые дешевые струйные принтеры, сегодня практически вышедшие из употребления, используют только три краски.

Модель CMY с дополнительной черной составляющей называется моделью CMYK. Черный цвет (Black) представлен в названии последней буквой для того, чтобы не путать его в сокращениях с синим (Blue). Эта система служит теоретической основой цифровой печати. Во многих графических пакетах цветовые координаты рассматриваются как красители, которые наносятся на поверхность

бумаги, поэтому интенсивность каждой координаты измеряется в процентах от 0 (отсутствие краски) до 100 (максимальная интенсивность краски).

В системе RGB световые потоки суммируются и результирующие цвета получаются яркими. В субтрактивной системе световые потоки вычитаются, производя более темные и менее насыщенные оттенки. Этим объясняется тот эффект, когда яркие и живые цвета картинки, представленной на экране монитора, часто становятся выцветшими и тусклыми после вывода на печать.

Если учитывать только номенклатуру технических устройств, то окажется, что модель CMYK менее распространена, чем модель RGB. Только печатные машины и некоторые типы принтеров высокого класса используют эту модель напрямую. Традиционно в этом ряду упоминают еще и барабанные сканеры, но внутренние сенсоры подобных устройств работают в системе RGB, а считанная с оригинала информация потом преобразуется в CMYK программным или аппаратным способом.

1.4.4. Модели YUV и YIQ

YUV и YIQ — аппаратно-ориентированные модели, используемые в телевидении и позволяющие сократить передаваемую полосу частот за счет использования психофизиологических особенностей зрения в коротковолновой области. Это модифицированные системы RGB, приспособленные для нужд телевидения.

В моделях YUV и YIQ телевизионный сигнал кодируется посредством трех координат: Y — сигнал яркости, одинаковый во всех моделях, и два сигнала, определяющих цвет пикселя — (U и V) или (I и Q), являющиеся цветоразностными сигналами. При этом зеленый цвет (G), принятый за базовый, отсутствует в составе телевизионного сигнала и восстанавливается в телевизионном приемнике вместе с красным (R) и синим (B) цветом на основе принятых сигналов. Эти модели позволяют принимать сигналы не только цветного, но и черно-белого телевидения.

В Европе телевидение основывается на модели YUV и использует два способа кодирования сигналов модели — системы PAL и SECAM. Приведем простые правила расчета координат этой модели по значениям RGB:

- модель YUV системы SECAM

$$\begin{aligned}Y &= 0,299 \times R + 0,587 \times G + 0,114 \times B, \\U &= 1,5(B - Y), \\V &= -1,9(R - Y);\end{aligned}$$

- модель YUV системы PAL

$$\begin{aligned}Y &= 0,299 \times R + 0,587 \times G + 0,114 \times B, \\U &= 1,493(B - Y), \\V &= -1,877(R - Y).\end{aligned}$$

В США, Канаде и Японии для передачи цветного телевидения используется система NTSC. Модель описания цветов в этой системе называется YIQ.

Координаты модели YIQ в системе NTSC рассчитываются по следующим формулам:

$$\begin{aligned}Y &= 0,299 \times R + 0,587 \times G + 0,114 \times B; \\I &= -1,74(R - Y) - 0,27(B - Y); \\Q &= 1,48(R - Y) + 0,41(B - Y).\end{aligned}$$

Для полноцветного изображения компонент Y, называемый сигналом яркости (luminance), принимает значения между минимальным и максимальным уровнями сигналов R, G, B.

1.4.5. Модель CIE XYZ

В 1931 г. Международный комитет CIE утвердил несколько стандартных цветовых пространств, описывающих видимый спектр. При помощи этих моделей можно сравнивать между собой цветовые пространства отдельных наблюдателей и устройств на основе единых стандартов.

В предложенных Комитетом трехкомпонентных моделях для определения цвета используются три независимые цветовые координаты. Модели CIE обладают свойством аппаратной независимости и широким цветовым охватом. Диапазон цветов, которые можно определить в этих системах, не ограничивается изобразительными возможностями технического устройства или визуальным опытом определенного наблюдателя.

За основное цветовое пространство, разработанное в CIE, принято пространство XYZ. Оно построено на основе восприятия цвета некоторым стандартным наблюдателем. Это гипотетический персонаж с усредненными зрительными характеристиками, созданными по результатам многочисленных экспериментов с реальными наблюдателями. Приведем упрощенную схему испытаний.

Три монохроматических источника света направляются на белый экран таким образом, что их цвет смешивается. Испытуемые с пультом в руках располагаются перед экраном. В их распоряжении предоставляются три ручки, управляющие яркостью этих источников света. На другой стороне экрана воспроизводится эталонная точка некоторого заданного цвета. Задача испытуемых состоит в том, чтобы меняя яркости управляемых источников света, сделать цвет контрольной точки совпадающим с эталонной.

Интенсивность источников света принимает значения в диапазоне от -1 до 1. При интенсивности источника света, равном 1, лампа источника работала на полную мощность. Нулевое значение соответствовало выключению источника. В положении меньше нуля свет источника «вычитался» из результирующего. Это достигалось путем увеличения соответствующего компонента яркости эталонной точки. Результаты разных испытаний немного отличались друг от друга, но усредненные показатели на удивление точно совпали с пиками чувствительности разных типов колбочек. Они и были приняты за показатели гипотетического стандартного наблюдателя.

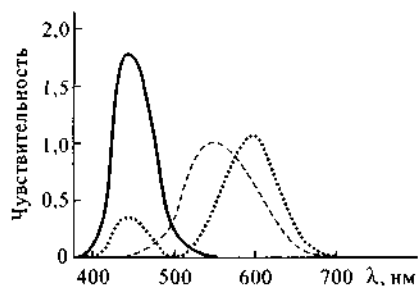


Рис. 1.10. Функции чувствительности системы CIE XYZ:

— — Z; ---- Y; X

По результатам проведенных испытаний были синтезированы три искусственные функции реакции глаза, зависящие от длины волны света и построенные таким образом, чтобы упростить ручные расчеты (рис. 1.10). Эти зависимости представляют собой математические абстракции, поскольку не существует естественных источников света с подобными характеристиками излучения. Отметим три особенности этих зависимостей. Во-первых, все они принимают только положительные значения; во-вторых, функция z равна нулю для большей части волн видимого

спектра; в-третьих, по значениям функции y можно рассчитать яркость измеряемого цвета.

На основе функций чувствительности по простым зависимостям рассчитываются абстрактные цвета CIE, обозначаемые X , Y , Z . Пространство, задаваемое этими цветами, достаточно мощное — оно включает в себя весь спектр видимых глазом цветов. Координаты цветности CIE (x , y , z) задаются следующими соотношениями:

$$x = \frac{X}{X + Y + Z}; \quad y = \frac{Y}{X + Y + Z}; \quad z = \frac{Z}{X + Y + Z}; \quad x + y + z = 1.$$

Работать с трехмерным цветовым пространством часто бывает неудобно, кроме того, во многих случаях целесообразно разделить хроматические и яркостные значения. При проецировании пространства XYZ на плоскость (x , y) получаем фигуру, которая называется хроматической диаграммой CIE (рис. 4 цветной вклейки). У этой диаграммы координаты x и y — относительные количества трех основных цветов XYZ, требуемых для составления нужного цвета. Яркость определяется величиной Y , а x и y задают цветность элемента цветового пространства. Таким образом, триада (x , y , Y) полностью определяет цвет. Обратное преобразование можно выполнить по формулам

$$X = x \frac{Y}{y}; \quad Y = Y; \quad Z = (1 - x - y) \frac{Y}{y}.$$

Хроматическая диаграмма весьма содержательна и заслуживает более подробного обсуждения. Каждый цветовой оттенок видимого спектра независимо от яркости изображается на диаграмме точкой. Метамерам — цветам, выглядящим одинаково, но имеющим разные описания, — соответствует одна точка диаграммы. Диаграмма наглядно показывает результаты сложения двух цветов. Для этого надо найти точки, соответствующие слагаемым цветам, и соединить

их прямой линией. Любая смесь выбранных компонентов лежит на этой прямой. Смесь трех цветов располагается внутри треугольника, построенного на хроматических координатах этих цветов.

Геометрическим местом чистых хроматических тонов служит линия, которая называется спектральной кривой (спектральным годографом). Все видимые цвета располагаются внутри фигуры, ограниченной спектральной кривой и линией пурпурных цветов. Чем ближе цвет к спектральной кривой, тем он чище и насыщеннее.

Точка, расположенная почти в самой середине диаграммы, описывает эталонный белый цвет. Если соединить эту точку с точкой на спектральной кривой, то получим геометрическое место точек, задающее цвета с одинаковым оттенком. Чем ближе цвет к спектральной кривой, тем большей чистотой он обладает. Хроматическая диаграмма дает точное описание колориметрического термина «чистота» — длина радиус-вектора с центром в белой точке. Светло-серые тона лежат в окрестности центра диаграммы, который представляет эталонный белый цвет.

В диапазоне между пурпурными и красными тонами зрение человека более чувствительно к небольшим изменениям цвета, чем в диапазоне между зелеными и желтыми тонами. Диаграмма отражает этот световой феномен. Ее часть, окрашенная преимущественно в желтые и зеленые тона, отличается более вытянутой формой. В белой, красной и пурпурной областях она более плотная.

1.4.6. Модели CIE Luv и CIE Lab

Несмотря на информативность и наглядность, хроматическая диаграмма CIE XYZ не может претендовать на статус универсальной цветовой модели. Она не обладает свойством перцепционной равномерности — расстояние между некоторыми ее точками непропорционально разности зрительного восприятия соответствующих цветов. Например, две точки в красной (нижней правой) части диаграммы, удаленные друг от друга на 0,1, воспринимаются наблюдателем как значительно различающиеся цвета. Тогда как две такие точки в зеленой (верхняя левая) части диаграммы представляют очень похожие цвета.

Предпринимались многочисленные попытки исправления этого недостатка, самые известные из них — модели CIE Luv и CIE Lab. Первая была предпринята в 1976 г. — модификация системы CIE XYZ, приспособленная к более равномерному восприятию цвета. Она строится на координатах u и v , которые рассчитываются по формулам

$$u = \frac{4x}{x + 15y + 3z}, \quad v = \frac{9x}{x + 15y + 3z}.$$

Хроматическая диаграмма модели CIE Luv, показанная на рис. 5 цветной вклейки, сохраняет структурные особенности диаграммы CIE XYZ: спектральную кривую, распределение ахроматических тонов и линию пурпурных цветов.

Модель CIE Lab (LAB) представляет собой нелинейное преобразование цветового пространства XYZ. Любой цвет в модели определяется значением яркости L (Lightness) и двумя хроматическими координатами a и b , которые рассчитываются на основе аргументов XYZ по громоздким иррациональным зависимостям. Проще говоря, L — это яркость, координата a принимает значения цветового круга от зеленого до красного, координата b — от голубого до желтого (рис. 6 цветной вклейки).

В природе не существует излучателей, которые могли бы воспроизвести диапазон цветовых значений хроматических координат a и b , поэтому модель применяется в теоретических исследованиях при обмене информацией о цвете и для синтеза цвета в компьютерных программах. Внутреннее описание цветов в Photoshop и в некоторых других программах обработки графики выполняется в системе Lab. Основным достоинством модели CIE Lab следует считать широкий цветовой диапазон. Ее хроматическая диаграмма позволяет определить цветовой охват любого устройства или модели, работающей по принципу сложения цветов. Внутри диаграммы Lab располагаются графики охватов любых моделей и устройств, основанных на принципе вычитания цветов: печатающих машин, принтеров и др. Система позволяет описать такие цвета и оттенки, которые недоступны ни для одной из прочих моделей. Например, яркий оранжевый, металлический блеск или даже ультранасыщенные тона, которые не способны дать природные красители и излучатели.

Для пользователей графических пакетов система Lab может показаться поначалу неудобной, поскольку она не согласуется с интуитивными представлениями о правилах синтеза цвета, существующих у каждого пользователя. Сильно упрощая ситуацию, ее можно сравнить с двухканальной моделью RGB с перевернутым каналом яркости. Смещение координат a и b дает новый цвет высокой интенсивности, для настройки его яркости следует воспользоваться третьей координатой L .

Обе модели (Lab и Luv) отличаются высокой перцептивной равномерностью. Их метрики различаются способами расчета, но в обеих системах близкие точки соответствуют сходным хроматическим оттенкам. Модель Lab доминирует в области промышленной печати и при генерации цвета в различных отраслях промышленности. Модель Luv преобладает в телевидении и киноиндустрии.

1.4.7. Интуитивные цветовые модели

Интуитивные цветовые модели базируются на предположении, что цвет может быть описан единственной монохроматической волной — цветовым тоном с заданием насыщенности и светлоты.

Набор из трех параметров — цветовой тон, насыщенность и светлота, или интенсивность (освещенность), — наглядно показывает, что видимый цвет трехмерен. Эти параметры можно интерпретировать как три координаты, с помощью которых можно графически представлять положение видимого цвета в

цветовом пространстве. В начале XX в. художник Мансел (A.H. Munsell), создатель цветковых таблиц, впервые дал интуитивное описание трехмерного цветового пространства. Сегодня на базе идей Мансела построено достаточно много цветковых пространств различных типов. Самой распространенной из них является модель HSB (рис. 7 цветной вклейки). Она была разработана как попытка преодолеть аппаратную зависимость модели RGB. В модели HSB все цвета определяются тремя координатами: оттенком (Hue), насыщенностью (Saturation) и яркостью (Brightness). Название модели образовано по первым буквам английских названий цветковых координат.

Цветовым тоном, или оттенком (Hue), называется спектрально-чистый цвет определенной длины волны, например чистый красный или чистый зеленый. Цветовой тон — объективная характеристика, поскольку ее можно измерить по длинам преобладающих в световом пучке волн.

Яркость характеризует интенсивность, энергию цвета. Изменение яркости можно представить как смешение чистого тона и черного цвета. Большое содержание черного делает цвет затененным, неинтенсивным. С уменьшением процента черного освещенность увеличивается. Солнечный луч обладает высокой яркостью света, свечение, исходящее от светлячка — очень низкой яркостью. Черный цвет имеет нулевую яркость, а белый — предельную.

Насыщенность (Saturation) описывает чистоту цвета. Один и тот же тон может быть тусклым или насыщенным. Изменение насыщенности можно представить как разбавление чистого цвета белым или серым. Чем больше содержание белого, тем более блеклым становится цвет. Все цвета естественного происхождения имеют низкую насыщенность, поэтому чистые тона выглядят слишком яркими, ненатуральными.

Кроме модели HSB существует несколько моделей, в которых яркостная и цветовая характеристики рассматриваются отдельно, например HSI, HLS. Во всех этих моделях цвет задается не как смешение трех цветов, а по значениям цветового тона, насыщенности и интенсивности. В модели HSI используется тон (Hue), насыщенность (Saturation) и интенсивность (Intensity), в модели HLS — тон (Hue), насыщенность (Saturation) и светлота (Lightness).

Для описания модели HSB удобно воспользоваться геометрической аналогией. Пусть цвета видимого спектра располагаются по кругу, как цифры на циферблате часов. Каждому оттенку соответствует точка на окружности. Чтобы указать положение спектрального цвета, достаточно задать угол поворота радиуса-вектора. В большинстве графических программ принято начинать отсчет от красного цвета и располагать основные и дополнительные цвета с приращением в 60° (см. рис. 7 цветной вклейки). Величина насыщенности описывается как длина радиуса-вектора. Чем менее насыщенным является цвет, тем ближе к центру окружности располагается представляющая его точка. Центр круга соответствует черному цвету. Обычно насыщенность измеряется в процентах: минимальная насыщенность равна 0, максимальная — 100. Чтобы учесть в модели яркость, надо добавить третью координату. Все цветовое пространство системы HSB можно

представить в виде стопки цветовых кругов, каждый из которых соответствует своему значению яркости. Яркость в большинстве графических программ изменяют в процентах в диапазоне от 0 (минимальная) до 100 (максимальная).

Система HSB очень удобна для пользователя. В ней можно синтезировать новые цвета и получать различные варианты заданного цвета, опираясь на интуицию. Например, известно, что чистый синий цвет лежит на цветовом круге под углом 240° . Если требуется сместить тон в сторону пурпурного оттенка, то для этого достаточно увеличить угол поворота. Цвет кажется слишком насыщенным? Решение известно. Надо сместить точку в радиальном направлении ближе к центру. Велика яркость? Уменьшаем соответствующую координату. Подобную стратегию синтеза цвета невозможно реализовать в системе RGB, поскольку трудно предвидеть последствия даже небольших изменений цветовых координат. Еще одним несомненным достоинством системы HSB является ее независимость от аппаратуры. Примерно такую оценку могли бы дать этой системе пользователи и разработчики компьютерных программ.

Мнения физиков и инженеров-оптиков по поводу этой системы, видимо, будут отличаться от приведенных оценок. Система HSB является абстрактной. Это значит, что нет таких устройств, которые синтезируют цвет в этой системе. Не существует и прямой процедуры измерения цветового тона и насыщенности. В любом методе ввода информации о цвете сначала измеряются красная, синяя и зеленая составляющие, которые потом пересчитываются в координаты HSB. Поскольку при вводе и выводе цвета система HSB привязана к системе RGB, то ее аппаратная независимость является во многом умозрительным тезисом и не имеет большого практического значения.

1.4.8. Цветовой круг

Цветовой круг — описание системы HSB, которое используется как мнемоническая фигура для упрощения ориентации в цветовом пространстве. У полиграфистов и оптиков он играет роль своеобразного навигационного прибора, без которого часто бывает сложно принять верное решение о способе синтеза искомого цвета.

На цветовом круге (рис. 8 цветной вклейки) на равном расстоянии друг от друга расположены первичные и вторичные цвета. Каждый вторичный (первичный) цвет находится между двумя первичными (вторичными). Сложение двух основных цветов дает дополнительный цвет, расположенный между ними. Например, смешивая зеленый и синий, получаем голубой. При смешении двух дополнительных цветов получаем основной цвет, лежащий между ними. Так, смесь желтого и пурпурного образует красный.

Пары цветов, расположенные на круге под углом 180° , называются комплементарными, или дополнительными. Таковыми являются зеленый и пурпурный, голубой и красный, синий и желтый. Добавление любой краски цветового круга компенсирует дополнительную краску, как бы разбавляет ее в результирующем цвете. Например, чтобы изменить цветовое соотношение в сторону зеленых то-

нов, следует понизить содержание пурпурного цвета, который является дополнительным к зеленому.

Приведем несколько правил сложения цветов. Они, конечно, объясняются основными физическими закономерностями, но самый простой способ их получения — это использование мнемоники цветового круга:

- наложение красного и зеленого с максимальной интенсивностью дает чистый желтый цвет. Уменьшение интенсивности красного смещает результирующий в сторону зеленых оттенков, а снижение интенсивности зеленого делает цвет оранжевым;

- смешение синего и красного в максимальной пропорции дает фиолетовый цвет. Уменьшение доли синего влечет сдвиг в область розового цвета, а уменьшение красного сдвигает цвет в сторону пурпурного;

- зеленый и синий цвета образуют голубой. Существует около 65 000 различных оттенков голубого, которые можно синтезировать, смешивая в разных пропорциях данные цветовые координаты;

- наложение голубой и пурпурной красок максимальной плотности дает глубокий синий цвет;

- пурпурный и желтый красители образуют красный цвет. Чем выше плотность составляющих, тем выше его яркость. Уменьшение интенсивности пурпурного придает цвету оранжевый оттенок, снижение доли желтой составляющей дает розовый цвет;

- желтый и голубой дают ярко-зеленый цвет. Уменьшение доли желтого порождает изумрудный, а снижение доли голубого — салатовый.

1.4.9. Стандартные источники света CIE

Восприятие цвета наблюдателем зависит не только от источника света, но и условий освещения. Точное определение характеристик источника света является важной частью описания цвета во многих приложениях. Стандарты источников света CIE предлагают универсальную систему предопределенных спектральных данных для нескольких широко применяемых типов источников света.

Стандартные источники света CIE, утвержденные в 1931 г., обозначены буквами A, B и C:

- источник цвета типа A представляет собой лампу накаливания с цветовой температурой примерно 2856 K;

- источник цвета типа B — прямой солнечный свет с цветовой температурой примерно 4874 K;

- источник цвета типа C — не прямой солнечный свет с цветовой температурой примерно 6774 K.

Впоследствии в стандарт CIE к этому набору типов были добавлены типы D, E и F. Типу D соответствуют различные условия дневного освещения с опре-

деленной цветовой температурой. Например, источники D50 и D65 — стандартные источники, широко применяемые для освещения специальных кабин для просмотра полиграфических оттисков (индексы «50» и «65» соответствуют цветовой температуре 5000 и 6500 К соответственно).

При проведении цветовых вычислений учитываются также спектральные данные источников света. Хотя источники света, по сути, являются эмиссионными (излучающими) объектами, их спектральные данные практически ничем не отличаются от спектральных данных отражающих цветных объектов. Соотношение определенных цветов в различных типах источников света можно выяснить, исследовав относительное распределение мощности световых потоков с различной длиной волны.

Таким образом, описания цветов, составленные по трем координатам, сильно зависят от стандартных цветовых систем CIE и от источников света. В свою очередь, спектральное описание цвета эту дополнительную информацию напрямую не использует. Тем не менее стандарты CIE играют важную роль в процессе преобразования цветовой информации из трехкоординатной формы представления данных в спектральную форму. Методы описания цвета можно разделить на два типа:

- *спектральные данные* фактически описывают свойства поверхности объекта, показывая, как эта поверхность воздействует на свет (отражает его, пропускает или излучает). На эти поверхностные свойства не влияют условия внешней среды, такие как освещение, индивидуальность восприятия каждого из зрителей и различия в методах интерпретации цвета;

- *трехкоординатные данные* задают цвет при помощи трех координат. Эти характеристики описывают, как представляется цвет зрителю или сенсорному устройству или как цвет будет воспроизводиться на каком-либо устройстве, например на мониторе или принтере. Координаты систем, например CIE XYZ и CIE Lab, задают положение цвета в цветовом пространстве. Системы воспроизведения цвета (например, RGB и CMY) описывают цвет посредством трех величин, которые порождают его при смешивании.

Как формат для спецификации и передачи информации о цвете спектральные данные имеют ряд определенных преимуществ перед трехкоординатными форматами. Прежде всего, спектральные данные являются единственным, объективным описанием цветного объекта. И наоборот, системы RGB и CMYK дают такое описание, которое зависит от внешних условий: типа устройства воспроизведения, условий освещения и др.

1.4.10. Формирование цветов на экране монитора

Давно миновали дни, когда оператор общался с вычислительной системой посредством телетайпа или колоды перфокарт. В настоящее время монитор —

неотъемлемая часть любого домашнего или профессионального компьютера. Современный пользователь часто не подозревает, что существуют отличные от дисплея интерфейсы, позволяющие общаться с машиной. Векторные мониторы, столь популярные в недалеком прошлом, используются в IT-областях предельно узкой специализации.

Сейчас самым распространенным устройством отображения информации является цветной растровый монитор. Работу растровых мониторов, независимо от того, выполнены ли они на классических электронно-лучевых трубках или на современных жидкокристаллических панелях, адекватно описывает модель RGB. Количество цветовых оттенков, доступных растровому монитору, ограничено и зависит от технических характеристик всех устройств, образующих видеотракт, главным образом от видеокарты (видеоадаптера).

Самыми простыми устройствами отображения являются монохромные мониторы, позволяющие изображать только два цвета. Наиболее развитые графические системы способны отображать 16,7 и более миллионов цветов. Максимальное число цветов, одновременно отображаемых на экране, определяется числом разрядов, выделенных для каждого пиксела в видеопамяти. В развитых графических системах на каждый пиксел отводится 24 (или более) двоичных разряда для описания цвета: восемь — для красной составляющей, восемь — для зеленой и восемь — для синей.

Используя восемь двоичных разрядов, можно представить $2^8 = 256$ различных градаций яркости. Три независимых цветовых канала способны создать в общей сложности $256 \times 256 \times 256 = 16\,777\,215$ цветовых градаций. Это означает, что видеоадаптер может отобразить более 16,7 млн цветов. Смешивая разные интенсивности красной, зеленой и синей составляющих, можно получить почти любой цвет видимой части спектра. В таких видеосистемах информация о цвете каждого пиксела монитора представляется в виде тройки значений R, G и B. Эти данные видеопамяти передаются непосредственно на монитор или сначала в таблицу цветности, а затем на монитор.

В дисплеях с таблицей цветности (Look up table, LUT) значение считанного пиксела не сразу передается на цифроаналоговое преобразование, а используется в качестве адреса. По этому адресу из таблицы выбираются значения яркостей по R, G, B, которые определяют действительный цвет точки монитора. Эта схема формирования изображения обладает несколькими преимуществами по сравнению с прямым выводом на экран. Так, обработкой всего лишь нескольких байтов таблицы цветности можно одновременно изменить цвет у всех точек изображения с одинаковым кодом пиксела, не затрагивая содержимого видеопамяти. С помощью таблицы цветности реализуются не только различные визуальные эффекты, но может быть осуществлена необходимая фильтрация изображения, получены любые битовые срезы, выполнена гамма-коррекция и коррекция цвета без обращения к видеопамяти.

При работе с цветом сознание человека оперирует терминами, отличными от координат RGB. Лишь немногие профессионалы способны по интенсивно-

стям отдельных каналов предсказать результирующий цвет. Еще меньше людей могут успешно решить обратную задачу — синтезировать искомый цвет в системе RGB, поскольку не существует простого алгоритма для определения координат RGB по тону, яркости и насыщенности оттенка. В различного рода графических редакторах эта задача обычно решается с помощью интерактивного выбора из палитры цветов и формированием цветов в палитре путем подбора значений RGB до получения требуемого визуального результата. Более удобно в этом случае использовать модели HSB или HLS, позволяющие непосредственно задать требуемый оттенок. Однако при занесении данных в таблицу цветности или в видеопамять полноцветных дисплеев требуется перевод значений цветового тона, яркости и насыщенности в систему RGB.

1.4.11. Интерполяция цветов

Интерполяция цветов, или псевдотонирование (half-toning), — способ генерации цветов, которые невозможно воспроизвести средствами графической системы непосредственно.

В современных графических приложениях псевдотонирование используется очень часто: увеличение цветового охвата ограниченных палитр, создание реалистических эффектов в играх, имитация художественных техник в растровой графике, моделирование цветовых градиентов, заполнение многоугольников методом Гуро и др. Существует много вариантов псевдотонирования, но все они основаны на одном принципе — замене пикселей с цветами, отсутствующими в палитре, конфигурациями точек, окрашенными в доступные цвета. Интерполяция цвета основывается на том, что глаз человека не обладает высокой разрешающей способностью, он усредняет и смешивает оттенки соседних пикселей, воспринимая совокупный цвет области, как некий новый цвет. Так, зеленый цвет можно получить чередованием желтых и синих точек, а оттенки серого можно имитировать смешением черных и белых точек.

Обсудим более подробно технику псевдотонирования на примере черно-белых изображений. Пусть имеется растровое полутоновое изображение, точки которого принимают произвольные значения в диапазоне от 0 до 1, где нулевая яркость соответствует черному цвету, а единичная описывает белый цвет. Рассмотрим возможные алгоритмы приведения такого изображения к монохромному, яркость точек которого может принимать только два значения.

Самый простой алгоритм, решающий поставленную задачу, — пороговый. Выбирается критическое значение яркости (обычно 0,5), которое является основанием для объявления полутоновых пикселей белыми или черными. Все точки, яркость которых выше порога, становятся белыми, остальные точки превращаются в черные. Этот простой алгоритм обычно дает не очень качественные результаты, поскольку заметно обедняет изображение.

Более совершенные алгоритмы псевдотонирования распределяют черные и белые пиксели в полученном изображении так, чтобы на каждом участке изображения концентрация белых пикселей была пропорциональна яркости этого участка в исходном изображении. Один из таких алгоритмов — упорядоченное псевдотонирование. В этом алгоритме растровое изображение разбивается на небольшие блоки одинакового размера (например, 3×3). Затем в каждом блоке находится средняя яркость изображения. В соответствии с этим значением выбирается количество белых пикселей в соответствующем блоке получаемого монохромного изображения. Обычно эти белые пиксели упорядочиваются в соответствии с некоторым регулярным шаблоном.

Есть другие алгоритмы достижения нужной концентрации белых пикселей в получаемом монохромном изображении. Например, существует класс алгоритмов, в которых монохромное изображение достигают в два этапа. Сначала к изображению добавляется случайный шум необходимой амплитуды, а затем применяется пороговое усечение яркостей. Иногда такой способ псевдотонирования называют диттерингом (dithering).

Псевдотонирование в некоторых случаях может вносить в изображение визуальные дефекты. Группы независимых пикселей в совокупности могут составлять паразитные образы (артефакты), отсутствующие в оригинале. Более надежной является так называемое диффузное псевдотонирование, которое обладает более высокой устойчивостью к появлению артефактов. В алгоритмах этого класса просматривается каждый пиксел изображения, а его новый цвет выбирается так, чтобы отличие нового цвета от исходного было минимальным. Затем вычисляется вносимая ошибка, т. е. разность между новым цветом и старым, и эта ошибка распределяется между соседними пикселями, немного изменяя их оттенки. Например, если новый цвет пикселя содержит меньше красного и зеленого, чем старый, то диффузное псевдотонирование добавит немного красного и зеленого окружающим пикселям. Такой адаптивный подход устраняет артефакты и, как правило, обеспечивает хорошие результаты.

Разновидность псевдотонирования, называемая автотипией, широко применяется в полиграфии для получения цветных и полутоновых оттисков.

1.5. Измерение цвета и калибровка технических средств

Цифровая обработка цветного изображения на современных вычислительных системах во многих случаях представляет собой сложный политехнический процесс, отдельные этапы которого выполняются на компьютерном оборудовании с разными характеристиками и принципами действия. В частности, этому описанию полностью отвечает современная технология подготовки полноцветных изданий, где невозможно получить продукт высокого качества без специальных мероприятий по сохранению цвета. Автоматизированный программно-технический комплекс, обеспечивающий качество цвета на всех этапах жиз-

ненного цикла публикации, называется системой управления цветом (Color Management System — CMS).

Проблемы рассогласования цветовых параметров во многом объясняются объективными техническими причинами:

- при вводе со сканера используется RGB — цветовое пространство конкретного сканера;
- при обработке графических изображений они воспроизводятся на экране монитора с его RGB цветовым пространством;
- документирование полученных изображений осуществляется с помощью принтера, использующего CMYK цветовое пространство.

Каждый сканер и монитор имеют собственные цветовые пространства RGB и каждый принтер — собственное CMYK-пространство, причем все эти цветовые пространства различны и могут иметь очень большие отличия. В такой ситуации дизайнерам при настройке цветов и выводе пробных отпечатков в настольных графических системах приходится иметь дело с огромным числом неоднозначных решений и работать практически наугад. Сканированные цвета выглядят на мониторе не так, как в оригинале; экранные цвета не совпадают с пробными отпечатками; цвета, сохраненные в файлах изображений, выводятся в различных устройствах на экран и на печать по-разному (в дизайн-студии, сервис-бюро, в типографии). Решить эту проблему помогают системы управления цветом CMS, которые действуют на уровне настольных графических систем, но также оказывают влияние и на решения, принимаемые на различных этапах технологической цепочки.

Разработки систем управления цветом ведутся уже не один десяток лет. Если первые версии систем управления цветом не принимались всерьез даже их создателями, то сейчас ни один квалифицированный компьютерный дизайнер не может рассчитывать на успех без знания основных принципов этой цифровой технологии.

1.5.1. Системы управления цветом

В основе современных систем управления цветом лежат две базовые концепции: калибровка и профилирование. Калибровка — изменение поведения устройства в соответствии с некоторыми признанными стандартами. Профилирование заключается в измерении характеристик устройства отображения и сохранении полученных данных. Это, по сути дела, регистрация фактического положения дел, настройки устройства при этом не требуется. Калибровка и профилирование взаимосвязаны.

Профилирование и калибровка были известны задолго до появления компьютерных систем управления цветом. Они использовались для настройки высококачественных барабанных сканеров, печатающих устройств, предназначенных для получения пробных цветных оттисков и пр. Только с появлением систем управления цветом были разработаны и приняты общие стандарты,

дающие единый фундамент процедурам настройки и измерения цвета. Беспорядку со специализированными фирменными форматами был положен конец в 1995 г., когда фирма Apple объявила о создании встроенной в операционную среду системы управления цветом ColorSync 2. Фирма предложила новый стандарт записи профайлов и сделала его открытым. Формат оказался удачным и был стандартизован международным консорциумом по свету ICC (International Color Consortium). Разработка принята сообществом разработчиков программного и технического обеспечения и в настоящее время все системы управления цветом основываются на профилях ICC.

Профилирование и калибровка технических устройств оказываются неработоспособными без надлежащей системной организации. Программно-аппаратная среда, объединяющая средства управления цветом в КГ, называется системой управления цветом и ее часто обозначают аббревиатурой CMS (Color Management System). Существует несколько таких систем, среди которых можно выделить двух явных лидеров: на платформе Windows — Image Color Management (ICM); на платформе Macintosh — ColorSync.

Все системы CMS (рис. 1.11) включают в себя три основных составляющих:

- базовое цветовое пространство системы. Аппаратно-независимый способ описания цветов, свободный от ограничений и особенностей классов и типов технических устройств. Это своего рода общий знаменатель, к которому приводятся цветовые пространства отдельных технических устройств, входящих в технологическую цепочку подготовки цветных публикаций. В последних CMS эти функции выполняют CIE Lab или CIE XYZ. Базовое пространство — важная теоретическая составляющая любой системы управления цветом. Для рядового пользователя она не имеет прикладного значения, поскольку является полностью закрытой;



Рис. 1.11. Структура системы управления цветом

- механизм согласования цветов. Совокупность программных средств, выполняющих преобразования между различными аппаратно-зависимыми цветовыми моделями. Иногда эту важную часть системы управления цветом называют методом согласования цветов и обозначают аббревиатурой CMM (Color Matching Method);

- профили устройств (профайлы). Профилем называется файл, который хранит информацию о цветовом охвате устройства и используемой в нем цветовой модели. Если известны профили всех устройств, связанных в технологическую цепочку, то появляется возможность для согласования их цветовых охватов. Базовые принципы (но не реализация) такого согласования очень просты. Надо подавить все оттенки, которые не могут быть воспроизведены хотя бы одним устройством технологической цепочки. Все реализуемые цвета должны быть синтезированы так, чтобы обеспечить наивысшее качество их воспроизведения в данной технологической среде.

1.5.2. Профили ICC

Профиль — документ, описывающий свойства прибора при передаче или отображении цвета. Правила формирования профилей описываются в стандарте международного консорциума по свету ICC. Это открытый документ, который доступен в сети по адресу www.color.org. Стандарт адресован разработчикам, это сложный технический текст на английском языке, трудный для восприятия неспециалистами.

Описание профиля устройства начинается с заголовка. В нем указывается тип устройства отображения (сканер, монитор, принтер и пр.), рекомендуемый модуль управления цветом, вид входного и выходного цветового пространства и другая техническая информация, необходимая для описания свойств устройства цветовоспроизведения. Основным объемом профайла занимают таблицы пересчета координат цветовых пространств. Кроме того, сюда входит разнообразная служебная информация, которая объясняет системе управления цветом правила обращения с данным устройством.

На всех стадиях разработки цветных публикаций используются профили:

- профиль сканера описывает то цветовое пространство, в терминах которого прибор описывает RGB-данные, полученные от светочувствительных элементов. Профиль сканера может быть внедрен в каждое оцифрованное изображение или применен к нему после открытия изображения в графическом редакторе;

- профиль монитора характеризует свойства RGB-пространства калиброванного монитора. В любой технологии обработки цифровой графики калиброванный и профилированный монитор играет ключевую роль. Без надлежащей подготовки монитора теряют всякий смысл разговоры об управлении цветом и получении достоверных результатов;

- выходной профиль описывает параметры цветовоспроизведения окончательного печатающего устройства. Обычно принтеры и печатающие станки работа-

ют в системе CMYK, но встречаются устройства, для описания которых применяется система RGB. Использование выходных профилей позволяет получить оттиски высокого качества на печатном оборудовании разного типа.

Рассмотренные профили описывают поведение реальных физических устройств: сканеров, мониторов и принтеров. Стандартами ICC предусмотрено использование профилей для связок устройств, работающих совместно, и профили абстрактных приборов, описывающие различные цветовые пространства.

Профили получили полную поддержку производителей компьютерной периферии. В настоящее время трудно рассчитывать на коммерческий успех продукта, не оснащенного точным описанием его цветовых свойств. Некоторые высококачественные сканеры и принтеры снабжаются несколькими профилями, предназначенными на работу в различных условиях. Например, для точного цветовоспроизведения на матовой и глянцевой бумаге принтер может использовать разные профили.

Предустановленные фабричные профили часто имеют ограниченную применимость в реальных условиях. Среды, в которых производитель выполнял калибровку прибора и измерение его цветопередачи, могут радикально отличаться от действительных эксплуатационных условий. Если для мониторов и сканеров иногда удается стандартизировать параметры рабочей среды, то для устройств печати это сделать намного сложнее. Так, одним из многих случайных негативных факторов для струйных принтеров является качество чернил, которые отличаются ограниченным сроком годности и высокой долей поддельных картриджей низкого качества.

Чтобы обеспечить надежную цветопередачу в процессе подготовки цветных изданий, часто приходится использовать заказные профили — описания, которые создаются пользователем и учитывают все особенности реальной производственной ситуации. С технической точки зрения — это несложная процедура, но она требует определенных технических ресурсов и денежных вложений.

Если ограничиться главными составляющими этой технологии, то для построения качественного профиля устройства требуются специальные измерительные приборы (колориметры и спектрофотометры) и программное обеспечение, предназначенное для обработки результатов измерения.

Рынок программного обеспечения предлагает множество различных программ-профайлеров. С одной стороны, это профессиональные пакеты стоимостью несколько тысяч долларов, например MonacoProfiler 3.2 или CompassProfile. С другой стороны, существует бесплатное программное обеспечение, которое входит в комплект поставки некоторых марок компьютерной периферии. Примером такой программы является пакет Colorficient, разработанный фирмой E-Color.

1.5.3. Инструменты для измерения цвета

Важнейшим условием создания качественного заказного профиля устройства отображения цвета является точное измерение цветовых параметров. В по-

следние несколько лет в отрасли, занимающейся разработкой и выпуском измерительных приборов, наблюдаются тенденции, сходные с направлением развития всей компьютерной периферии. Во-первых, это снижение стоимости приборов, во-вторых, упрощение правил эксплуатации. Если раньше цена качественного спектрофотометра могла доходить до нескольких сотен тысяч долларов, а для работы с ним требовался дипломированный специалист в области оптики, то сейчас эти устройства стали общедоступными по цене и эксплуатационным нормам.

Все измерительное оборудование можно разбить на три группы: денситометры, колориметры и спектрофотометры. Самыми простыми по техническому устройству и правилам эксплуатации являются денситометры. Они не дают полной информации о цветовых параметрах оригинала: обычно эти приборы считывают значение оптической плотности в точке или небольшой области, ограниченной несколькими пикселями. Некоторые модели высокого уровня могут выполнять простую статистическую обработку массива данных, а также контролировать баланс серого, ахроматичность и некоторые другие характеристики оригинала.

Колориметры представляют более развитый класс измерительных приборов, поскольку они способны снимать информацию о значениях красного, зеленого и синего цветов в каждой пробной области. Некоторые аппаратные калибраторы мониторов представляют собой колориметры, снабженные специальным программным обеспечением и приспособленные для работы в специфических условиях компьютерного дисплея.

Спектрофотометры являются самыми сложными, точными и дорогостоящими приборами измерения цвета. Это приборы, предназначенные для регистрации интенсивности излученных, отраженных и поглощенных световых потоков. Регистрируемую часть спектра они разбивают на несколько коротких поддиапазонов (обычно от 16 до 256) и измеряют интенсивность излучения в каждом из них. Спектрофотометры дают самую подробную информацию о цветовых свойствах изображения, поскольку они оперируют со спектральным составом цвета. Данные о распределении световых волн различной длины могут быть преобразованы в любую другую удобную форму, например цветовое пространство.

До последнего времени спектрофотометры были сложными и дорогими приборами; работу с ними могли позволить себе только крупные издательства, фирмы или препресс-бюро. Сейчас ситуация меняется, на рынке появились сравнительно недорогие приборы, доступные небольшим творческим коллективам и рекламным фирмам. Такие устройства, например, выпускают фирмы X-Rite и DataColor.

Профилирование любого технического устройства, репродуцирующего цвет, выполняется по единой схеме — измерение отклика прибора в ответ на некоторое эталонное воздействие. Так, аппаратный калибратор монитора посылает на вход видеосистемы компьютера известный сигнал RGB, а затем использует колориметр для измерения показанных монитором значений цвета. Расхождение между эталонным и зарегистрированным сигналом является характеристикой

монитора, которая должна быть записана в профиль данного устройства. По такой же схеме выполняется профилирование печатающих устройств. Выводится на печать эталонная картинка, которая содержит образцы цвета с заранее известными координатами. Отпечатанная версия обрабатывается спектрофотометром и сравниваются исходные и полученные данные. Все обнаруженные различия записываются в профиль печатающего устройства.

1.5.4. Создание профиля монитора

Создание профиля монитора можно сравнить с созданием технического паспорта, который описывает особенности воспроизведения цвета этого устройства. Калибровка монитора — приведение технических характеристик прибора в соответствие с некоторым корпоративным или общепромышленным стандартом. Процедура создания профайла монитора всегда тесно связана с его калибровкой. Следует сначала привести характеристики монитора к некоторому стандартному состоянию, и только после этого занести в профиль состояние калиброванного монитора. Практика показывает, что использование стандартного профайла, который поставляется вместе с монитором фирмой-производителем, дает худшие результаты, чем использование нового заказного профиля. Тем более что профилирование монитора — сравнительно простая процедура, не требующая высокой квалификации и значительных капитальных вложений.

Монитор и принтер — принципиально различные устройства по физическому принципу действия. Никакая калибровка не в состоянии сделать эквивалентными цветовые охваты этих приборов. Речь может идти только о более точном отображении цвета RGB на мониторе и цвета CMYK на принтере. Только при этих условиях можно воспользоваться теми преимуществами, которые дает профессиональный графический редактор при работе с цветными изображениями.

Изменить цветопередачу монитора можно только воздействуя на доступные для изменения параметры этого устройства. Обычно к числу таковых относятся яркость, контрастность, сведение (для экранов на ЭЛТ), цветовая температура и гамма-функция. При помощи программных или аппаратных средств настройки мониторов создается специальная таблица преобразований, которая называется LUT (Look up Table). Она содержит сведения о том, как следует изменить входной сигнал монитора, чтобы обеспечить на нем искомые изменения. Таблица загружается в специальный раздел памяти графического адаптера и выполняет функции своеобразного фильтра. Графический сигнал, подаваемый на монитор, пропускается через этот раздел и преобразуется в соответствии с данными, записанными в таблицу. Описанная схема позволяет сделать вывод, что способность графической подсистемы компьютера к калибровке в первую очередь зависит от технических характеристик монитора.

Существует несколько способов калибровки мониторов. Самыми распространенными являются аппаратная и визуальная технологии калибровки. Аппаратная калибровка применяется в тех случаях, когда к качеству цветопередачи

предъявляются повышенные требования. Монитор снабжается специальной внешней системой, основной частью которой является колориметр. Он измеряет выходной сигнал монитора на эталонный сигнал, который посылается калибратором на вход видеоадаптера. Специальное программное обеспечение, поставляемое вместе с калибратором, обрабатывает разницу между откликом монитора и эталоном и вносит в корректирующую таблицу все необходимые изменения.

Визуальная калибровка не требует дополнительного технического оснащения; она может быть выполнена при помощи специального программного обеспечения, на основе зрительной оценки оператора. На рынке программных продуктов можно найти много программных утилит, предназначенных для калибровки и профилирования мониторов. На платформе Windows самой известной программой такого класса является Adobe Gamma. Визуальная калибровка не предъявляет никаких особых требований, она может быть применена к любой связке монитор — видеокарта.

Калибровка — не разовая акция, а регулярное мероприятие. Мониторы на электронно-лучевых трубках отличаются заметной нестабильностью технических характеристик. Цветовые характеристики этих устройств могут значительно меняться даже в течение одного сеанса работы, колеблются и внешние условия их функционирования. Со временем выгорают зерна люминофора и яркость монитора снижается. Особенно чувствительны к выгоранию точки синего цвета. Это значит, что со временем белый цвет монитора получит легкий коричневый оттенок и понизится общая яркость изображения. Оператор скорее всего не заметит происходящих изменений, поскольку его система цветовосприятия просто приспособится к ним, но эти тренды могут иметь решающие последствия для всей системы цветовоспроизведения.

1.5.5. Создание профиля сканера

В принципе возможно успешное управление цветом при работе с некалиброванным сканером или цифровой камерой, но для этого требуется выполнение нескольких обязательных условий. Тщательно откалиброванный монитор профессионального уровня, корректное цветовосприятие оператора и стабильные технические характеристики всех устройств, входящих в технологическую цепочку, — минимальные требования, без выполнения которых всякие разговоры о точном цвете являются утопией.

Схема создания профиля сканера, или цифровой камеры, довольно проста. Для этого требуется оцифровать печатный образец и сравнить результаты с эталонными данными, которые получены вычислениями или в результате измерений точными приборами. В качестве печатного образца обычно используют специальные цветовые мишени IT8, признанные стандартами международной организацией по цвету. Это отпечатанная коллекция эталонных цветов, числовые координаты которых получили точную предварительную оценку. Существ-

ует три основных вида мишеней: IT8.7/1 — для прозрачных оригиналов, IT8.7/12 — для непрозрачных оригиналов и IT8.7/3 — для калибровки печатных устройств. В разделе, посвященном сканированию, приведены примеры таких мишеней производства фирм Corel и Kodak. Эталон стандарта IT8.7/3 отличается от эталонов сканирования большим количеством цветовых образцов и их специальным расположением. Результаты цветовых измерений эталонов обычно хранятся в числовом виде на диске или компакт-диске. Для создания профиля требуется сравнить результаты сканирования мишени с эталонными, заранее полученными значениями цветовых координат. По результатам сравнения специальное программное обеспечение строит искомый профиль прибора. Существует множество программ для создания профилей, ориентированных на самые различные категории пользователей и цифровых устройств. Так, к этому классу относятся Magic Match фирмы UMAX, ScanOpen, AGFA ColorTune и др.

Работа с утилитами профилирования не представляет никаких сложностей и доступна даже новичкам в области препресс. Большая часть вычислений выполняется программами автоматически, от пользователя требуется правильно расположить мишень на стекле сканера, тщательно выровнять ее и запустить процедуру сканирования.

Процедура профилирования лишается всякого смысла, если обработка эталона и штатные сеансы сканирования будут выполняться с разными установками (градиационными кривыми, уровнями яркости и пр.). Поэтому оцифровку эталона следует проводить в условиях, соответствующих рабочим.

На первый взгляд описанная схема калибровки не имеет недостатков. Более внимательный анализ раскрывает несколько слабых мест этой процедуры. Во-первых, цветовые мишени со временем стареют. Отклонения цветов от стандарта может быть незаметно при поверхностном визуальном осмотре, но для системы прецизионного цветовоспроизведения имеет значение даже легчайший дрейф параметров. Во-вторых, не существует таких промышленных технологий, которые позволяют создавать продукты с абсолютно идентичными техническими характеристиками. Полиграфия отличается, скажем, от микроэлектроники или лазерной техники значительно большими производственными допусками, поэтому печатные версии цветовых мишеней могут значительно отличаться. Измерять каждый экземпляр мишени и создавать для нее файл данных слишком расточительная затея. Обычно производители делают один файл измерений на целую партию мишеней. Размер партии зависит от производителя и может достигать десятков тысяч образцов. Чтобы ликвидировать этот источник цветовой погрешности, в комплект поставки некоторых типов сканеров высшего класса вместе с печатными эталонами входят специальные приборы — спектрофотометры, предназначенные для измерения цветовых координат.

Созданный профиль представляет собой файл с расширением .icc. Для хранения этого типа данных отводится специальная папка. На платформе Windows профили общего назначения хранятся обычно в каталоге Системный корневой каталог\System32\Spool\Drivers\Color. Чтобы использовать профиль, следует ак-

тивизировать соответствующий режим в программе управления сканированием. В результате в каждое изображение, созданное сканером, будут внедрены корректирующие данные подключенного профиля.

В этом разделе речь шла только о технике создания профилей для сканеров. Популярные в наше время цифровые камеры по своему техническому устройству очень похожи на эти приборы, но работают в отличных окружающих условиях. Если для любого типа сканеров источник света строго фиксирован (им является лампа прибора), то съемка цифровыми камерами может выполняться практически в любых условиях освещения. Все это делает теоретически безупречную схему создания профиля практически неприменимой для цифровых камер.

1.5.6. Создание профиля печатающего устройства

Создание профиля печатающего устройства — намного более сложная процедура, чем профилирование сканера или монитора. Главной причиной является различие свойств носителей и красящих материалов. Для иллюстрации этого факта достаточно обратиться к практике домашней печати цветных высококачественных изображений на струйном принтере. Каждое сочетание типа бумаги и чернил создает уникальные условия печати, которые требуют использования собственного профиля. Нестабильность внешних условий (температура, влажность, условия освещения и пр.) вносит свой вклад в это разнообразие. Второй причиной является сложность измерения результатов печати. Сканер, выполняя оцифровку оригинала, производит, по сути дела, измерение цветовых характеристик. В этой ситуации он работает как измерительный прибор, который, вероятно, нуждается в юстировке, но не требует никаких дополнительных измерительных мероприятий. Для монитора неплохие результаты может дать визуальная оценка изображения опытным специалистом по цвету. Для принтеров и печатных машин все обстоит намного сложнее. Для получения точных числовых значений цветовых координат печатного оттиска нужны специальные измерительные приборы — спектрофотометры или колориметры.

Принято считать, что все принтеры и печатные машины работают в системе CMYK. Это справедливо для подавляющего большинства современных устройств печати, но далеко не для всех. Некоторые фотопринтеры являются подлинными устройствами RGB, которые используют краски Red, Green и Blue для воспроизведения цвета на фоточувствительной бумаге или пленке. Их следует профилировать в качестве RGB использованием соответствующих измерительных инструментов и источников света.

Некоторые типы струйных и лазерных принтеров, печатающие красками Cyan, Magenta, Yellow и Black, но не использующие язык PostScript, являются скрытыми устройствами RGB. В любой операционной среде для описания графических данных используются специальные языковые средства. На платформе

Windows таким средством является язык Windows GDI, на платформе Macintosh — язык QuickDraw. Оба языка поддерживают цветовую модель CMYK с заметными ограничениями. В результате, когда растровый редактор или настольная издательская система запускает печать цветного изображения CMYK, драйвер принтера выполняет преобразование данных CMYK в систему RGB. Устройства печати, работающие на основе языка PostScript, лишены этого недостатка. Ранее они относились к элитной категории печатной техники и использовались главным образом в предпечатной подготовке высококачественных цветных изданий. В наше время все больше принтеров среднего уровня используют аппаратные или программные эмуляторы языка PostScript для вывода графических данных на печать.

Общая схема профилирования печатающих устройств любого класса и типа состоит в получении печатного оттиска некоторого эталонного графического файла. Результаты работы измеряются при помощи спектрофотометра или колориметра. Специальное программное обеспечение создает профайл на основе информации о разнице эталонных и фактических значений цветовых образцов.

Преимущества точного профилирования устройства печати часто не компенсируют издержки этого мероприятия. Многие опытные дизайнеры, работающие с прецизионным цветом, предпочитают использовать простой и проверенный метод итераций, когда печатный оттиск сравнивается с изображением на откалиброванном мониторе, а все замеченные расхождения компенсируются стандартными инструментами цветокоррекции растрового редактора.

1.5.7. Передача цветовых значений

Значения цветовых координат в системе RGB или CMYK — математическая абстракция, которая способна дать точное описание цвета только на идеальном выводном устройстве. Мнимая определенность цветовых координат рассыпается при попытке ее прямого применения на практике. Реальная цифровая техника далека от совершенства, поэтому результаты отображения одного цвета на разных устройствах могут значительно отличаться друг от друга.

Для обеспечения однозначности отображения цвета на разных устройствах используют системы управления цветом. Чтобы решить эту задачу, система CMS должна знать, как цифровое устройство видит, отображает или печатает цвет. Если система располагает сведениями об особенностях сканера, монитора или принтера, которые используются в данной проектной ситуации, то она может внести необходимые корректировки и обеспечить согласованное отображение цветов по всей технологической цепочке.

Профили являются основным источником данных об особенностях цветовоспроизведения конкретных технических устройств. Самый простой способ передачи этой информации — внедрение профиля непосредственно в графический файл изображения. Эта операция не меняет вид изображения. Приложения,

которые не могут работать с профилями, просто игнорируют данные этого типа. Для систем управления цветом внедренный профиль дает сведения о правилах интерпретации цветовых координат изображения.

Во всех случаях, когда система CMS выполняет преобразования цветовых координат, она использует данные двух профилей — исходного и целевого. Исходный профиль внедрен в изображение и описывает особенности его родительского устройства, например сканера или монитора. Целевой профиль относится к принимающему, или выводному, устройству, например другому монитору или принтеру.

После оцифровки цветного изображения в растровый редактор, например Photoshop, будет передан значительный массив RGB-данных. Чтобы редактор правильно интерпретировал эти данные, ему следует сообщить сведения о свойствах сканера, который был использован в данной ситуации. Эти сведения доставляет профиль устройства. Чтобы печатная версия изображения соответствовала оригиналу, требуется в качестве целевого принтера, например, использовать профиль выбранного устройства печати. Оба профиля сообщают CMS ключевые данные для перекодирования информации о свете. Сначала CMS по профилю сканера восстановит истинные значения RGB оригинала, а затем, используя профиль принтера, переведет их в искомые величины CMYK.

Описанная процедура является ключевой операцией любой системы управления цветом. Ее аналогия с переводом с одного языка на другой настолько очевидна, что в редкой статье или книге, посвященной этому предмету, не упоминается об этом сравнении. Основной функцией CMS является перевод с языка одного технического устройства на язык другого. В любом переводе неизбежны упрощения и изыятия; есть они в работе CMS.

Каждое техническое устройство способно воспроизвести фиксированный диапазон цветов. Этот диапазон называется цветовым охватом, гаммой или цветовым пространством. Например, в цветовое пространство любого монитора не входят краски, насыщенность которых превышает свечение зерен люминофора. Обычный четырехкрасочный принтер не способен печатать цвета типа металлик и т. д. Цвета исходного пространства, которые невозможно отобразить в целевом пространстве, называются цветами вне гаммы (*out of gamut*). Для таких тонов должна быть найдена подходящая замена. В стандарте, описывающем профили ICC, рассматриваются четыре метода имитации цветов, лежащих вне гаммы:

- **Perceptual Intent** (перцепционное преобразование). Данный метод основан на сжатии исходного цветового пространства до размеров целевого. Это сжатие выполняется таким образом, чтобы были сохранены исходный вид изображения и общие цветовые отношения оригинала. Яркость и насыщенность преобразуемых цветов могут немного измениться. Перцепционное преобразование применяется в случаях, когда велико количество цветов, выходящих за пределы целевого пространства. Оно дает хорошие результаты для фотографических изображений с широкой цветовой гаммой;

- **Saturation Intent** (преобразование с сохранением насыщенности). Это преобразование выполняет отображение исходного пространства в целевое. Приоритет отдается сохранению относительной насыщенности цветов, при этом оттенки могут испытывать незначительные изменения. Данный метод преобразования дает хорошие результаты для деловой графики, плакатов, географических карт и других изображений, выполненных в ограниченной цветовой палитре;

- **Relative Colorimetric Intent** (преобразование по относительной колориметрии). При преобразовании цветового пространства по этому методу исходный белый цвет отображается в белый цвет целевого пространства. Эта операция является моделью для всех остальных исходных цветов, которые преобразуются в целевые подобно белой точке. Цвета вне гаммы подгоняются к самому близкому оттенку целевого пространства. Метод дает хорошие результаты для изображений с небольшим числом оттенков, лежащих вне гаммы;

- **Absolute Colorimetric Intent** (преобразование по абсолютной колориметрии). Данный метод оставляет все цвета, выходящие за пределы целевого пространства, без изменений. Это преобразование сохраняет целостность образа за счет консервации отношений между цветами. Если целевое пространство имеет меньшие размеры, то некоторые исходные цвета могут стать одинаковыми. Метод дает хорошие результаты, если в исходный профиль занесена корректная информация о белой точке.

Таким образом, для работы системы CMS требуется предоставить ей данные об исходном и целевом профиле и выбрать метод преобразования. Любое преобразование цветового пространства связано с неизбежными потерями. Их причинами могут быть несовпадение размеров цветовых пространств, округление числовых данных, возможная передискретизация, накопление погрешностей и пр.

1.6. Теоретические основы оцифровки

Прошли дни, когда сканер или цифровой аппарат считались жемчужинами компьютерной периферии, а их немногочисленные обладатели принадлежали к элите пользователей. В настоящее время это недорогие приборы массового применения, которым оснащена почти каждая вторая персональная вычислительная система.

Перечень профессиональных и любительских задач, для решения которых применяются эти приборы, выглядит очень внушительно: оцифровка изображений, создание электронных архивов и коллекций, распознавание символов, копирование печатных и рукописных страниц, пополнение баз данных и многое другое. Ранее сканеры использовались в полиграфии, рекламном деле и Web-дизайне. Сейчас даже в мелком бизнесе и при обучении трудно преуспеть без подобного оснащения. Сканирование и цифровая съемка носят массовый характер и как всякое ремесло оно обладает своей технологией и секретами.

Для того чтобы создать резервную копию платежного поручения или телефонных счетов, не требуется штудировать книжку по цифровой обработке изображений. С другой стороны, оцифровка на барабанном сканере — таинство, секретами которого владеют немногие посвященные. Операторы этих устройств образуют своего рода касту; ее члены не нуждаются в советах со стороны, с возникающими проблемами они справляются самостоятельно. Когда процессом сканирования управляет новичок или любитель, то можно получить множество вариантов оцифрованного изображения, принципиально различающихся по своему качеству и размерам.

Устройство сканеров и цифровых аппаратов настолько близки, что позволяют отнести эти приборы к одному классу технических систем. Существенная разница заключается лишь в области применения этих типов устройств и предпочтений пользователей. Как правило, владельцы камер далеки от техники, область их интересов связана с художественными аспектами фотографирования. Поэтому далее будем рассматривать единственный тип оцифровывающих устройств — сканеры.

Результаты оцифровки зависят от множества самых разнообразных факторов, которые можно разделить на четыре группы:

- тип сканирующего устройства;
- разрешение;
- режим сканирования;
- препроцессорная и постпроцессорная обработка оригинала.

1.6.1. Типы сканирующих устройств

Прибор для считывания графической информации имеет множество технических реализаций, различающихся между собой производительностью, областью применения и типом обрабатываемых оригиналов. Ручные сканеры, планшетные сканеры, барабанные сканеры, слайд-сканеры, высокопроизводительные полуавтоматические приборы считывания — далеко не полный перечень типов оцифровывающих приборов. На уровне физических принципов действия все они описываются одной функциональной схемой, но различаются по внешнему виду, техническим характеристикам и, конечно, приемам работы.

Ручные сканеры присутствуют в этом перечне только ради полноты классификации. Это устройства вчерашнего дня, которые повсеместно сняты с производства. По своим техническим характеристикам они являются аутсайдерами среди приборов данного класса, обладают низким разрешением, невысокой скоростью работы, узкой кареткой и требуют ручного управления. Оперативность и небольшие габаритные размеры — вот причины, которые примиряют некоторых пользователей с присутствием этих приборов в составе компьютерной периферии.

Планшетные сканеры предназначены для оцифровки отдельных листов или разброшюрованных печатных изданий. Они работают с оригиналом по принци-

Нет единых рекомендаций по выбору тактики и параметров сканирования для технических систем со столь различным устройством. Все советы, приведенные в этом разделе, относятся к планшетным сканерам офисного и полупрофессионального применения.

1.6.2. Разрешение

Разрешение — один из самых распространенных терминов в компьютерной графике. Его употребляют по отношению к самым различным приборам и объектам и, может быть, этим объясняется значительная часть тех сложностей, с которыми сталкиваются пользователи, начинающие свой путь в компьютерной графике.

Растровые изображения состоят из совокупности элементарных графических элементов — точек, которые иногда называют пикселями (от англ. picture element). На элементарном уровне любая растровая картинка представляет собой регулярную сетку точек, каждая из которых обладает независимыми параметрами яркости и цветности. Сложение цветов и яркостей этих элементарных частичек изображения воспринимается наблюдателем как целостный графический образ.

Важнейшей характеристикой точечного изображения является его разрешение. Разрешение — количество точек (dot) или пикселей (pixel), приходящееся на единицу длины. Как правило, в качестве линейной единицы измерения используются дюймы (inch). Отсюда наименование этого параметра — dpi (dot per inch) или ppi (pixel per inch).

Разрешение — логическая единица измерения. Она описывает плотность точек графического изображения. На логическом уровне описания ни сами пиксели, ни результирующее изображение не имеют физических размеров. Они обретают конкретную протяженность только при выводе на определенное техническое устройство — принтер, монитор, проектор и пр.

Пусть изображение с разрешением в 100 dpi имеет высоту и ширину по 200 пикселей. Его фактическая высота (ширина) легко находится делением высоты (ширины), измеренной в точках, на разрешение. Изображение представляет собой квадрат со стороной 2 дюйма. Если, не меняя количества точек по сторонам, увеличить разрешение в 2 раза, то фигура получит новые размеры, равные 1 дюйму. И наоборот, уменьшение разрешения влечет за собой увеличение фактических размеров, если пиксельные размеры остаются неизменными.

Если известны физические размеры изображения и его разрешение, то легко найти количество составляющих точек. Пусть сканируется квадратная картинка со стороной в 3 дюйма и разрешением 100 dpi, тогда оцифрованное изображение будет включать в себя 300 точек по каждому направлению.

Что произойдет, если оставляя число точек неизменным, менять фактические линейные размеры изображения? Точные ответы дает простая и наглядная аналогия. Представим, что носителем изображения является материал с неограниченной способностью к растяжению и сжатию. Если сильно растянуть такую

страницу, то увеличатся и размеры отдельных точек. В результате дискретная структура картинки, ранее не заметная для наблюдателя, станет очевидной. Типичный пример такой ситуации иллюстрирует рис. 9 цветной вклейки, где приведено изображение с разрешением в 72 dpi и его вариант, увеличенный в 5 раз.

Понятие «разрешение» применяется не только к растровым изображениям; оно служит важнейшей характеристикой многих цифровых приборов и процессов. Так, качество сканера, объем графической информации, который способен обработать этот прибор, во многом зависит от его разрешения. Это в полной мере относится к планшетным, ручным, листовым сканерам и оцифровывающим устройствам, предназначенным для обработки слайдов и диапозитивов. У приборов, построенных по классической схеме, горизонтальное разрешение зависит от плотности фоторецепторов сканирующей головки, вертикальное — определяется минимальным шагом смещения каретки вдоль оригинала. Иногда первую величину называют оптическим разрешением, а вторую — механическим. У многих современных моделей сканеров эти параметры различаются. Как правило, механическое разрешение выше оптического. Обычное разрешение современных планшетных сканеров равняется 1200x2400 dpi, а у моделей полупрофессионального класса оно может достигать 2400x4800 dpi, лучшие представители этого типа приборов могут иметь еще более высокое разрешение.

Очевидна связь между разрешением сканера и качеством оцифровки. Сканирование с более высокими установками разрешения при прочих равных условиях позволяет получить более качественный вариант картинки. Большая плотность выборки позволяет внести в цифровую версию мелкие детали, которые в противном случае могли бы быть просто пропущены.

Иногда применительно к сканерам слово «выборка» вместо «точка», и единицы измерения плотности оцифровки spi (sample per inch) вместо dpi. Их аргументацию можно принять, если бы не многолетняя терминологическая традиция, которая разрешает описывать привычными терминами «точка» и dpi любые цифровые устройства (мониторы, сканеры, видеокамеры) и процессы (сканирование, видеомонтаж и пр.).

Результат оцифровки зависит от размеров пикселей. Точки большого размера огрубляют растровое изображение, делают видимой его дискретную структуру. При неизменных размерах оригинала плотность выборки и размеры точек связаны по закону обратной пропорциональности. Чем выше разрешение, тем меньше размеры элементов изображения, снятых приборов с оригинала (рис. 10 цветной вклейки).

Утверждение о положительном влиянии высокого разрешения на результаты оцифровки хорошо согласуется с нашим повседневным опытом и легко принимается на веру. Однако, как и большинство постулатов, очевидных для здравого смысла, оно справедливо только для некоторой усредненной ситуации. Можно привести примеры, когда увеличение плотности выборки не дает заметного улучшения качества и, более того, способно повлечь за собой деградацию оцифрованного изображения.

В описаниях сканеров иногда указывают очень большие значения разрешения, заведомо превосходящие технические возможности этих приборов. В таких случаях скорее всего речь идет о так называемом интерполированном разрешении. Интерполяцией в математике называют процесс вычисления промежуточных значений функции или величины по их опорным значениям. Тот же самый смысл имеет это понятие и в сканировании. На основе матрицы оцифрованных точек, снятых прибором с оригинала, при помощи специального программного обеспечения строятся промежуточные пиксели. Их цветовые и яркостные параметры рассчитываются по соседним точкам на основе алгоритмов усреднения или по более сложным зависимостям. Иными словами, программа сканирования самостоятельно рассчитывает «недостающие» точки, например, получив со сканера сетку размером 5х5 точек, она может расширить ее до размеров Юх 10 и более.

В немногих случаях использование искусственно завышенного интерполированного разрешения является оправданным. Например, сканирование штриховой графики (карандашных рисунков, рукописного или печатного текста, планов, чертежей и пр.) позволяет получить более гладкие границы объектов и линий. За более высокое качество результата часто приходится платить значительным увеличением размеров графического файла.

Уже упоминалось о том, что по объективным техническим причинам оптическое и механическое разрешение сканера могут не совпадать. Если в техническом паспорте устройства указывается разрешение 600х1200, то это значит, что максимальная вертикальная плотность точек в 2 раза выше горизонтальной. Несложный анализ показывает, что если для сканирования выбрано разное разрешение по осям координат (например, 600х1200 dpi), то в оцифрованном оригинале будут потеряны исходные пропорции. Этого не происходит, потому что сканер самостоятельно уравнивает плотности по направлениям и добавит по горизонтали недостающие точки за счет интерполяции.

Процедура интерполяции часто используется и в тех случаях, когда задано разрешение сканирования, не кратное оптическому. Например, сканер способен работать с разрешением 300х600 dpi, а в установках управляющей программы задано 175 dpi. Если обрабатывается оригинал маленького размера, который планируется значительно увеличить, то использование высокого интерполированного разрешения становится не только оправданным, но и часто необходимым.

1.6.3. Глубина цвета устройства оцифровки

Термин «глубина цвета» относится не только к цветовым моделям — это одна из важнейших технических характеристик любого устройства оцифровки. Как, например, истолковать строчку технического описания сканера, в которой говорится о его 24-битной глубине цвета? Это значит, что данный прибор может производить 8-разрядную выборку для каждого цветового канала. Иными словами, это устройство среднего уровня, способное сканировать в цвете оригина-

лы с ограниченным цветовым диапазоном, например плакаты, афиши, географические карты, архитектурные планы, рисованные книжные иллюстрации и т. п.

Давно прошли времена, когда сканеры могли продуцировать только полутонные черно-белые изображения, т. е. имели глубину цвета, равную 8 битам. Большинство современных устройств оцифровки обладают 30-битной и более глубиной цвета. Лучшие марки полупрофессиональных планшетных сканеров имеют глубину, равную 48 битам, что составляет 16 бит на один канал. Это позволяет представить колоссальное количество тоновых градаций $2^{16} - 2^{16} - 2^{16} = 281\,474\,976\,710\,656$. Зачем такая высокая разрядность? Только несколько сотен цветов имеют названия, глаз человека не способен различить все градации даже пространства True Color, и не существует печатного оборудования, которое способно передать все оттенки столь богатой палитры. Две основные причины заставляют повышать глубину цвета устройств ввода. Первая причина — технологическая. Матрица фоторецепторов в сканерах более высокой разрядности обладает, как правило, повышенной чувствительностью и в меньшей степени «загрязняет» изображение собственными шумами. Вторая причина — программная. Большое количество битов увеличивает гибкость редактирования на всех последующих этапах обработки изображения. Многие операции с изображениями, например гамма-коррекция, изменение цветового пространства, обедняют тоновое пространство, снижают число цветовых градаций. Если начинать обработку 16-битовых каналов, то, имея достаточный запас, можно безболезненно пережить потерю некоторых малозначительных деталей. Совершенно иная ситуация складывается при работе с 8-битовыми каналами. Здесь любые потери могут иметь решающие последствия для качества изображения.

1.6.4. Диапазон оптических плотностей

Глубина цвета, или разрядность битового представления, описывает общее число градаций цвета (света или яркости), которые способно распознать сканирующее устройство. Диапазон оптических плотностей (оптический диапазон) определяет гладкость перехода между соседними тонами в оцифрованном изображении.

Оптическая плотность — характеристика обрабатываемого оригинала, которая вычисляется как десятичный логарифм отношения падающего светового потока к потоку, отраженному от непрозрачного сканируемого объекта (для прозрачных объектов, слайдов или фотографических негативов используется сила прошедшего света). Эта величина является численной характеристикой непрозрачности прозрачных оригиналов и отражающей способности непрозрачных объектов.

Предельные значения оптической плотности непрозрачных объектов равны 0 и 4. Совершенно белые непрозрачные объекты, которые отражают весь падающий световой поток, имеют минимальное значение этой величины. Идеально черные объекты и материалы, поглощающие весь попадающий на них свет, обладают максимальной оптической плотностью, равной 4.

родной организацией по стандартизации (ISO), используются мишени IT8. Это печатный образец, представляющий эталонные цвета различной яркости и насыщенности и оттенки серого цвета. Стандартом фиксируется только часть содержимого мишени, значительный ее фрагмент не регламентируется и может быть выбран по усмотрению производителя.

Многие производители сканеров, фотографического оборудования и программного обеспечения включают в комплект поставки такие образцы, например Kodak IT8 или Kodak Q60. На рис. 11 цветной вклейки показан цветовой эталон, который канадская фирма Corel поставляет вместе со своим графическим пакетом PhotoPaint.

Следует отсканировать этот образец и проверить различимость градаций серого цвета. На эталоне фирмы Corel оттенки серого представлены 24 образцами, расположенными в нижней его части. Сканер с низкими значениями оптического диапазона не сможет распознать тоновые переходы в левой и правой частях серого клина даже при самой тщательной обработке образца. На отсканированном эталоне темные области будут закрашены черным цветом без заметных тоновых переходов и нюансов, а области светлых тонов превратятся в совершенно белые. Это значит, что для данного прибора оригиналы подобной плотности будут недоступны. Описанная процедура дает только приблизительную оценку значения оптического диапазона. К сожалению, пока не разработана процедура точного измерения диапазона оптических плотностей, доступная для рядового покупателя или оператора планшетного сканера.

Оптический диапазон устройства оцифровки и число двоичных разрядов, приходящихся на один канал, связаны между собой. Можно показать, что максимальный оптический диапазон не может превосходить десятичного логарифма от количества возможных тоновых градаций одного канала. Если на один канал сканера приходится 8 бит, то, как показано ранее, максимальное число различных оттенков канала равно $2^8 = 256$. Максимальный оптический диапазон такого устройства оцифровки не может превосходить десятичного логарифма от 256, т. е. 2,4. В действительности у серийных приборов значения этой величины лежат значительно ниже теоретического максимума. Например, у самых популярных на текущий момент сканеров с глубиной цвета 36 бит теоретический предел равен 3,6. В реальности приборы этого класса имеют намного более скромные значения оптического диапазона, обычно много ниже 3,0.

1.6.5. Размеры изображений

Растровая графика всегда считалась отраслью информатики с повышенными требованиями к вычислительной мощности компьютера. Стремительный прогресс технического обеспечения снимает многие жесткие ограничения на обработку растровых изображений на персональном компьютере, но дефицит ресурсов не преодолен, а только отодвинут.

Точечная дисперсная структура растровых изображений во многом объясняет повышенные требования к подсистеме памяти компьютера. Качественная картинка требует плотной упаковки элементарных частичек изображения — пикселей, и для каждого из них нужно хранить сведения о цвете и яркости. Чтобы обеспечить возможность отмены ошибочных действий в памяти компьютера, приходится хранить несколько версий обрабатываемого изображения. В процессе редактирования надо помнить и множество дополнительных объектов, связанных с оригиналом, например снимки состояний, текстуры, кисти и пр. Перерасход оперативной памяти активизирует обращения к дисковой подсистеме и как следствие существенно замедляет работу компьютера.

Размеры изображения можно контролировать на стадии первичной оцифровки. Любая программа управления сканером выводит данные об объеме оцифрованной версии картинки. Результат зависит от физических размеров оригинала, разрешения сканирования и выбранной цветовой модели.

Пусть изображение размером 6х4 дюйма сканируется с разрешением 300 dpi. Количество выборок по горизонтали и вертикали находится умножением ширины и высоты оригинала на разрешение: $6 \times 300 = 1800$, $4 \times 300 \text{ dpi} = 1200$. Общее число точек равняется $1800 \times 1200 = 2\,160\,000$. Легко подсчитать необходимые затраты памяти для различных цветовых моделей. Если оригинал цветной и выбрана система RGB, то на каждую точку будет отведено 24 двоичных разряда, т. е. 3 байт. Для вычисления общих затрат памяти в байтах требуется умножить число точек на 3, что дает 6 480 000 байт или почти 6,5 Мбайт. Если сканировать этот оригинал в градациях серого, то результирующий объем будет в 3 раза меньше 2,16 Мбайт. Режим LineArt, где на каждую точку отводится по одному биту, потребует $2\,160\,000 \times 1 \text{ бит}$, или 264 000 байт.

Связь между размерами изображения и его разрешением не является линейной. Удвоение разрешения увеличивает объем занимаемой памяти в 4 раза, утроение — в 9 раз. Так, небольшая на первый взгляд разница между 150 и 200 dpi может обернуться многими мегабайтами дискового пространства и оперативной памяти.

1.6.6. Масштабирование

Изображение, оцифрованное на сканере, представляется на мониторе компьютера и обрабатывается в растровом редакторе таким образом, чтобы получилась печатная версия высокого качества. Эта цепочка операций настолько привычна для большинства пользователей, что мало кто задумывается о тех непростых трансформациях, которые претерпевает оригинал на этом пути.

Экранная версия изображения — это просто матрица точек со своими размерами по высоте и ширине. Изображение, имеющее размеры 600 х 400, будет занимать фиксированную долю экранного пространства на любом мониторе, независимо от принципа его действия. Оно закроет почти весь экран, если для

него выбрано разрешение 640 x 480, на экране с разрешением 1024 x 768 оно займет примерно 1/4 пространства, наконец, при разрешении 1600 x 1200 будет занято чуть более 1/9 площади экрана. При этом физические размеры, т. е. те размеры, которые рассчитываются в дюймах и сантиметрах, будут зависеть от диагонали монитора. А каковы будут размеры картинки при выводе ее на печать? В большинстве растровых редакторов по умолчанию размеры печатной версии совпадают с габаритами сканированного оригинала (если быть предельно точным, то с размерами области сканирования). Пусть требуется отпечатать изображение размером 600 x 600 пикселей. Будем считать, что размеры — данность, сейчас не имеет значения способ их получения, разрешение сканирования и установки печати. Если задать размеры печатной версии в 10 дюймов, то разрешение оригинала будет равно $600 \text{ dot} / 10 \text{ inch} = 60 \text{ dpi}$. Приведем ряд значений разрешения для разных размеров печатного оттиска:

- $600/5 = 120$;
- $600/3 = 200$;
- $600/2 = 300$.

Все эти изменения совершенно не затрагивают экранную версию, все ее достоинства и недостатки заложены на этапе сканирования и изменения области печати не влияют на качество оцифрованного оригинала. А вот на качество печатной версии это влияет, и существенно. Для любого печатного оборудования есть некоторое оптимальное значение разрешения цифрового изображения, когда устройство печати будет способно передать максимальное число деталей оригинала. Качество результата зависит и от типа выбранной бумаги. Это влияние особенно сильно проявляется для наиболее популярных в наше время печатных устройств — цветных струйных принтеров.

Пусть для выбранного принтера и сорта бумажного носителя оптимальным является значение разрешения, равное 200 dpi. Какие последствия вызовет вывод на печать выбранного оригинала с разрешением в 120 dpi? Это разрешение приведет к потере качества, поскольку часть деталей исчезнет при печати. А если побороться за результат, выбрав более высокое разрешение печати? Если, например, выставить 300 dpi или более, то принтеру будет передана избыточная информация, которой он просто не сможет воспользоваться.

Предположим, что сканированная версия изображения демонстрирует посредственное качество при выводе на монитор. Можно ли поправить дело, отпечатав ее на высококачественной бумаге с высоким разрешением? Простейший анализ показывает, что повышение разрешения печати не исправит ситуацию, поскольку объема графической информации эта процедура не меняет. Принтер использует только те данные, которые заложены в изображение на этапе оцифровки. Эти мысленные эксперименты, конечно, упрощают реальное положение дел, но действие принципа разумной достаточности для выбора оптимального разрешения печати вряд ли можно оспорить. Сейчас мы не обсуждаем технику его расчета; она будет подробно рассмотрена далее.

Итак, если зафиксировать точечные размеры изображения, то любые изменения разрешения влекут за собой модификацию области печати. Справедливо и обратное утверждение. В растровой графике это преобразование принято называть масштабированием. Зачем масштабировать изображение? Причины для этого многообразны и часто очень весомы. Многие современные цифровые камеры среднего уровня производят изображения небольшого размера, которые, будучи отпечатанными, занимают площадь почтовой марки. Настольные издательские системы требуют изображения фиксированных размеров, которые могут не совпадать с оригинальными габаритами и пр. Масштабирование не меняет физические размеры графического файла, поскольку не воздействует ни на один из параметров (число точек, глубина цвета), от которых зависит его значение.

1.6.7. Дискретизация

Изменение числа точек растрового изображения называется дискретизацией. Эта операция очевидным образом влияет на размеры экранной версии изображения, поскольку меняет габариты прямоугольного растрового поля. Поясним процедуру дискретизации на примере изображения, представленного на рис. 12 цветной вклейки. На рисунке показан фрагмент глаза утенка с многократным увеличением. Оригинальная версия картинки, которая занимает среднюю позицию, имеет разрешение в 72 dpi. Увеличение разрешения в 2 раза влечет возрастание количества точек и увеличение линейных размеров экранной версии изображения. Уменьшение разрешения производит прямо противоположные последствия.

Операцию изменения числа точек растрового массива иногда называют передискретизацией, или ресамплингом (от англ. Resampling), что в дословном переводе означает повторная выборка.

В отличие от масштабирования дискретизация — это операция не элементарная с вычислительной точки зрения, поскольку она радикально меняет структуру изображения. Пусть имеется изображение размером 400 x 400 точек. Если сократить его экранные размеры до 300 x 300 точек, на первый взгляд это означает незначительное вмешательство в оригинал — уменьшение всего лишь на четверть. Иная картина открывается, если подсчитать количество точек до операции и после. Исходное изображение состояло из $400 \times 400 = 160\,000$, а после преобразования оно насчитывает $300 \times 300 = 90\,000$ — почти в 2 раза меньше. Понятно, что такая масштабная по своим последствиям операция не может не сказаться на качестве картинки.

Еще более сложные задачи приходится решать при увеличении количества точек. Если при их уменьшении программа просто отбрасывает лишние пикселы, то при увеличении матрицы дополнительные точки надо «придумать». Добавление новых пикселов выполняется по специальным алгоритмам интерполяции.

В растровой графике получили распространение три основных метода дискретизации, которые различаются между собой скоростью работы и точностью результатов:

- Nearest Neighbor (метод ближайшего соседа). Самый простой метод интерполяции, обладающий высокой скоростью работы и результатами не самого высокого качества. В качестве образца для нового пиксела берутся характеристики его ближайшего фактического соседа. Метод дает неплохие результаты для областей с регулярной геометрией, например прямых линий, прямоугольников и пр.;

- Bilinear (билинейная интерполяция). Этот метод несколько сложнее в реализации, но дает лучшие результаты по сравнению с методом Nearest Neighbor. Параметры новой точки рассчитываются усреднением цветовых или тоновых характеристик соседних действительных пикселей изображения. Свои преимущества метод показывает при уменьшении количества точек изображения. Рациональной областью его применения является обработка изображений среднего качества;

- Bicubic (бикубическая интерполяция). Это лучший метод интерполяции, по этой причине он принят по умолчанию в профессиональных растровых редакторах, например Photoshop. Новые точки рассчитываются по существующим соседям на основе несколько более сложных алгоритмов, чем в предыдущем методе.

Что происходит с разрешением и областью печати при выполнении процедуры дискретизации? Ответ дает формула, которая определяет понятие разрешения растрового изображения:

$$\text{Длина} \times \text{Разрешение} = \text{Количество точек.}$$

Это соотношение показывает, что при любой дискретизации изображения должны меняться его фактическая длина или разрешение. С точки зрения математики обе возможности равноправны, важно только сохранить равенство правой и левой частей уравнения.

Операцию дискретизации могут выполнять не только графические редакторы, но и устройства оцифровки. При обработке оригинала с разрешением, которое не является целой частью максимального оптического разрешения сканера, осуществляется процедура, во многом напоминающая билинейную интерполяцию, выполняемую растровыми редакторами при изменении числа точек изображения. Рассмотрим эту ситуацию более подробно. Пусть требуется оцифровать оригинал шириной 3 дюйма на сканере с максимальным оптическим разрешением в 600 dpi. Простым умножением можно найти количество светочувствительных элементов, которые будут задействованы в этой процедуре. Оно равно $600 \times 3 = 1800$. Если установлено разрешение, равное половине максимального (300 dpi), то в процессе оцифровки будет участвовать 900 датчиков, т. е. каждый второй. Работу в таком режиме можно организовать элементарными средствами, не внося глубокие изменения в алгоритмы управления прибором. Совсем иная

ситуация, если выбрать такую плотность оцифровки, которая не является целой частью максимального оптического разрешения. Это приведет к нарушению регулярности расположения активных датчиков, поэтому подлинный вид сканируемого оригинала может быть сформирован только с участием специальных корректирующих алгоритмов, работающих по принципу программной интерполяции.

Выбор разрешения сканирования часто обосновывается рациональными доводами, но, несмотря на веские аргументы и стройные логические рассуждения, у пользователя почти всегда остается значительная свобода выбора. Даже в мысленном эксперименте трудно представить себе такую ситуацию, когда невозможно отступить от рассчитанного разрешения сканирования. В большинстве случаев качество изображения не претерпевает критических изменений даже при значительных отклонениях разрешения от рассчитанных оптимальных значений. Поэтому следует выбирать такую плотность оцифровки, которая приближает расчетное значение сверху и одновременно является целой частью максимального оптического разрешения выбранного устройства сканирования. Иными словами, если сканер способен работать с разрешением в 300 dpi, то кратные числа 75, 100, 150 dpi предпочтительнее, чем установки сканирования, не являющиеся целой частью от 300, например 120 или 175 dpi. Если для некоторого оригинала при помощи расчета или иным путем получено оптимальное разрешение, равное 140 dpi, то в реальной сессии сканирования целесообразно установить 150 dpi.

Отметим еще раз принципиальные различия между масштабированием и дискретизацией. Первая операция влияет только на печатную версию изображения, она никоим образом не воздействует на актуальные пиксели, поэтому экранная версия картинки не претерпевает никаких изменений даже при значительных преобразованиях масштаба. Ее результаты можно заметить только при выводе документа на печать. Вторая операция более сложная технически и более ответственная по своим результатам. Она выполняет глубокую перестройку изображения, при определенных условиях воздействуя на каждый его пиксел.

Вопросы для самоконтроля

1. Назовите основные задачи, решаемые когнитивной КГ.
2. Сформулируйте принципы построения векторного формата.
3. Перечислите основные положения растровой графики.
4. Какой диапазон длин волн соответствует видимой части электромагнитного излучения?
5. Что такое спектральная кривая?
6. Как называется область физической оптики, занимающаяся измерением энергии света?
7. Назовите основные светотехнические величины и единицы их измерения.
8. Как называются нервные клетки, регистрирующие хроматическую составляющую светового потока?

9. Перечислите основные цвета в порядке уменьшения чувствительности распознавания их глазом человека.
10. Что такое хроматическая адаптация и какую роль играет этот феномен в системах управления цветом?
11. В чем заключается явление метамеризма?
12. На какие классы можно разделить все известные модели описания цвета?
13. Какие цвета называются дополнительными, или комплиментарными?
14. Что такое цветовой охват и какая из цветовых моделей обладает наибольшим охватом?
15. Дайте определение спектральной кривой.
16. Назовите основные цвета в порядке их расположения на цветовом круге.
17. Таблица цветности. Ее место в системе отображения цвета.
18. Перечислите основные задачи и разновидности псевдотонирования.
19. Структура и основные функции системы управления цветом.
20. Назовите основные положения калибровки устройств цветопередачи.
21. Что такое профиль устройства и чем он отличается от профайла?
22. Что такое глубина цвета и как этот технический параметр влияет на качество системы цветовоспроизведения?
23. Дайте определение оптической плотности и диапазона оптических плотностей. Назовите предельные значения, которые может принимать диапазон оптических плотностей.
24. В чем заключается процедура ресамплинга растрового изображения?
25. В чем заключается процедура интерполяции растрового изображения? Перечислите основные способы интерполяции.

2. ТЕХНИЧЕСКИЕ СРЕДСТВА КОМПЬЮТЕРНОЙ ГРАФИКИ

Рассмотрены технические средства, с помощью которых реализуются математические методы и алгоритмы КГ. Особое внимание уделено графическим подсистемам современных компьютеров, основным особенностям функционирования графических адаптеров и графических процессоров. Подробно описана работа графического конвейера и реализация различных аспектов получения фотореалистичных трехмерных изображений.

2.1. Общие сведения об ЭВМ, используемых для обработки графической информации

Технические средства (ТС) и общесистемное программное обеспечение (ПО) являются инструментальной средой графической системы. Они образуют физическую среду, в которой реализуются математические методы, алгоритмы, математические модели в рамках специального ПО компьютерной графики (КГ). Пользователь (инженер, дизайнер, художник, редактор) взаимодействует с этой средой и, используя средства КГ, создает графические объекты различной сложности. Технические средства и общесистемное ПО реализуют различные, но взаимосвязанные функции по созданию графической информации (ГИ), ее преобразованию, хранению и выводу.

С помощью технических средств КГ решают следующие задачи:

- ввод исходной графической информации;
- оперативное общение пользователя с графической системой;
- преобразование графической информации;
- хранение графической информации в различных форматах;
- отображение графической информации;
- документирование графической информации.

Основу ТС КГ, решающих перечисленные задачи, составляют вычислительные системы (ВС), включающие процессоры, оперативную память, внешние запоминающие устройства, устройства ввода ГИ, устройства вывода ГИ, устройства взаимодействия пользователя с компьютером, телекоммуникационные и сетевые устройства. Перечисленные задачи ТС решают совместно с общесистемным ПО, под которым обычно подразумеваются операционные системы (ОС) ЭВМ. Назначение ОС — организация вычислительного процесса в ВС, рациональное распределение вычислительных ресурсов между решаемыми зада-

ся наборы задач, предлагаемые независимой и некоммерческой организацией Standard Performance Evaluation Corporation (SPEC). Оценки производительности большинства современных ЭВМ, публикуемые этой организацией, являются достаточно объективными и надежными в отличие от оценок, приводимых производителями микропроцессоров и ЭВМ. Существует также ряд организаций (например, www.ixbt.com), проводящих сравнительное тестирование ЭВМ на различных популярных задачах и публикующих результаты тестирования.

Емкость ОЗУ определяет возможности ЭВМ выполнять сложные программы, обрабатывающие большие объемы данных. Емкость ОЗУ измеряется в байтах, килобайтах (1 Кбайт = 1024 байт), мегабайтах (1 Мбайт = 1024 Кбайт) и гигабайтах (1 Гбайт = 1024 Мбайт). Емкость ОЗУ у ЭВМ, используемых для обработки графической информации, колеблется от десятков мегабайт до единиц гигабайт.

Емкость ВЗУ определяет возможности ЭВМ по хранению и архивированию больших объемов данных и определенного числа программ, характеризующего универсальность ЭВМ. Емкость ВЗУ у ЭВМ, используемых для обработки графической информации, составляет от десятков гигабайт до единиц терабайт.

Пропускная способность подсистемы ввода-вывода определяет возможности ЭВМ по обмену информацией с периферийными устройствами (ПУ) ЭВМ и другими устройствами. Она измеряется максимальным числом единиц информации, передаваемых через подсистему ввода-вывода за единицу времени. Реальная пропускная способность подсистемы ввода-вывода современных ЭВМ измеряется от сотен килобайт до десятков гигабайт в секунду.

Другие параметры ЭВМ характеризуют надежность функционирования, стоимость приобретения и эксплуатации ЭВМ и т. п. В популярной литературе часто приводятся параметры, относящиеся обычно к отдельным устройствам. Например, значения тактовой частоты центрального процессора ЭВМ, тактовой частоты системной шины, емкости памяти видеоадаптера, емкости кэш-памяти второго уровня и т. п.

2.1.2. Классификация ЭВМ

ЭВМ можно классифицировать по различным признакам. ЭВМ, используемые для обработки графической информации можно разделить на две группы — универсальные (общего назначения) и специализированные. Большинство ЭВМ, оперирующих с ГИ, относятся к универсальным, а специализированные ЭВМ предназначены для решения узкого круга очень сложных задач КГ. Примером специализированных графических ЭВМ может служить многопроцессорная суперЭВМ Onyx4 UltimateVision фирмы Silicon Graphics, содержащая от 2 до 64 ЦП и от 2 до 32 графических процессоров.

Часто ЭВМ классифицируют по числу обрабатываемых потоков команд и потоков данных. Эта классификация была предложена Флинном (Flinn) в 1955 г.

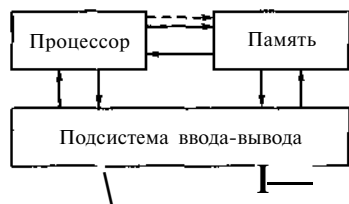


Рис. 2.1. Структура однопроцессорного компьютера

и к настоящему времени устарела, однако ее часто используют при объяснении работы ЭВМ, поскольку современные компьютеры содержат элементы большинства из этих структур. В соответствии с этой классификацией различают четыре типа структуры компьютеров:

- SISD (Single Instructions — Single Data) — однопроцессорный компьютер, который в любой момент времени управляется единственным потоком команд и обрабатывающий единственный поток данных. Структура такого компьютера приведена на рис. 2.1;

- SIMD (Single Instructions — Multiple Data) — многопроцессорный компьютер, управляемый единственным потоком команд (т. е. все процессоры одновременно выполняют одну и ту же команду или пропускают ее) и обрабатывающий несколько потоков данных. В таких компьютерах процессоры соединены в виде матрицы, возможно, многомерной или одномерной, где каждый процессор обрабатывает собственный поток данных. В системе команд многих современных процессоров имеется набор SIMD-операций, предназначенных для работы с векторами и используемых для обработки графической информации. Структура матричного компьютера изображена на рис. 2.2;

- MISD (Multiple Instructions — Single Data) — многопроцессорный компьютер, управляемый несколькими потоками команд (т. е. все процессоры одновременно выполняют команды, относящиеся к разным потокам) и обрабатывающий единственный поток данных. В этих компьютерах процессоры соединены в конвейер и результаты с выхода одного процессора поступают на вход следующего. Ряд узлов современных процессоров организован также в виде конвейера. Структура конвейерного компьютера приведена на рис. 2.3;

- MIMD (Multiple Instructions — Multiple Data) — многопроцессорный компьютер, управляемый несколькими потоками команд и обрабатывающий много потоков данных. В таких компьютерах процессоры могут быть соединены в ви-

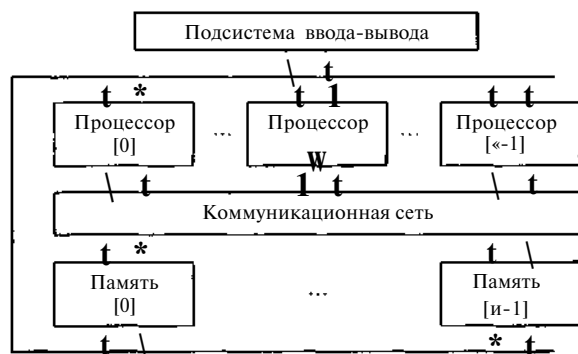


Рис. 2.2. Структура матричного компьютера

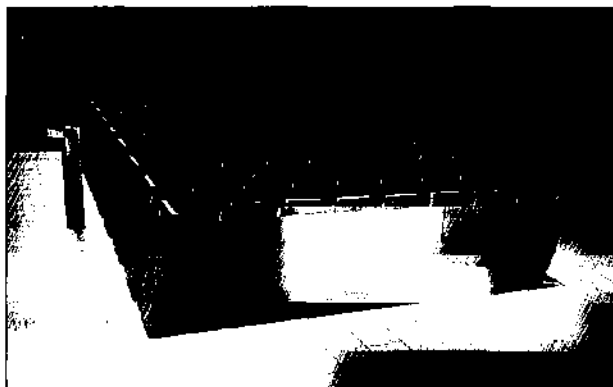


Рис. 2.5. Суперкомпьютер BlueGene/L фирмы IBM

числений, таких как моделирование атмосферных явлений, решение астрономических задач, прогнозирование погоды, разведка нефтяных и газовых месторождений и т. п. Как правило, суперкомпьютеры создаются под конкретные задачи или научные программы и изготавливаются в единичных экземплярах из серийных комплектующих. Суперкомпьютеры содержат сотни и тысячи процессоров, имеют большую оперативную память и очень высокое быстродействие. Они состоят из большого количества различных аппаратных средств, стоят десятки миллионов долларов, занимают большие помещения, а иногда и специально построенные здания (рис. 2.5).

Многие современные суперкомпьютеры созданы по кластерной технологии (Cluster). По этой технологии компьютер строится из нескольких десятков вычислительных машин, связанных между собой и функционирующих как единая система. Кластерные суперкомпьютеры легко масштабируются и позволяют получать высокое быстродействие и высокую готовность.

Быстродействие суперкомпьютеров обычно измеряется во ФЛОПСах (FLOPS — Floating Point Operations Per Second). ФЛОПС — количество арифметических операций с плавающей запятой, выполняемых в секунду. Производные единицы: 1 МегаФЛОПС (МФЛОПС) = 1 млн арифметических операций в секунду; 1 ГигаФЛОПС (ГФЛОПС) = 1 млрд арифметических операций в секунду; 1 ТераФЛОПС (ТФЛОПС) = 1 трлн арифметических операций в секунду.

Организация TOP500 Supercomputer sites (www.top500.org) с 1993 г. дважды в год публикует статистику по 500 наиболее мощным суперкомпьютерам, определяя их производительность на тестовой программе High-Performance Unpack Benchmark (HPL) решения системы алгебраических уравнений. Характеристики пяти лучших компьютеров по данным на ноябрь 2006 г. приведены в табл. 2.1.

Мэйнфреймы — высокопроизводительные компьютеры с большими вычислительными ресурсами, способные решать сложные задачи, обрабатывать большие объемы данных и выполнять обработку нескольких тысяч запросов одновременно.

Таблица 2.1

Вычислительная система	Производитель	Заказчик	Число процессоров	Тип процессора	Максимальная производительность (TFLOPS)
BlueGene/L EServer Blue Gene	IBM	DOE/NNSA/LLNL	131072	Power PC 440	280,60
Red Storm	Cray Inc.	NNSA/Sandia National Laboratories	26 544	Opteron 2,4 GHz dual core	101,4
BGW EServer Blue Gene/L	IBM	IBM Thomas J. Watson Research Center	40 960	Power PC 440	91,29
ASCI Purple eServer pSeries p575	IBM	DOE/NNSA/LLNL	12 208	PSeries 575	75,76
MareNostrum BladeCenter JS21 Cluster	IBM	Barcelona Supercomputing Center	10240	PPC 970, 2,3 GHz	62,63

Конструктивно мэйнфреймы выполняются в едином корпусе в форме шкафа или тумбы (отсюда и их название), к которому могут подключаться многочисленные терминалы (рис. 2.6). Как правило, мэйнфреймы отличаются очень высокой надежностью.

Мэйнфреймы обычно используют для хранения и обработки больших баз данных, а также для создания крупных web-узлов с большим количеством клиентов.

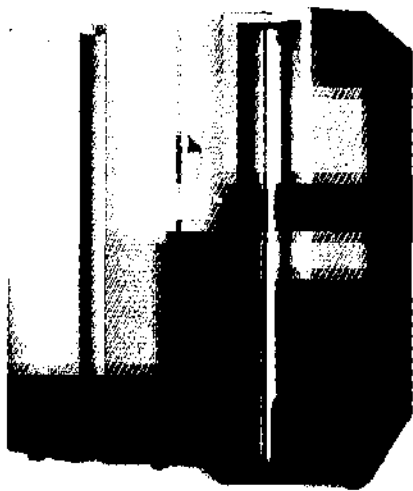


Рис 2.6. Мэйнфрейм IBM zSeries 990
2003 г.

Серверы — компьютеры, которые в вычислительных сетях являются центральными управляющими и информационными узлами. На серверах хранится большое количество информации, в том числе и *графической*, которую могут использовать все компьютеры, подключенные к сети, в зависимости от их статуса.

Сервер определяет работоспособность всей сети, сохранность баз данных и другой информации, поэтому серверы имеют систему хранения данных, отличающуюся большой емкостью и высокой надежностью, возможность замены неисправных блоков при непрерывной работе («горячая» замена)



Рис. 2.9. Персональная рабочая станция

разнообразными устройствами для ввода графической информации и манипулирования изображениями — от простой мыши до больших графических планшетов или специальных шлемов для работы в режиме виртуальной реальности.

Персональные рабочие станции — графические рабочие станции, выполненные на вычислительной платформе, используемой в персональных ЭВМ, как правило, это платформа WINTEL. Вычислительная платформа — совокупность центрального процессора (в данном случае — микропроцессор фирмы Intel) и ОС (в этом случае — вариант ОС Windows фирмы Microsoft). Обычно в качестве персональных рабочих станций используются высокопроизводительные персональные ЭВМ, укомплектованные дополнительными периферийными устройствами в зависимости от назначения станции (рис. 2.9).

Персональные компьютеры (ПК) — компьютеры, предназначенные для индивидуального использования одним пользователем автономно или в сети совместно с другими компьютерами. Персональные компьютеры бывают настольные, переносные и карманные. С точки зрения аппаратной и программной совместимости большинство современных ПК совместимы с IBM PC.

Персональные настольные компьютеры предназначены для работы в лабораторных условиях, в офисе или кабинете. Их располагают непосредственно на рабочем месте, обычно на столе, в соответствии с их названием. Это наиболее распространенные компьютеры, составляющие большую часть всех компьютеров в мире. Настольные персональные ЭВМ в зависимости от их возможностей и назначения можно разделить на профессиональные, офисные, учебные и бытовые.

Как правило, конструктивно настольные компьютеры и рабочие станции состоят из центральной части — системного блока — монитора, клавиатуры и мыши, подключенных к системному блоку. Конструктивное оформление системного блока отличается большим разнообразием — от классического горизонтального или вертикального до самых экзотических решений дизайнеров (рис. 2.10). В некоторых моделях ПК монитор и системный блок совмещены.



Рис. 2.10. Настольный компьютер с вертикальным (а) и горизонтальным (б) системным блоком

Переносные (мобильные) персональные компьютеры широко используются наравне с настольными компьютерами. Современные переносные компьютеры часто называют ноутбуками (от англ. notebook), или блокнотными компьютерами. Ноутбук функционально аналогичен настольному ПК и часто не уступает ему по техническим параметрам. В ноутбуках используется такое же ПО и ОС, что и в настольных ПК. Основная особенность ноутбука — возможность автономной работы с питанием от встроенного аккумулятора. Это позволяет использовать ноутбук в различных условиях при отсутствии питающей сети. Конструктивно ноутбук содержит жидкокристаллический дисплей, клавиатуру, совмещенную с системным блоком, жесткий диск и оптический дисковод (CD-ROM, CD-RW или комбинированный DVD+RW). Рядом с клавиатурой размещается манипулятор для управления курсором. Размеры ноутбуков соответствуют портфелю или небольшой сумке.

Широкое распространение ноутбуков сдерживалось их высокой стоимостью по сравнению с настольными компьютерами, однако по мере развития технологии изготовления для них комплектующих их стоимость снижалась, что обусловило повышение спроса и интенсивное развитие их производства. В настоящее время все основные производители настольных компьютеров предлагают большое число моделей ноутбуков, отличающихся функциональными возможностями и стоимостью. Более того, за последние годы появились новые виды ноутбуков:

- *мультимедийные* — отличаются достаточной производительностью и функциональными возможностями, необходимыми для комфортного решения большинства задач мультимедиа: качественное воспроизведение фильмов с DVD с многоканальным звуком; воспроизведение музыки на Hi-Fi уровне; редактирование и монтаж видеофайлов; 3D игры в высоком разрешении; большой экран с широкими углами обзора (как правило, 17" по диагонали); полный набор коммуникационных возможностей и интерфейсных портов; удобная полноразмерная клавиатура. Их стоимость обычно превышает 2500 долл.;

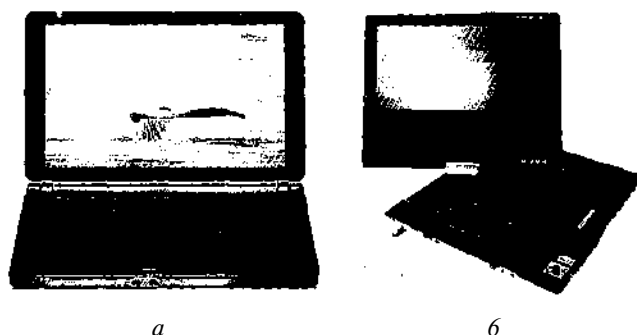


Рис. 2.11. Субноутбук (а) и планшетный ноутбук (б)

- *субноутбуки* — самые компактные и легкие модели (размеры менее 11", вес до 2 кг, диагональ дисплея до 11") с достаточно высокой функциональностью, предназначенные в первую очередь тем, кто часто путешествует. Субноутбук может обеспечить неплохую комфортность в работе, однако высокой производительности от подобных компьютеров ожидать не следует. Часто их возможности ограничены офисными приложениями, интернет-браузером, почтовым агентом и прочими не особо требовательными к ресурсам приложениями (рис. 2.11, а). Их стоимость составляет около 2000 долл.;

- *планшетные ПК (Tablet PC)* — по оснащенности и габаритным размерам близки к субноутбукам, но оснащены сенсорным экраном, позволяющим выполнять различные операции с помощью стилуса или пальца, включая ввод рукописного текста и его распознавание. Выпускаются два вида планшетных ПК: чистые планшетные ПК (без клавиатуры) и планшетные ноутбуки (имеющие обычную клавиатуру и поворотно-откидной сенсорный экран (рис. 2.11, б)). Стоимость планшетных ПК составляет около 2000 долл.

Карманные (или наладонные) переносные компьютеры (КПК) помещаются на ладони или в кармане. КПК также называют *наладонники* (от англ. — palmtop). Кроме палмтопов существуют карманные компьютеры, которые называют PDA (personal digital assistant). Общее название карманных компьютеров — handheld computers — компьютеры, которые держат в руках (рис. 2.12).

Все карманные компьютеры в зависимости от наличия клавиатуры делятся на две большие группы: КПК с клавиатурой и КПК без клавиатуры. КПК с клавиатурой внешне похожи на ноутбук, уменьшенный до карманных размеров. КПК без клавиатуры оснащены сенсорным экраном и информация вводится на экран при помощи специальной указки — стилуса, при этом может использоваться экранная клавиатура или написание символов стилусом непосредственно на экране. Стоимость карманных ПК колеблется от 300 до 1000 долл.

В карманных компьютерах программы хранятся в микросхемах энергонезависимой памяти. В набор программ обычно входит ОС, текстовый и графический редакторы, система баз данных и электронные таблицы, программы для

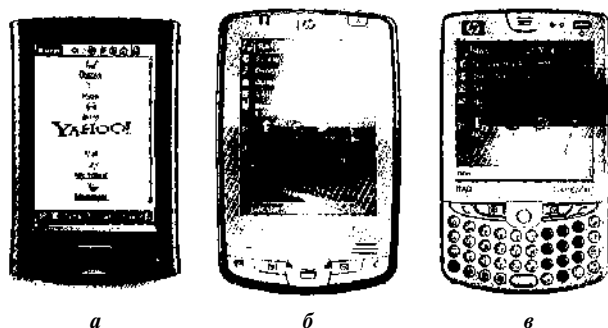


Рис. 2.12. Карманные компьютеры:

а — палмтоп; *б* — Pocket PC; *в* — PDA

работы в Интернете. Эти компьютеры позволяют обрабатывать документы, работать с базами данных, производить вычисления, читать электронные книги, слушать музыку, просматривать фильмы и работать в Интернете. Переносной и карманный компьютеры удобны для использования в поездках.

Карманные компьютеры в зависимости от используемой ОС также делятся на две группы — Palm OS и Windows Mobile. Причем в отличие от настольных компьютеров и ноутбуков конструкция КПК сильно связана с типом ОС и замена типа ОС для КПК возможна только теоретически. ОС Palm OS разработана специально для КПК и не предъявляет особых требований к их ресурсам. ОС Windows Mobile разработана фирмой Microsoft и хорошо интегрирована с ОС Microsoft для настольных компьютеров, что расширяет функциональные возможности КПК, однако при этом возрастают требования к ресурсам КПК.

Из всех перечисленных классов в России наибольшее распространение получили персональные ЭВМ, персональные рабочие станции и ноутбуки. Причем указанные классы ЭВМ базируются на процессорах семейства x86 фирмы Intel. Другие классы ЭВМ в России широкого применения не нашли. В связи с вышесказанным в дальнейшем основное внимание будет уделяться аппаратным средствам ЭВМ на платформе IBM PC с процессорами, совместимыми с архитектурой x86 фирмы Intel, относящимися к трем распространенным классам.

Каждая ВС обладает определенными функциональными возможностями обработки ГИ. Все возможности ВС реализуются совместно программными и аппаратными средствами и должны органично сочетаться с возможностями пользователя ЭВМ. Разделение функций между аппаратными и программными средствами направлено на повышение эффективности ВС при решении различных задач. Степень разделения функций между аппаратными и программными средствами зависит от уровня развития микроэлектроники и вычислительной техники.

Программные средства дешевы, гибки и доступны, аппаратные средства сложнее в реализации и специализированы. Соотношение по стоимостным затратам между аппаратными и программными средствами постоянно изменяется

в сторону снижения стоимости аппаратной реализации функций. Поэтому одной из тенденций развития ВС является решение все большего числа функций аппаратными средствами. Особенно ярко эта тенденция проявляется в области визуализации трехмерных графических изображений.

2.1.3. Аппаратные средства ЭВМ

Традиционно аппаратные средства ЭВМ делят на две группы — центральные и периферийные устройства.

К *центральному устройству*, непосредственно участвующим в обработке данных, относятся центральный процессор, оперативная память и подсистема ввода-вывода.

К *периферийным устройствам* (ПУ) относятся устройства, реализующие функции ввода, вывода, подготовки данных и хранения больших объемов информации. Общим для всех ПУ является то, что они преобразуют форму представления данных без изменения их содержания.

Центральный процессор (ЦП) предназначен для преобразования информации в соответствии с выполняемой программой, управления вычислительным процессом и устройствами, работающими совместно с процессором.

Оперативная память (ОП) предназначена для хранения, приема и выдачи данных и программ. Функции ОП в современных компьютерах реализуются сложной многоуровневой системой запоминающих устройств (ЗУ). Функции основного хранилища информации выполняет оперативное запоминающее устройство (ОЗУ), имеющее большую емкость и высокое быстродействие. Однако быстродействие ОЗУ не позволяет работать на тактовой частоте ЦП, поэтому для согласования скорости работы ЦП и ОЗУ используется многоуровневая кэш-память. При этом верхний (самый быстродействующий) уровень кэш-памяти обычно работает на тактовой частоте ЦП и размещается непосредственно на кристалле ЦП. Число уровней кэш-памяти определяется разницей в быстродействии ЦП и ОЗУ и обычно равно двум.

Подсистема ввода-вывода предназначена для реализации обмена данными между ОЗУ и различными ПУ без участия ЦП, согласования скорости работы ПУ и ОЗУ, унификации протоколов обмена данными и обеспечения возможности изменения конфигурации ВС путем подключения разнообразных ПУ. С каналами подсистемы ввода-вывода, к которым подключаются непосредственно ПУ, связано понятие *интерфейса* — совокупности аппаратных средств сопряжения канала и устройства управления (контроллера) ПУ, а также унифицированных сигналов и протоколов обмена данными между устройствами.

Одновременная работа нескольких ПУ обеспечивается большой разницей в скорости работы ПУ и канала. Подсистема ввода-вывода может осуществлять обмен данными с несколькими ПУ, распределяя между ними время использования общих средств подсистемы в режиме мультиплексирования.

ПУ, используемые в системах КГ, можно разделить на несколько групп:

- внешние запоминающие устройства;
- устройства отображения ГИ;
- устройства оперативного взаимодействия пользователя с ЭВМ в графическом режиме;
- устройства документирования ГИ;
- устройства ввода ГИ;
- устройства организации локальных вычислительных сетей.

Построение и функционирование компьютеров опирается на два основополагающих принципа:

- программное управление вычислительным процессом;
- хранение программ и данных в общей памяти.

При выполнении любой программы все устройства ЭВМ взаимодействуют между собой. Схема, отражающая связи между устройствами, называется структурной схемой ЭВМ. Структурные схемы современных ЭВМ отличаются большим разнообразием. Для персональных ЭВМ на базе процессоров x86 наиболее распространена структурная схема, включающая микросхему ЦП, микросхемы памяти и две микросхемы северного и южного мостов (так называемый чипсет), связывающие все устройства ЭВМ. Фактически чипсет в современных ЭВМ выполняет значительную часть функций подсистемы ввода-вывода, дополненных функциями блоков, отвечающих за объединение устройств ЭВМ в единую систему. Подобная схема на базе чипсета i915 Express и процессора Pentium 4 фирмы Intel представлена на рис. 2.13.

При выполнении программы процессор выбирает из ОП очередную команду и определяет, какие действия необходимо совершить при выполнении этой команды. Затем выбирает из памяти числа, над которыми надо проделать действия, определяемые данной командой, выполняет их и одновременно выбирает из памяти следующую команду, отправляет полученный результат в память — и весь цикл повторяется сначала. Так, постоянно взаимодействуя, процессор и память выполняют любую программу, обращаясь если необходимо к ПУ.

Процессоры. Процессоры можно классифицировать по разным признакам. По назначению различают процессоры центральные, специализированные, передачи данных, коммуникационные и др. В зависимости от особенностей системы команд процессора часто упоминаются следующие типы процессоров и компьютеров: CISC — Complex Instruction Set Computer, RISC — Reduced Instruction Set Computer и VLIW — Very Long Instruction Word.

В CISC-процессорах обычно используется принцип микропрограммного управления и для них характерно наличие команд регистр-память, большое число методов адресации, переменная длина кода команды и большое число тактов при исполнении команды. В RISC-процессорах применяется управление с жесткой логикой и для них характерен сокращенный набор команд, команды в основном типа регистр-регистр, малое число методов адресации, фиксированная

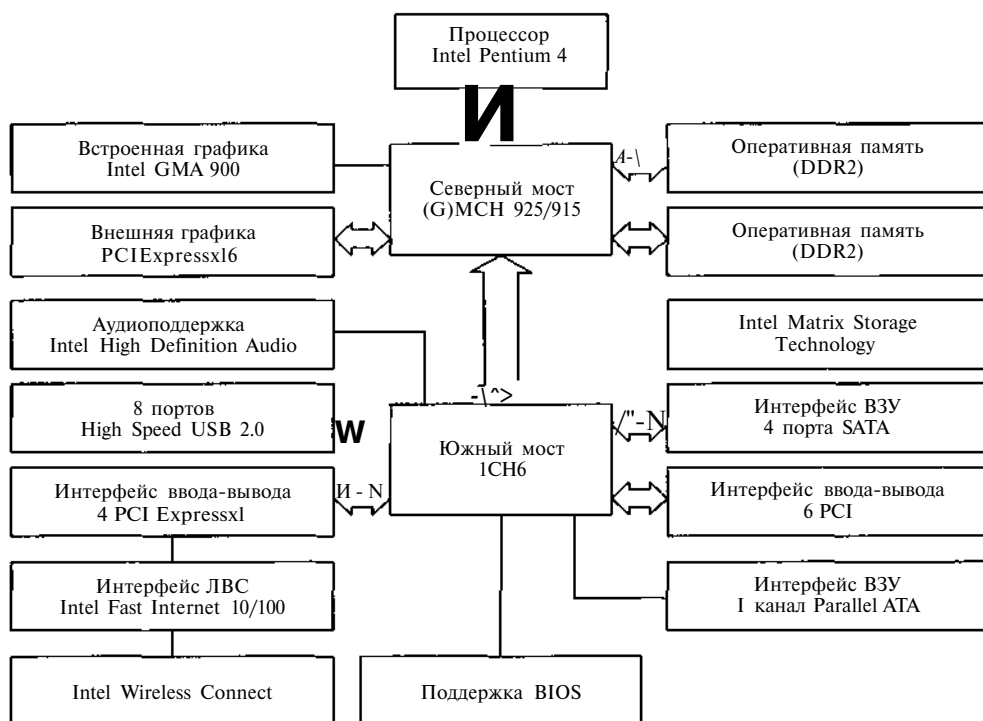


Рис. 2.13. Структурная схема ЭВМ на базе чипсета i915 Express

длина кода команды и исполнение любой команды за один такт. Последняя особенность требует конвейеризации декодирования и исполнения команд. RISC-процессоры имеют существенно больший потенциал повышения производительности процессора по сравнению с CISC-процессорами при незначительном усложнении программирования. Поэтому практически все современные процессоры относятся к RISC-процессорам.

Процессор **VLIW** (очень длинное слово команды) по архитектуре относится к процессорам, у которых в слове «команды» упакованы коды нескольких простых взаимонезависимых операций. В этом случае после выборки командного слова содержащиеся в нем операции декодируются и исполняются независимо и параллельно, что позволяет повысить производительность процессора. Элементы VLIW-архитектуры используются в семействе 64-разрядных процессоров Itanium фирмы Intel.

Центральный процессор дешифрирует и выполняет команды программы, взаимодействует с подсистемой ввода-вывода, иницируя и контролируя ее работу, воспринимает и обрабатывает сигналы, поступающие от различных устройств ЭВМ и ПУ (запросы прерывания). Функционирование ЦП — выполнение последовательности команд, определяемой исполняемой программой. Каждая команда — совокупность кода операции, которую необходимо выполнить, и

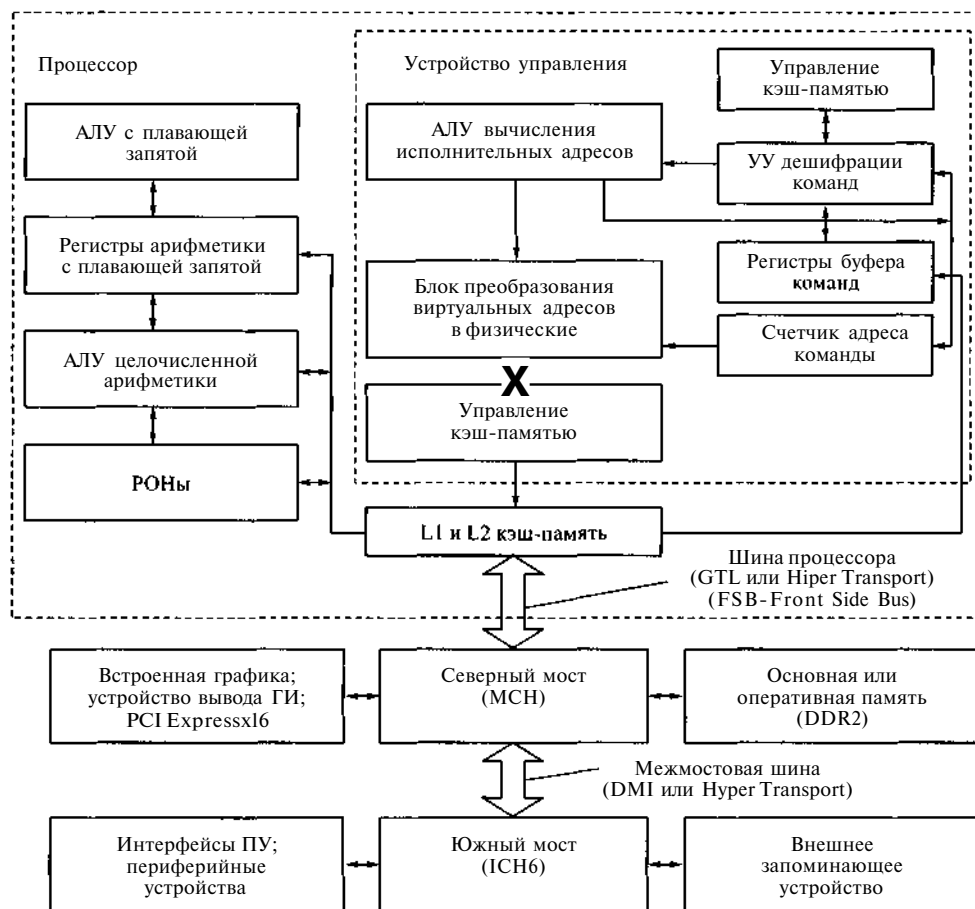


Рис. 2.14. Упрощенная структурная схема ЭВМ

кода, определяющего операнды, участвующие в операции. Все процессы в ЦП синхронизированы с сигналами определенной частоты *{тактовой частоты}*, вырабатываемыми специальным генератором. Величина, равная периоду тактовой частоты, называется *машинным тактом*.

Упрощенная структурная схема ЦП, его связей с чипсетом и другими устройствами ЭВМ приведена на рис. 2.14. Здесь изображены наиболее важные узлы центральной части компьютера, такие как ОП, кэш-память, регистры, устройство управления, арифметическое устройство, устройство адресной арифметики, блок преобразования адресов и др.

Шина процессора (FSB — Front Side Bus) представляет собой сложное устройство, к которому через микросхемы северного и южного моста чипсета могут подключаться: устройства процессора, ОП, ВЗУ и устройства ввода-вывода, устройства взаимодействия с компьютерной сетью. Шина обеспечивает взаимный

обмен информацией между устройствами, подключенными к ней. Существует несколько типов шин (GTL, Hyper Transport и др.), различающихся быстродействием, логикой работы, числом и правилами подключения устройств к ней. Быстродействие шины процессора в основном определяется частотой ее работы (не путать с тактовой частотой ядра процессора), которая значительно ниже тактовой частоты процессора. В состав шины входят регистры, хранящие принимаемую и передаваемую информацию, и собственная система управления.

Шина процессора обеспечивает интерфейс между процессором и устройствами, внешними по отношению к процессору. От внутренних схем процессора она получает заявки на прием и выдачу информации другим устройствам шины.

Одним из устройств шины является *оперативная*, или *основная*, *память*, из которой выбираются команды и числа. Для того чтобы выбрать из памяти, например команду, следует по шине запросить готовность памяти к работе, послать в устройство управления памятью адрес байта, с которого начинается эта команда, принять эту команду на выходной регистр памяти и отправить в процессор. Примерно такие же операции производятся при выборке чисел и при записи информации из процессора в ОП.

В процессорах имеется также система *внутренних шин*, абонентами которой являются блоки самого процессора. По внутренним шинам передаются команды, операнды и адреса, т. е. осуществляется обмен содержательной информацией между внутренними регистрами и блоками процессора. Внутренние шины осуществляют передачи данных намного быстрее, чем внешняя шина. От пропускной способности внутренних шин во многом зависит быстродействие процессора.

Кроме того, в процессоре есть сеть передачи управляющих сигналов, включающих в работу блоки и схемы процессора.

Устройство управления (УУ) — основной узел процессора, который задает ритм работы всех его устройств и организует их согласованное взаимодействие. В состав устройства управления входят:

- УУ, вырабатывающее управляющие сигналы для чтения очередной команды из ОП, формирования адресов операндов, чтения операндов в арифметико-логическое устройство (АЛУ), выполнения операции в АЛУ, записи результата в ОЗУ или локальную память (ЛП), инициирования процедур ввода-вывода и т. п.;
- управляющие регистры, предназначенные для временного хранения управляющей информации;
- система приоритетных прерываний, обеспечивающая реакцию ЭВМ на запросы прерываний от различных источников внутри и вне ЦП;
- система контроля и диагностики для обнаружения сбоев и отказов в аппаратных средствах ЦП и устранения последствий сбоев.

Система приоритетных прерываний необходима для реализации мультипрограммного режима работы. *Прерывание программы* — способность ЦП при возникновении некоторых ситуаций, требующих немедленной реакции ЭВМ, прекращать выполнение текущей программы и передавать управление программе, реализующей реакцию ЭВМ на возникшую ситуацию. Устройства, требующие

вмешательства ЭВМ, называются *источниками прерывания*, а выдаваемые ими сигналы, вызывающие прерывание программы, — *запросами прерывания*. Причинами прерывания могут быть сбои и отказы в работе аппаратных средств, запросы на обмен данными от ПУ, программные ошибки и др. При обработке запроса прерывания процессор прерывает ход вычислительного процесса, формирует код прерывания, слово состояния прерванной программы и обеспечивает переход к программе, обрабатывающей данное прерывание. Для управления последовательностью обработки запросов прерываний используется группирование источников запросов и присвоение каждой группе определенного приоритета. Обработка одновременно возникающих запросов прерывания осуществляется в соответствии с их приоритетами. В ЦП обычно имеется возможность программного управления приоритетами некоторых групп прерываний.

К управляющим регистрам относится счетчик адреса команды, в котором автоматически формируется адрес команды, подлежащей выборке из ОП. Этот регистр назван счетчиком в связи с тем, что он в основном работает как счетчик, значение которого автоматически наращивается на некоторую константу для получения следующего по порядку адреса команды, подлежащей считыванию из памяти. Этот процесс изменения содержимого счетчика команд прерывают команды переходов, которые загружают в счетчик новое содержимое — адрес перехода, извлекаемый из команды перехода или вычисленный другим способом. Полученный тем или иным способом адрес команды передается в ОП для выборки следующей команды, подлежащей выполнению.

Важную роль играет управляющий регистр, характеризующий состояние процессора. В разрядах этого регистра можно задать указания о выполнении программы в режиме пользователя или в режиме работы ОС, блокировании прерываний, включении или отключении механизмов защиты памяти. В нем хранится информация о характере результата выполненной операции, необходимая для выполнения команд условных переходов и многое другое в зависимости от типа и сложности процессора.

К устройству управления относится дешифратор команд — логически сложное устройство, которое на основе анализа поступившего из ОП кода команды формирует серию сигналов и данных, управляющих работой соответствующих устройств процессора. В процессе работы дешифратора команд из кода команды выделяется поле кода операции. В зависимости от значения выделенного кода производится дальнейший анализ. Выделяется также адресное поле, в котором указываются правила формирования адресов операндов или адреса перехода и выдается соответствующая команда в устройство *адресной арифметики* на формирование необходимого адреса. Чаще всего устройство адресной арифметики производит модификацию кода адреса, записанного в адресном поле команды, путем суммирования его с содержимым одного из регистров процессора, называемого адресным регистром или регистром-модификатором. Сформированный исполнительный виртуальный адрес передается в блок преобразования виртуальных адресов — в физические и затем в ОП или другие устройства процессора.

Команды, поступившие в ЦП из ОП, проходят достаточно сложный и длительный путь до момента реального исполнения тех действий, которые указаны в соответствующей команде. Сначала команда в исходном машинном виде извлекается из ОП по адресу, сформированному в счетчике команд. Как правило, она извлекается из быстрой кэш-памяти ЦП, в которую попадает из более медленной ОП. Затем выбранная команда поступает в специальные регистры буфера команд. В этих регистрах последовательно накапливаются команды, подлежащие исполнению. Здесь команда подвергается предварительному преобразованию к виду, удобному для дальнейшей ее расшифровки, после чего попадает в дешифратор команд, в котором разделяется на поля кода операции и адресные поля.

На основе кодов, содержащихся в этих полях, генерируются управляющие импульсы, заставляющие работать исполнительные блоки процессора. По информации, находящейся в адресном поле, формируется адрес операнда, который используется в процессе выполнения команды. Для формирования такого исполнительного адреса иногда требуется выполнение некоторых арифметических операций. Во многих процессорах существуют специальные устройства адресной арифметики, выполняющие необходимые операции с адресами. Полученные таким образом адреса операндов еще не пригодны для того, чтобы непосредственно обратиться к ОП. Их еще следует преобразовать. Специальный блок процессора занимается преобразованием исполнительных логических адресов в физические адреса ОП для их реального извлечения из нее и размещения в регистрах ЦП.

С устройством управления тесно связан генератор тактовой частоты, выдающий тактирующие импульсы. Многочисленные электронные схемы устройств процессора срабатывают только в дискретные моменты времени, связанные с появлением на входах этих схем тактирующих импульсов. Время работы отдельных устройств процессора измеряют необходимым для этого числом тактов.

Тактовая частота — важнейший технический параметр, определяющий быстродействие процессора. Разные операции для выполнения требуют различного количества тактов. Самые короткие, например логические операции, требуют для своего выполнения всего один такт, некоторые, например деление, могут потребовать более десяти тактов. Разработчики электронных схем, выполняющих те или иные операции, стремятся к тому, чтобы сократить число тактов, необходимых для выполнения операций.

Существует несколько критериев определения производительности процессора, один из которых формулируется как среднее число тактов, необходимых для выполнения одной команды. В современных процессорах это число меньше единицы, т. е. за один такт могут выполняться несколько операций.

Первые ЭВМ, появившиеся в мире, работали на частоте несколько килогерц, затем частота повысилась до сотен мегагерц. Современные процессоры работают на частотах, достигающих нескольких гигагерц.

Блок преобразования адресов выполняет работу по преобразованию логического (виртуального) адреса в адрес физической памяти. Компиляторы преобразуют исходную программу в последовательность команд машины (машинный код). В процессе этого компилятор вычисляет адреса памяти, по которым предполагается разместить переменные, константы и другие объекты программ. Эти адреса предполагаемого размещения называются логическими (иногда программными). Оттранслированная программа без изменений размещается ОС в свободном месте физической памяти машины, что, естественно, требует согласования логических адресов с физическими адресами ее реального размещения в процессе выполнения программы.

При независимой трансляции несвязанных друг с другом программ их логические адреса будут, как правило, пересекаться. ОС компьютера, располагая программные объекты в физической памяти, заботится о том, чтобы физические адреса данных и команд независимых программ не совпадали между собой.

В простейшем варианте компилятор готовит программу в относительных адресах. Преобразование относительных адресов в физические адреса при загрузке программ и данных для исполнения производится путем прибавления к нему константы (базового адреса), указывающей на начало программы в физической памяти. Относительный адрес также можно считать частным случаем программного или логического адреса, близкого к понятию виртуального адреса, преобразование которого в реальный адрес памяти производится не аппаратурой блока преобразования адресов, а программным путем на стадии загрузки или с использованием средств базирования.

Виртуальный адрес отражает модель памяти, предоставляемой программе. Таких моделей существует несколько: линейная, страничная, сегментная, сегментно-страничная модели. В современных процессорах блок преобразования адресов производит отображение виртуальной модели памяти в простейшую линейную модель физической памяти.

Кэш-память — буферное, скрытое от пользователя ЗУ ассоциативного типа между ОП и процессором, позволяющее минимизировать время обращения к основной ОП и тем самым сократить общее время решения задач. Кэш-память имеет более высокое быстродействие, но значительно меньше по объему по сравнению с ОП. Кэш-память имеет ряд особенностей:

- прозрачна для программиста, т. е. отсутствуют команды обращения и управления кэш-памятью;
- управление ею полностью реализовано на аппаратном уровне.

В кэш-памяти хранятся наиболее часто используемые команды и данные, автоматически выбираемые из ОП на основе анализа потока обращений к ней. При обращении процессора к ОП он фактически обращается к кэш-памяти, и при наличии необходимой команды или данных они выбираются гораздо быстрее из кэш-памяти, чем из ОП. При отсутствии требуемых данных или команды в кэш-памяти происходит сбой кэш-памяти (*cache missing*) и необходимо

обращение к ОП и перепись требуемых данных в кэш-память. Для нормального функционирования кэш-памяти при каждом сбое необходимо решить две проблемы:

- *замещение* — при замене данных в кэш-памяти необходимо выбрать данные, удаляемые для освобождения места под новые данные;
- *когерентность* — соответствие данных в ОП и их копии в кэш-памяти.

В большинстве современных процессоров для снижения риска сбоя имеется два-три уровня кэш-памяти, обозначаемых латинской буквой L и отличающихся емкостью и скоростью обмена. При этом, как правило, кэш-память первого уровня (L1) имеет емкость до нескольких десятков килобайт, работает на частоте ядра процессора и разделена на две независимых части: кэш-память команд и кэш-память данных. Кэш-память второго уровня (L2) обычно имеет емкость до нескольких мегабайт и также работает на частоте ядра процессора. Для обеспечения возможности работать на частоте ядра процессора оба уровня кэш-памяти размещаются непосредственно на кристалле микросхемы процессора.

Блок целочисленной арифметики, или арифметики с фиксированной точкой, или целочисленное АЛУ, выполняет арифметические и логические операции над целыми числами в двоичном представлении: сложение, вычитание, умножение, деление с остатком, логические операции, сдвиги. В процессе функционирования для повышения быстродействия АЛУ часто использует локальную память (ЛП), в которой могут храниться исходные операнды, промежуточные результаты, адресные константы и другие данные. Локальная память — самая быстрая в структуре ЭВМ и обычно реализуется в виде набора регистров общего назначения. Под разрядностью АЛУ понимают разрядность чисел, подлежащих обработке. Разрядности n для целых положительных чисел соответствует диапазон представимых чисел от 0 до $2^n - 1$.

Блок арифметики с плавающей точкой (АЛУ с плавающей точкой) выполняет арифметические операции с вещественными числами, представленными порядком и мантиссой. Диапазон вещественных чисел, представимых в компьютере, как правило, значительно больше количества представимых в компьютере целых чисел. Это связано с формой представления чисел и тем, что для представления чисел с плавающей запятой выделяется, как правило, большее число двоичных разрядов, к разрядам для представления мантиссы добавляются разряды для представления порядка.

Необходимо отличать число и диапазон представимых чисел. Если суммарное число двоичных разрядов в слове, выделенных для представления чисел, равно n , то число различных чисел, которые могут быть представлены этим словом, равно 2^n . Большой интерес с точки зрения точности вычислений представляет отображение представимых в машине чисел на числовую ось.

Пусть числа представляются как $2^p \cdot m$, где порядок p занимает три разряда, включая разряд знака, и изменяется от +3 до -3, а мантисса (m) занимает четыре разряда, включая разряд знака, и изменяется от +7 до -7. В такой машине можно представить 2^{m+p} чисел, в нашем примере — 128, которые на числовой оси будут

расположены неравномерно. Нормализованным числом в машинном представлении называют число, абсолютная величина мантиссы которого находится в диапазоне от 1 до 0,5. Существует понятие *цены младшего разряда* мантиссы, равной 2^{-m} . Знание этой величины важно для оценки точности вычислений.

Регистры общего назначения (РОН) в основном служат для промежуточного хранения операндов и результатов вычислений и непосредственно связаны с целочисленным АЛУ. Их также можно использовать для вычисления адресной информации. Если операнды находятся на быстрых РОН, то процесс вычислений ускоряется, так как нет необходимости обращаться в основную сравнительно медленную ОП.

Регистры арифметики с плавающей запятой служат для промежуточного хранения операндов и результатов вычислений над вещественными числами, представленными с плавающей точкой. Эти регистры обычно по разрядности в 2-3 раза длиннее быстрых РОН, что необходимо для достижения высокой точности выполнения арифметических операций над вещественными числами.

Наборы мультимедийных инструкций. В 1996 г. стало ясно, что ПК «созрели» для мультимедиа. Фирмой Intel была выдвинута концепция NSP (Native Signal Processing — собственная обработка сигналов), согласно которой мощный ЦП должен сам выполнять обработку сигналов (аудио, 3D-графики, видео, связь), которые традиционно выполняли DSP (digital signal processor) — цифровые сигнальные процессоры. Для этого нужно было дополнить стандартный набор 1386-команд мультимедийным расширением. В настоящее время вместо аббревиатуры NSP предпочитают HSP (Host Signal Processing, где Host означает ЦП и противопоставляется сопроцессору DSP).

Специфика мультимедийных задач состоит в том, что они требуют выполнения специальных однопоточных операций над большими потоками данных сигнального типа. Для того чтобы повысить эффективность команд, стали использовать длинные и составные (упакованные) операнды так, чтобы обработать их одной командой. Например, в 64-битный операнд может быть включено восемь чисел по 8 разрядов, четыре числа по 16 разрядов, два числа по 32 разряда и одно число в 64 разряда. В результате одной командой можно, например, увеличить яркость нескольких графических пикселей.

Мультимедийные наборы внедряются достаточно трудно и медленно, так как необходимо переписать все приложения в области мультимедиа с учетом новых команд процессора. В хронологическом плане наборы мультимедийных команд прошли достаточно длительный путь развития. Первым был набор MMX. MMX (multimedia extension) — мультимедийное расширение набора команд. Команды MMX работают с 64-битными операндами целого типа (обычные операнды 32-разрядные). Главной является команда, реализующая скалярное произведение (перемножить два числа и добавить произведение к сумме). Набор был внедрен компанией Intel в 1997 г. Уже первый процессор Pentium MMX-166 справлялся с MIDI-софт синтезом по таблице волн. Ускорение при декодировании MPEG-клипов возрастает на 40 %, при микшировании аудиосиг-

налов — на 25 %, при выводе 3D-графики — на 30 %. Графический пакет Adobe Photoshop начинает работать в 3 раза быстрее для большинства операций. Использование набора значительно ускоряет также сжатие аудиосигнала.

Особенностью набора MMX являлось то, что в качестве MMX-регистров использовались регистры для чисел с плавающей точкой. Это удешевляло процессор, сохраняло совместимость по корпусам новых процессоров MMX со старыми, но приводило к тому, что команды MMX не могли выполняться параллельно с командами плавающей арифметики, которые в то время широко использовались в 3D-графике. Кроме того, процессору нужно было время на переключение с MMX на команды плавающей арифметики.

SSE и 3DNow! Следующим шагом стало создание новых расширений из мультимедийных команд, работающих уже с числами с плавающей точкой. Такие команды примерно в 2 раза повышают производительность процессора в соответствующих приложениях 3D-графики, программном декодировании DVD-видео, распознавании речи и др. Они с успехом заменяют обычные команды процессора.

Однако конкурентная борьба внесла свой вклад в развитие мультимедийных наборов команд и в результате два основных производителя процессоров разработали функционально равноценные, но различные, т. е. несовместимые наборы. Первым появился в середине 1998 г. набор AMD 3DNow! (в процессорах K6-2). Набор команд оперирует все теми же 64-разрядными MMX-регистрами. Полгода спустя, в начале 1999 г., появился набор Intel SSE (Streaming SIMD Extensions — потоковое SIMD-расширение) в процессорах Pentium III. Команды оперируют новыми, специально для этого выделенными 128-разрядными регистрами XMM. В каждый такой регистр можно записать по четыре числа с плавающей точкой одинарной точности (32 разряда). Осенью 1999 г. набор 3DNow! пополнился и получил название Enhanced 3DNow!. Он был реализован в процессоре Athlon. В настоящее время под 3DNow! часто понимают именно Enhanced 3DNow! (иногда встречается обозначение 3DNow!+).

Оба набора функциональны, эффективны и были поддержаны в Microsoft DirectX (это основные мультимедийные библиотеки), однако их различие создавало проблемы разработчикам приложений, вынужденным поддерживать оба набора.

Поскольку выделенные регистры SSE более эффективны, чем регистры MMX, то компания AMD лицензировала SSE и в середине 2001 г. встроила его в свои процессоры Athlon XP, а потом и в Duron (с ядром Morgan). Реализация SSE от AMD (по утверждению AMD более эффективная, чем у Intel) получила название 3DNow! Professional. Таким образом, сейчас набор SSE стал стандартом де-факто.

SSE2 — обширный набор инструкций от Intel, который также работает с 128-разрядными регистрами XMM. Составными операндами могут быть как числа с плавающей точкой, так и целые. Те команды, которые раньше работали с вдвое меньшими регистрами MMX (64-разрядными), теперь работают и с ре-

гистрами XMM. Еще одна особенность — в 128-разрядный регистр XMM можно упаковать уже не только четыре 32-разрядных числа с плавающей точкой (одинарной точности), как в SSE, но и два 64-разрядных числа с плавающей точкой (двойной точности). Можно сказать, что SSE2 заменяет MMX и SSE. Более того, теперь у приложений 3D-графики и видео, переписанных с использованием SSE2, отпадает необходимость в блоке плавающей арифметики. Кроме того, в набор были включены команды по обмену между регистрами и кэш-памятью. Набор SSE2 был впервые реализован в процессоре Pentium 4 в 2000 г.

SSE3 — набор команд от Intel, дополненный 13 новыми командами, разработан в начале 2004 г. После успешного создания набора команд SSE2 с Pentium 4, учитывая пожелания больших компаний, разрабатывающих программное обеспечение, были созданы команды SSE3.

SSE4 — очередная версия набора команд для мультимедиа от Intel должна появиться во второй половине 2007 г. вместе с Penrin, обновленной архитектурой Core для 45-нм процессоров Pentium. В этот набор предполагается добавить 50 новых команд для ускорения игровых приложений, работы с мультимедиа и 3D-моделирования.

Устройства памяти. Производительность компьютера сильно зависит от организации и параметров подсистемы памяти, реализуемой совокупностью ЗУ.

К основным параметрам ЗУ относятся: информационная емкость, быстродействие и удельная стоимость хранения информации.

Емкость ЗУ — максимальное количество данных (выраженное в словах, байтах, битах и т. п.), которое может храниться в ЗУ.

Быстродействие ЗУ характеризует временные соотношения процессов записи и чтения информации. Для оценки быстродействия используют различные параметры, из которых наиболее популярны время доступа, время выборки, (максимальная) тактовая частота и (максимальная) пропускная способность. Время доступа или время выборки — время от начала операции обращения к ЗУ до момента возможности доступа к данным или появления данных на выходных шинах ЗУ. Тактовая частота — максимально возможная частота обращения к ЗУ при гарантировании работоспособности. Пропускная способность равна произведению тактовой частоты на число байтов данных, выбираемых за один такт.

Удельная стоимость хранения информации — стоимость хранения единицы информации, учитывающая капитальные и эксплуатационные затраты.

По функциям, выполняемым в общей структуре памяти ЭВМ, выделяют три уровня памяти (соответственно три типа ЗУ): сверхоперативная память (СОЗУ); оперативная память (ОЗУ); внешняя память (ВЗУ). Такое деление памяти ЭВМ на несколько иерархических уровней объясняется невозможностью удовлетворения ряда противоречивых требований в одноуровневом ЗУ: большая емкость памяти, высокое быстродействие, малые удельная стоимость хранения информации и размеры. Наибольшую емкость имеют ВЗУ, а самым высоким быстродействием характеризуются СОЗУ. В качестве СОЗУ используется кэш-память, где хранятся копии наиболее часто используемых команд и данных из основного ОЗУ. Форми-

рование содержимого кэш-памяти осуществляется аппаратными средствами без участия программы, которая, обращаясь к ОЗУ, фактически в большинстве случаев оперирует с информацией, выбранной из кэш-памяти. В связи с тем, что в современных компьютерах предъявляются очень высокие требования к быстродействию памяти, кэш-память часто имеет сложную многоуровневую (до трех уровней) структуру и размещается непосредственно на кристалле микропроцессора.

Объем памяти современных ЭВМ в зависимости от их класса достигает десятков гигабайт при времени выборки до нескольких наносекунд. Емкость внешней памяти, иногда называемая массовой, составляет десятки терабайт при времени выборки до нескольких миллисекунд.

По принципу действия запоминающих элементов (ЗЭ) ЗУ различают полупроводниковые, магнитные и оптические с неподвижными и подвижными ЗЭ и др. Для СОЗУ и ОЗУ в настоящее время используются полупроводниковые ЗУ, в качестве ВЗУ — оптические и магнитные ЗУ с подвижными ЗЭ.

По составу операций обращения к ЗУ различают:

- ЗУ, в которых возможны запись и чтение информации — RAM (Random Access Memory) — память с произвольной выборкой), так как обращение к этой памяти возможно в любой момент времени к произвольно выбранной ячейке;
- постоянные, в них возможно только чтение информации — ROM (Read Only Memory). Постоянные ЗУ делят на программируемые (PROM — Programmable Read Only Memory), в которых возможна только однократная запись информации, и перепрограммируемые (EPROM — Erasable Programmable Read Only Memory), в которых возможна многократная перезапись информации, хотя и с очень малой скоростью.

По организации доступа к информации различают:

- ЗУ с произвольным (прямым или циклическим) доступом — СОЗУ, ОЗУ и накопители на магнитных дисках (НМД). В них время поиска информации не зависит или слабо зависит от расположения информации;
- ЗУ с последовательным доступом. Время поиска информации в них определяется расположением информации на носителе, как, например, в накопителях на магнитной ленте (НМЛ). При обращении к ЗУ одновременно выдается определенное количество двоичных разрядов, называемое шириной выборки.

По способу размещения и поиска информации ЗУ бывают:

- адресные — где каждой единице информации (байт или слово), хранимой в ЗУ, соответствует некоторый код, однозначно определяющий ее местоположение в памяти;
- безадресные — где поиск информации осуществляется не по адресу, а по другим признакам.

Среди безадресных наиболее распространены ЗУ двух типов:

- 1) стек (стековая память) — информация записывается и считывается через одну и ту же ячейку одномерной области памяти (вершину стека). По мере записи или считывания слова содержимое стека сдвигается. Считывание информа-

ции подчиняется следующему правилу: первым читается последнее записанное слово. Это позволяет использовать стек для запоминания состояния процессора при обработке прерываний;

2) ассоциативная память — поиск информации осуществляется одновременно во всех ячейках памяти по ее содержимому (ассоциативному признаку). Это позволяет в некоторых случаях существенно ускорить поиск и обработку данных.

В зависимости от организации обращений к ЗУ в современных ЭВМ применяются:

- одновходовые моноблочные ЗУ, адресация в которых идет последовательно, начиная с нулевого адреса (в любой момент времени обслуживается только одно обращение к ОЗУ);
- одновходовые многоблочные ЗУ (секционные, с расслоением), в которых память разбивается на блоки, способные работать одновременно и независимо друг от друга (допускается совмещение обращений к разным блокам ОЗУ во времени);
- многovahовые многоблочные ЗУ, в которых допускается несколько одновременных обращений к одному блоку.

Основная оперативная память. Оперативную память обычно относят к центральным устройствам ЭВМ как неотъемлемую часть любой вычислительной машины.

Запоминающее устройство для хранения программ и данных было названо оперативным, поскольку оно позволяет не только хранить информацию, но и оперативно ее изменять — сравнительно быстро выполняет операции считывания и записи.

В отличие от постоянных запоминающих устройств, хранящих однажды записанную в них информацию, работающих только на считывание, ОП может работать как на считывание, так и на запись. Накопители на магнитных дисках и магнитных лентах также позволяют записывать и считывать информацию, но их не относят к ОП в связи с низкой скоростью работы, не пригодной для оперативной работы процессора.

Вычислительную машину можно определить как устройство, изменяющее состояние памяти, переводящее ее из одного состояния в другое. В этом смысле память можно считать главным объектом работы вычислительной машины.

Существует несколько абстрактных определений памяти, связанных с теорией алгоритмов, в которых память рассматривается как структурированное множество слов некоторого алфавита. Память реального процессора также представляет собой конечное строго упорядоченное множество слов одинаковой длины над двоичным алфавитом. Порядковый номер слова называется его адресом. Соответственно, говоря об *адресном пространстве*, имеют в виду число слов в памяти. Употребляют термин «разрядность адреса», понимая под этим число двоичных разрядов, необходимых для двоичного представления номера самого большого адреса.

Объем памяти в современных машинах измеряется числом байтов, размещенных в ней. Иногда объем памяти измеряется в словах, содержащих несколько байтов. В настоящее время принято называть 2-байтовую последовательность полусловом, 4-байтовую последовательность — словом, 8-байтовую последовательность — двойным словом.

Особенности функционирования памяти в мультипрограммных ВС. Для реализации мультипрограммного режима в ВС необходимы защита памяти от несанкционированного доступа и динамическое распределение памяти между программами. Эти проблемы решаются совместно аппаратными и программными средствами. Для повышения быстродействия памяти стремятся максимально использовать аппаратные средства.

Более сложной является проблема динамического распределения памяти для повышения эффективности использования ОП при размещении в ней нескольких программ. Цель различных способов динамического распределения памяти — уменьшение фрагментации (появление несвязанных неиспользуемых участков) ОП и предоставления всем активным программам всего прямоадресуемого пространства. Наиболее эффективно динамическое распределение памяти осуществляется в системах виртуальной памяти со страничной организацией. *Виртуальная память* — способ организации памяти, при котором каждой активной программе предоставляется все прямоадресуемое пространство памяти независимо от размеров имеющейся физической памяти.

Все поле памяти, а также программы и данные при страничной организации разбиваются на части фиксированного размера — страницы. Программы и данные разбиваются на математические страницы, а поле памяти — на физические страницы. Размер страницы в различных ЭВМ изменяется от 512 до 4096 байт. Преобразование математического адреса в физический осуществляется непосредственно при обращении к памяти совместно аппаратными и программными средствами.

Требования к объему и быстродействию ОП и кэш-памяти растут практически непрерывно по мере развития аппаратных и программных средств. Сегодня даже в персональных ЭВМ оперативная память имеет емкость от сотен мегабайт до нескольких гигабайт.

ЗУ, используемые для реализации ОП и кэш-памяти в компьютере, относятся к типу памяти RAM (память с произвольной выборкой), поскольку обращение происходит в любой момент времени к произвольно выбранной ячейке. Память этого класса подразделяется на два типа — память с динамической (Dynamic RAM, DRAM) и статической (Static RAM, SRAM) выборкой. В первом случае значение бита информации в ячейке определяется наличием или отсутствием заряда на микроконденсаторе (управляемом одним-двумя транзисторами). В статической памяти в качестве ЗЭ используются триггеры (имеющие два устойчивых состояния), реализованные на четырех-шести транзисторах. Однако в связи с особенностями функционирования быстродействие DRAM ниже, а емкость больше. Вследствие большего числа транзисторов на ячейку и соответственно меньшей емкости при одинаковой площади кристалла память SRAM су-

щественно дороже. Обычно модули DRAM применяют в ОП и видеопамяти, а модули SRAM — в кэш-памяти.

Память DRAM отличается наибольшим разнообразием в технической реализации. Среди основных вариантов можно отметить следующие:

- синхронная динамическая память DRAM (Synchronous DRAM — SDRAM);
- модули памяти фирмы Rambus (Rambus DRAM — RDRAM);
- память, способная передавать два слова за такт, считывая сигнал по фронту и спаду, получила название DDR (Double Data Rate — с удвоением потока данных);
- память DDR2, способная передавать четыре слова за такт. DDR3 способна передавать восемь слов за такт.

Синхронная динамическая память (SDRAM). Согласно спецификации SDRAM, все команды и обмен данными по шине памяти проходят синхронно с тактовыми импульсами шины процессора. Поэтому все циклы внутри операции имеют одинаковую продолжительность. Помимо этого стандарт синхронной памяти определяет параметры работы с банками памяти и режимы пакетной передачи данных. Дополнительно введен так называемый регистр режима, выставляемый при подаче питания на модуль памяти. В ходе работы с его помощью могут меняться параметры пакетного режима (последовательный или с чередованием), размеры пакетов (от одноразрядного до полностраничного) и величина задержки CAS (измеряемая в тактах). При этом величина задержки CAS определяется как время, необходимое для стробирования адресной строки и активации банка памяти.

Ячейки в динамической памяти образуют так называемую матрицу, состоящую из строк и столбцов. При считывании данных содержимое одной строки (строка считается страницей — page) целиком переносится в буфер (на элементах статической памяти). После этого в строке считывается значение (0 или 1) нужной ячейки и содержимое буфера вновь записывается в прежнюю строку динамической памяти (операция перезарядки — precharge). Такие переносы данных осуществляются изменением состояния конденсаторов ячеек, т. е. происходит процесс заряда (или разряда). Для исключения потери данных периодически проводятся циклы *регенерации* с определенной частотой (refresh rate), которые обычно инициируются специализированными микросхемами. В современных модулях используют циклы регенерации, именуемые 1K, 2K или 4K, что означает число строк (в тысячах), обновляемых за один такт.

Ранее практически повсеместно в модулях памяти применялся *контроль четности* с целью проверки достоверности информации. Для этого при записи байта вычисляется сумма по модулю 2 всех информационных разрядов и результат записывается как дополнительный контрольный разряд. При чтении байта снова вычисляется контрольный разряд и сравнивается с полученным ранее.

Обнаружение и исправление ошибок (ECC — Error Checking and Correction) — специальный алгоритм, который заменил контроль четности в современных мо-

дулях памяти. Каждый бит данных включается более чем в одну контрольную сумму, поэтому при возникновении в нем ошибки можно восстановить адрес и исправить разряд с ошибкой. При сбое в двух и более разрядах ошибка лишь фиксируется, но не исправляется.

Для адресации ячеек памяти используют особенности матричной структуры. Полный адрес ячейки состоит из адресов строки и столбца. Для считывания (записи) информации на микросхему сначала подается сигнал RAS (Row Address Strobe), импульс строба строк, а затем (одновременно или с небольшой задержкой) — код адреса строки. После этого через нормируемое время задержки должен быть подан код адреса столбца, перед которым проходит сигнал CAS (Column Address Strobe) — импульс строба столбцов. Под временем выборки микросхемы подразумевают промежуток между сигналами RAS. Следующее обращение к памяти возможно только через некоторое время, необходимое для восстановления внутренних цепей. Этот промежуток называют временем перезарядки, причем оно составляет почти 90 % общего времени выборки. Данные из ячеек через усилители поступают в регистр микросхемы, откуда они становятся доступными после открытия линии DOUT (Data OUT). При операциях записи данные поступают по линии DIN (Data IN), а цикл выполняется в обратном порядке.

Любое системное устройство, обладающее правом *прямого доступа* к памяти (по одному из каналов DMA — Direct Memory Acces), при необходимости посылает запрос, содержащий адрес и размер блока данных, а также управляющие сигналы. Поскольку доступ к памяти по каналам DMA одновременно могут иметь несколько устройств (например, процессор, видеокарта с интерфейсом AGP, контроллер шины PCI, жесткий диск), образуется очередь запросов, хотя каждому потребителю ресурсов памяти требуются собственные данные, часто расположенные не только в разных микросхемах, но и в разных банках памяти. Вследствие этого возникают значительные задержки при чтении-записи данных.

Под *банком памяти* понимается:

- группа модулей асинхронной памяти одинаковой емкости, которая должна быть установлена одновременно, чтобы занять всю ширину шины данных (иначе система не будет работать);
- микросхема (или группа) на модуле асинхронной (DRAM) или синхронной (SDRAM) памяти, занимающих всю ширину шины данных.

В первом случае модули DRAM (SIMM) имеют половинную ширину шины данных и их необходимо устанавливать попарно. Во втором случае на одном модуле SDRAM (DIMM) монтируется несколько комплектов микросхем, каждый из которых имеет полную ширину шины данных. На модуле образуется два (обычно для SIMM), четыре (обычно для DIMM) или более банков, доступ к которым происходит независимо. В связи с тем что ширина шины данных в памяти типа RDRAM существенно меньше (например, в модулях PC600 и PC800 — 16 бит), микросхема емкостью 128 Мбит разбита на 32 банка памяти. Таким образом, двухканальный вариант RDRAM имеет 64 банка памяти.

В синхронной памяти (SDRAM) для запуска сигналов стробирования используется внешний импульс от шины памяти, поступающий с той же частотой, на которой она работает. Поэтому вместо указания времени стробирования в наносекундах (типовые значения — 12, 10, 8, 7 и 6 нс) для модулей SDRAM указывают максимально допустимую рабочую частоту в мегагерцах, например PC 100 или PC 133. Для памяти DDR SDRAM принято указывать частоту считывания данных: например, при физической частоте 200 МГц память маркируется как DDR400. Такой же принцип реализован в маркировке памяти Rambus: PC800, PCЮ66 и т. д.

Важным параметром скорости работы памяти является величина CL (CAS Latency) — время задержки (измеряемое в тактах шины памяти), протекающее от подачи сигнала CAS до начала пересылки данных. Для памяти SDRAM типовые значения равны CL3 и CL2, для памяти DDR значения могут иметь дробный характер, например CL2.5. Это объясняется тем, что в DDR-технологии один такт синхросигнала условно разбивается на два тика (tick), длительность которых составляет половину цикла (полупериод). Следовательно, передача данных может начаться на половине такта.

Время пересылки данных обычно записывают в следующем виде: 6-2-2-2 (измеряют в тактах микропроцессора). Это означает, что на первую пересылку данных из произвольной ячейки памяти потребовалось 6 тактов шины, а на все последующие ячейки — по 2. Так, для памяти DDR2-800 стандартом определены три варианта задержек: 6-6-6; 5-5-5; 4-4-4.

Стандарты на память устанавливаются ассоциацией JEDEC, объединяющей большинство компаний, действующих в этой области. Конкретная спецификация устанавливает набор требований к модулям памяти для гарантированного обеспечения их работы в требуемых условиях. Жестко регламентирована длина проводников в модуле памяти, длина пути тактового импульса и его временные параметры, определена ширина дорожек и расстояния между ними, электрические и другие параметры. Часто оговариваются прочие технологические требования: обязательность применения согласующих сопротивлений (терминаторов); позолоченных контактов и т. д.

К началу 2006 г. для памяти DDR2 были приняты спецификации DDR2-400, DDR2-533, DDR2-667 и DDR2-800, где цифры обозначают частоту передачи данных, при которой гарантирована работоспособность. Для модулей памяти DDR и DDR2 принято обозначать спецификацию по частоте передачи данных (например, DDR2-400 или DDR-333) или по пропускной способности — PC3200, PC2700 (измеряется в Мбайт/с). Хотя компания Rambus сама устанавливает спецификации на свою память, она придерживается принципов обозначения, принятых для DDR (частота или пропускная способность).

Память типа DDR SDRAM обеспечивает максимальную (пиковую) полосу пропускания только в случае передачи единых массивов данных. При работе с разрозненными данными производительность резко падает. Переход на более жесткие технологические нормы позволяет увеличить частоту работы памяти до

200 МГц (DDR-400), что обеспечивает пиковую пропускную способность 3200 Мбайт/с или для DDR2-800 — пропускную способность 6400 Мбайт/с при двухканальном варианте. Кроме того, в зависимости от используемого чипсета память DDR и DDR2 может работать как в одноканальном, так и двухканальном режиме. Несмотря на повсеместный переход на память DDR2, память стандарта DDR по-прежнему успешно продолжает сосуществовать с DDR2.

Модули DDR и DDR2 выполнены в тех же размерах, что и DIMM, но имеют 184 контакта, один ключ (вырез в нижней части платы) и рассчитаны на напряжение питания 2,5 В. Таким образом, с существующими разъемами DIMM они не совместимы. Модули памяти типа DDR2 производятся в новом форм-факторе в виде 240-контактных модулей DIMM, электрически не совместимых со слотами для модулей памяти типа DDR (по количеству выводов, расстоянию между выводами и цоколевке модулей). Таким образом, стандарт DDR2 не предусматривает обратной совместимости с модулем DDR.

Память типа DDR и DDR2 широко используют производители видеокарт. За счет иной корпусировки микросхем и отбора кристаллов они ставят на видеокартах более скоростную память (с частотой синхронизации до 400 МГц!). Видеопамять принято классифицировать и маркировать по минимально допустимому времени выборки данных, измеряемому в наносекундах.

Технология Rambus DRAM. Технология Rambus DRAM предусматривает совершенно новый подход к построению архитектуры подсистемы памяти. Во-первых, разработан специальный интерфейс Rambus для подключения модулей памяти к контроллеру. Во-вторых, модули памяти соединены с контроллером специальными каналами с шириной шины данных 18 (16+2) бит и шины управления — 8 бит. В-третьих, разработаны новые модули памяти RIMM.

Каждый канал Rambus способен поддерживать до 32 банков и теоретически может работать на частоте до 800 МГц. Рабочая частота канала задается собственным генератором подсистемы памяти. К контроллеру можно подключить несколько каналов Rambus. Сам контроллер работает на частоте до 200 МГц, которая определяется уже частотой системной шины. Данные передаются по обоим фронтам сигнала, что удваивает скорость обмена. При подключении к контроллеру двух каналов пропускная способность шины памяти дополнительно удваивается.

На практике реальная пропускная способность RDRAM существенно ниже расчетных значений. Тем не менее в сочетании с чипсетом Intel 850 память RDRAM показывает наилучшую эффективность по сравнению с другими типами памяти, в том числе DDR SDRAM. Для сложных задач, связанных с обработкой больших массивов данных, память RDRAM вполне отвечает своему назначению.

Подсистема ввода-вывода. Потенциальные возможности и эффективность ВС во многом определяются подсистемой ввода-вывода. Традиционно в течение многих лет в высокопроизводительных компьютерах подсистема ввода-вывода базировалась на специализированном процессоре с собственной системой команд, который управлял всей подсистемой ввода-вывода компьютера. Помимо

процессора подсистема ввода-вывода содержала несколько каналов ввода-вывода, различающихся параметрами и режимами работы с периферийными устройствами. Наличие различных каналов ввода-вывода позволяло реализовать большой набор интерфейсов, обслуживающих большое число практически любых периферийных устройств.

На программном уровне процессор ввода-вывода управлялся специальными канальными программами, написанными в кодах команд процессора ввода-вывода и включенными в состав операционной системы. Такая организация подсистемы ввода-вывода была достаточно эффективной и гибкой, что позволило ей с некоторыми изменениями сохраниться до настоящего времени в компьютерах высокой производительности и суперкомпьютерах с малой степенью параллелизма. В многопроцессорных суперкомпьютерах с массовым параллелизмом выделяется от одного до нескольких процессоров специально для организации подсистемы ввода-вывода.

Для рабочих станций и персональных ЭВМ такая организация подсистемы ввода-вывода была слишком громоздкой и отсутствовала необходимость в такой сложной организации подсистемы. В то же время использование подсистемы ввода-вывода на базе общей шины ввода-вывода, как в первых персональных ЭВМ, было малоэффективным и бесперспективным. Поэтому в результате достаточно длительного и сложного пути развития сложилась современная организация подсистемы ввода-вывода персональных ЭВМ и рабочих станций, в той или иной степени отвечающая предъявляемым к ней требованиям и допускающая ее развитие и совершенствование.

На первом этапе отказались от специальных канальных программ или специальных модулей ОС и перешли к использованию системы BIOS (Basic Input Output System) — базовой системы ввода-вывода, размещаемой на системной плате. Так называют аппаратно встроенное в компьютер программное обеспечение, которое доступно без обращения к диску. В микросхеме BIOS содержится программный код, необходимый для управления клавиатурой, видеокартой, дисками, портами и другими компонентами. BIOS использовалась уже в первых персональных ЭВМ.

Обычно BIOS размещалась в микросхеме ПЗУ (ROM, Read-Only Memory), расположенной на системной плате компьютера (этот узел часто называют ROM BIOS). Такая технология позволяет обеспечить постоянную доступность BIOS независимо от работоспособности внешних по отношению к системной плате компонентов (например, загрузочных дисков). Поскольку доступ к RAM (ОП) осуществляется значительно быстрее, чем к ROM, многие изготовители предусматривают при включении питания автоматическое копирование BIOS из ROM в ОП. Задействованная при этом область ОП называется теньвым ПЗУ (Shadow ROM).

В микросхемах BIOS используют различные типы памяти для хранения программного кода PROM. Отличие PROM от ROM состоит в том, что в ROM данные заносятся в процессе производства, а в PROM можно записывать данные с помощью устройств, называемых программаторами. EPROM — специальный

тип PROM, которая может очищаться с использованием ультрафиолетовых лучей и перезаписываться. Память типа EEPROM похожа на EPROM, но операции стирания-записи выполняются при помощи электрических сигналов.

Большинство современных системных плат комплектуется микросхемами Flash BIOS, код в которых перезаписывается при помощи специальной программы. Такой подход облегчает модернизацию BIOS при появлении новых компонентов, которым нужно обеспечить поддержку (например, новейших типов микросхем оперативной памяти). Современные типы BIOS, поддерживающие технологию Plug-and-Play, называют PnP BIOS, при этом поддержка технологии PnP обеспечивается только микросхемами Flash ROM. Полная поддержка технологии Plug-and-Play со стороны ОС Windows возможна только в случае применения PnP BIOS.

На втором этапе была предложена организация подключения всех устройств компьютера к процессору по внешней шине процессора (FSB) через специальный набор микросхем системной логики, установленный на системной (материнской) плате и часто называемый Chip Set (чипсет). Он обеспечивает работу процессора, системной шины, интерфейсов взаимодействия с ОП и другими компонентами компьютера. Очевидно, что главной проблемой на современном этапе является необходимость поддержки напрямую множества несовместимых интерфейсов.

Начнем с того, что системная шина (соединяющая процессор и контроллер памяти) и прочие интерфейсы являются асинхронными. Они не согласованы ни по характеру сигналов, ни по тактовой частоте, ни по пропускной способности. Для увязки данных и приведения их к удобной для обмена форме требуются операции преобразования и кэширования, реализуемые в специальных блоках чипсета. Сколько необходимо таких стыковочных блоков? Как минимум, шесть только для основных интерфейсов. Если же добавить сюда мост ISA-PCI, сетевой контроллер IEEE 1394, то число стыковочных блоков приблизится к десяти.

Структурная схема системной логики (на примере современного чипсета Intel 945G для процессоров Pentium 4) показана на рис. 2.15.

Многие предшествующие системные наборы включали две базовые микросхемы, которые в англоязычной компьютерной литературе принято называть соответственно North Bridge (северный мост) и South Bridge (южный мост). Северный мост обычно обеспечивал управление графической шиной AGP, шиной системной памяти, шиной PCI и взаимодействие с системной шиной процессора AGTL+. Южный мост управлял интерфейсами IDE, USB, ACPI, IEEE 1394, включал мост ISA-PCI, контроллеры клавиатуры, мыши, FDD. Северный мост подключался непосредственно к внешней шине процессора, а между собой мосты соединялись шиной PCI или другим более быстрым интерфейсом. Существовали чипсеты, содержащие встроенные видео- и звуковые контроллеры (например, Intel 810, nVidia nForce).

Современные системные наборы построены аналогично, но используют другие интерфейсы, обеспечивают больший набор функций, работают с новейшими

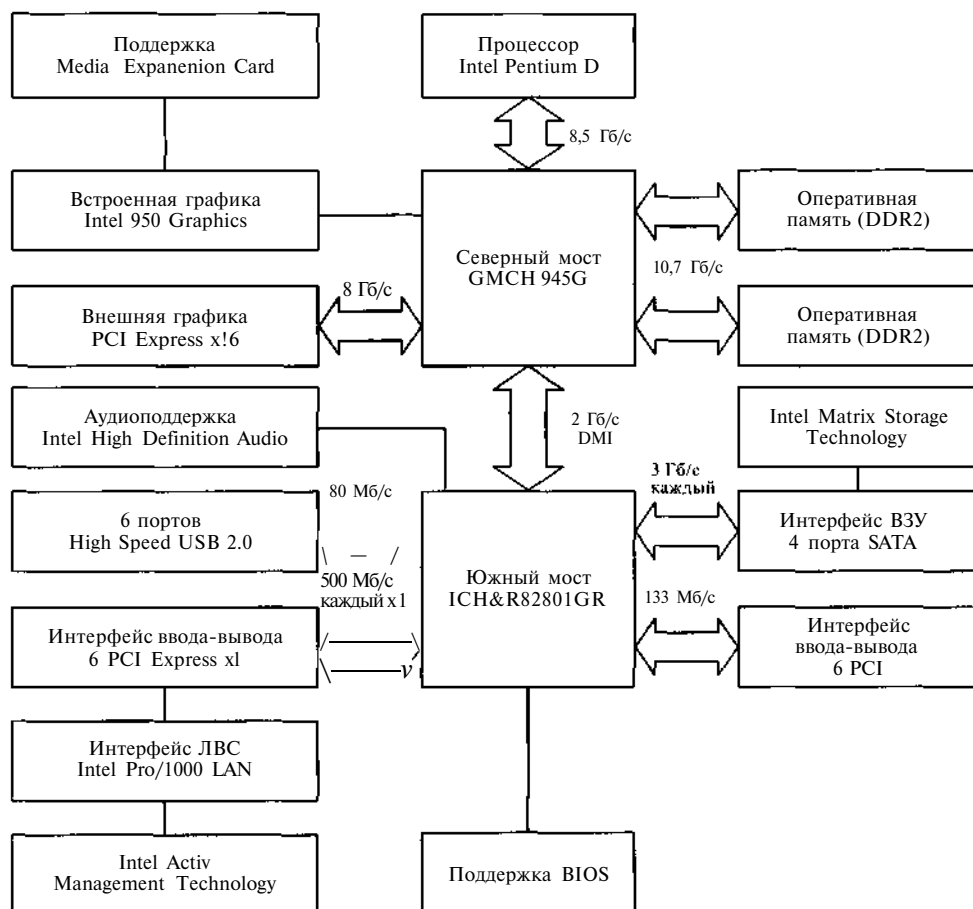


Рис. 2.15. Схема системной логики на основе чипсета Intel 945G

моделями устройств ЭВМ и содержат различные встроенные контроллеры аудио, видео и другие средства. В качестве примера можно привести характеристики линейки чипсетов компании Intel i945 Express, предназначенной для работы с двухъядерными процессорами Intel, структурная схема которой приведена на рис. 2.15.

Северный мост 945G:

- поддержка процессоров Intel Celeron D, Pentium 4, Pentium 4 Extreme Edition, Pentium D и Pentium Extreme Edition с частотой системной шины 533/800/1066 МГц;
- двухканальный контроллер памяти DDR2-400/533/667 с поддержкой до четырех модулей DIMM без ECC общим объемом до 4 Гб;
- графический интерфейс PCIEx16;
- интегрированное графическое ядро GMA 950.

2.2. Графическая подсистема ЭВМ

Шина DMI (с пропускной способностью около 2 ГБ/с) соединяет северный и южный мосты.

Южный мост ICH7/R:

- до четырех (у ICH7)/6 (у ICH7R) портов PCIEx1;
- до шести слотов PCI;
- до 2 устройств (1 канал) ATAL00;
- до четырех портов Serial ATA на четыре устройства SATA300 (SATA II, второе поколение стандарта), с поддержкой AHCI и функций вроде NCQ;
- возможность организации RAID-массива уровней 0, 1, 0+1 (10) и пяти из SATA-дисков;
- до восьми устройств USB 2.0;
- High Definition Audio (7.1) или AC'97-аудио (7.1) и MC'97-модем;
- специальный интерфейс для подключения фирменного 100-мегабитного сетевого контроллера Intel серии 82562;
- обвязка для низкоскоростной и устаревшей периферии и пр.

Классификация чипсетов возможна по различным характеристикам, однако более интересной является классификация с позиций пользователей компьютеров. Во-первых, по степени актуальности чипсеты различают:

- устаревшие (их выпуск продолжается, но линейка продуктов для данного процессорного разъема не обновляется);
- актуальные (модификации, в принципе отвечающие текущим требованиям);
- современные (последние модели для современных процессорных разъемов).

Во-вторых, по обслуживаемому процессору чипсеты для настольных систем разделяют согласно поддерживаемому процессорному разъему: Socket 478 (Pentium 4, Celeron 4) Socket 775 (Pentium 4, Celeron 4), Socket 754 (Athlon 64, Sempron), Socket 939 (Athlon 64, Athlon FX), Socket 940 (Athlon FX, Opteron).

Отдельную группу образуют чипсеты типа «все в одном» (графика, звук, сетевые интерфейсы) с высокой степенью интеграции (например, Intel 915/925, nVidia nForce4 и др.).

2.2. Графическая подсистема ЭВМ

Графическая подсистема — неотъемлемая часть практически любой современной ЭВМ, поскольку она является основным средством общения человека с ЭВМ независимо от режима (текстовый или графический). В далекое прошлое ушли все другие средства общения, такие как, например, пульт управления, пультовая пишущая машинка и т. п. Графическая подсистема компьютера состоит из аппаратной и программной частей. Аппаратная часть включает графический контроллер или видеоадаптер (внешний или встроенный в северный мост чипсета), монитор, а также интерфейсы, обслуживающие графическую подсистему: один между видеоадаптером и северным мостом чипсета; второй — между видеоадаптером и устройством отображения (монитор, телевизор, проектор и

т. п.). Программная часть обеспечивает поддержку интерфейсов видеоадаптера, монитора и приложений на уровне BIOS, ОС, драйверов и специализированных прикладных языков программирования (API). Графическая подсистема используется всеми прикладными программами (от простейших текстовых редакторов до программ трехмерного моделирования), поскольку почти вся информация, выводимая компьютером, визуализируется графической подсистемой. Приложение использует функции видеоадаптера при посредничестве драйвера, который выступает интерпретатором команд для графического процессора. В соответствии с полученными командами видеоадаптер выводит на экран монитора отображаемое изображение.

Как было сказано выше, в КГ в настоящее время используются две разновидности графических изображений — плоские, или 2О-изображения, и объемные, или 3О-изображения. Кроме того, изображения могут быть представлены в векторной или растровой форме. В процессе функционирования графическая подсистема компьютера решает три группы задач:

- задачи плоской, или 2О-графики. В основном это задачи графического интерфейса GUI (Graphic User Interface) для обеспечения взаимодействия между различными программами и пользователем. Современные графические интерфейсы принадлежат к группе так называемых \WIMP*-интерфейсов, предложенных более 15 лет назад и доживших до наших дней без существенных изменений. Графическим ядром предыдущих версий Windows являлся существующий уже больше 11 лет интерфейс на основе библиотеки GDI, которой на смену идет библиотека GDI+. Эта библиотека сочетает в себе (по крайней мере, по замыслу) все достоинства своего предшественника и предоставляет множество новых мощных возможностей. Кроме того, при проектировании библиотеки GDI+ заранее ставилась цель переноса приложений на 64-битные платформы. Кроме задач GUI к этой группе относятся задачи создания, редактирования и отображения плоских изображений. Двухмерные изображения могут быть представлены как в векторной (машиностроительные чертежи, географические карты и т. п.), так и в растровой форме (цифровые фотографии, сканированные рисунки и т. п.);

- задачи трехмерной, или 3О-графики. 3О-графика, как правило, представляет собой геометрические модели 3О-объектов, совокупность которых образует сложную геометрическую модель некоторого трехмерного объема, называемого трехмерной сценой. Последовательность геометрических моделей трехмерной сцены, отражающая динамику изменений, порождает трехмерный видеоряд. Геометрические модели 3О-графики, как правило, создаются и обрабатываются специальными программами, исполняемыми на ЦП, и хранятся в ОП компьютера. Графическая подсистема компьютера в этом случае используется для вывода и отображения трехмерных сцен. Современные видеоадаптеры способны при выполнении некоторых ограничений самостоятельно осуществлять создание и

* Аббревиатура WIMP расшифровывается: Window — окно, Icon — пиктограмма, Menu — меню. Pointer — указатель или курсор.

обработку геометрических моделей. При этом современные устройства вывода графических изображений (мониторы, принтеры, плоттеры, проекторы и т. п.) являются растровыми и могут выводить только плоские изображения или формировать на их основе стереоскопические изображения. Исключение составляют устройства вывода стереолитографии и станки с ЧПУ, используемые для создания трехмерных объектов или их физических моделей по геометрическим моделям. Эти устройства являются векторными и управляются специальными программами, которые генерируются на основе геометрических моделей объектов;

- задачи вывода и обработки видеографики. Графическая подсистема современных компьютеров, как правило, содержит аппаратные средства, обеспечивающие ускорение обработки, компрессии и декомпрессии видео и повышение качества отображения видео на экране монитора. Кроме того, поддерживается работа с различными источниками видео (DVD, HDTV, PVR, включая новейшие технологии Blu-ray и HD DVD) в различных форматах.

При проведении сравнительных испытаний различных видеоадаптеров, составляющих основу графической подсистемы компьютера, испытания обычно проводятся для задач всех трех групп, и полученные результаты используются для итоговой оценки видеоадаптера.

Трехмерная компьютерная графика в значительной степени специализируется по области применения. Можно выделить четыре основные области применения 3D-графики, использующие достаточно обособленные программные средства и методы, а также аппаратное обеспечение: видеопродукция, используемая в телевидении и кинопроизводстве; промышленное проектирование и дизайн; компьютерные игры; обучающие системы и компьютерные тренажеры.

Видеопродукция. Видеопродукция предъявляет исключительно высокие требования к реалистичности создаваемых трехмерных сцен. Такую трехмерную графику часто называют *фотореалистичной*. Действительно, в некоторых случаях невозможно различить объекты, существующие в реальном мире и созданные средствами КГ. Как правило, инструментами разработчиков трехмерной графики, используемой в кинопроизводстве, служат мощные графические станции и специализированные программы, причем синтез и обработка изображений в этом случае осуществляется, как правило, на уровне ЦП. Фотореалистичная графика высокого качества требует затрат ресурсов, превосходящих современные возможности IBM PC. Визуализацией кадров занимаются десятки компьютеров, объединенных в локальные сети или суперкомпьютеры. Время обработки одного кадра иногда составляет несколько часов даже для мощных ВС. Расчет визуализации эпизода, занимающего в фильме несколько минут, нередко загружает мощные компьютеры на несколько дней. Однако динамика развития аппаратных средств КГ настолько высока, что недалеко то время, когда качество изображений, которое сегодня представлено в лучших кинофильмах, увидим на мониторе домашнего компьютера в трехмерной игре.

Промышленное проектирование и дизайн. В области проектирования и дизайна широко используется математическое и геометрическое моделирование.

При этом предъявляются высокие требования к точности параметров создаваемых объектов: по размерам, характеристикам материала, соответствию стандартам и т. д. Часто такие системы сопрягаются с промышленным и испытательным оборудованием с числовым управлением. При этом недостаточно обеспечить высокую точность и скорость вычислений, графическая подсистема должна адекватно, с высоким качеством и быстро отображать объект на экране. Здесь важны параметры четкости и контрастности изображения, соответствие пропорций, отсутствие геометрических искажений, правильная передача и преобразование цветов и часто требуется оперативная фотореалистичная визуализация. Геометрические модели трехмерных сцен отличаются высокой сложностью, большим числом объектов, источников освещения и разнообразных материалов и создаются на уровне ЦП с использованием сложного и развитого ПО. В этой области обычно используется два варианта визуализации трехмерных сцен: оперативная, или текущая (в процессе создания проекта и его компонентов), итоговая, или презентационная (фотореалистичная, высококачественная визуализация проекта при завершении какого-либо этапа его создания). В первом варианте используется упрощенное каркасное или полигональное представление, которое возможно без наложения текстур и настройки освещения, и визуализируется в основном средствами графического адаптера. Во втором варианте качество визуализации часто превышает возможности современных графических адаптеров (включая профессиональные) и выполняется с использованием специальных программ (типа 3D Studio MAX и т. п.) и на уровне ЦП.

Компьютерные игры. В играх необходимо сочетать реалистичность объектов с возможностями аппаратной части распространенных компьютеров, обеспечить приемлемую скорость при хорошем качестве отображения сцен на массовых видеоподсистемах, учитывать не столько реальные характеристики объектов, сколько общее впечатление от игры — ее привлекательность. Как показывает практика, требования к аппаратным ресурсам, предъявляемые компьютерными играми, являются одним из основных стимулов развития графических адаптеров. Большая часть выпускаемых в мире видеоадаптеров ориентирована в основном на игровые программы, которые наиболее полно используют возможности распространенных графических карт.

В современных играх иногда применяют точное геометрическое моделирование трехмерных сцен и их фотореалистичную визуализацию, что требует наличия значительных ресурсов в игровом компьютере и стимулирует быстрое развитие графических процессоров, обладающих такими возможностями.

Игры с высокими требованиями к эффективности графической подсистемы часто используют в качестве тестовых программ. В частности, повсеместно признанными игровыми тестами являются Quake III Arena, Unreal Tournament 2004, Chronicles of Riddick, Doom 3, FarCry и др.

Обучающие системы и компьютерные тренажеры. Современные обучающие системы и компьютерные имитаторы-тренажеры для повышения эффективности и сокращения сроков обучения требуют создания виртуальной

обучающей среды, воспроизводящей целый комплекс особенностей, характерный для трех предыдущих областей: фотореалистичность, точное соответствие физической среде, возможность подключения реального оборудования, высокая скорость отображения. Если для профессиональных тренажеров, особенно авиационно-космических, часто разрабатывают специализированные графические системы, имеющие высокую стоимость, то для обучающих систем высокого уровня используются обычные персональные ЭВМ, оснащенные графической подсистемой, обладающей функциональностью, близкой к функциональности специализированных аппаратно-программных комплексов для тренажеров.

2.2.1. Принципы работы графического адаптера

Видеоподсистема компьютера предназначена для отображения графических изображений, хранящихся в той или иной форме в памяти компьютера или создаваемых на основе геометрических моделей 3Э-изображений на плоском экране монитора. Современные мониторы являются растровыми устройствами, в которых изображение на экране создается из тысяч пикселей. *Пиксел* — минимальный элемент изображения на экране, который может быть сгенерирован компьютером. Число пикселей в горизонтальной строке экрана, умноженное на число пикселей в столбце экрана, равно полному числу пикселей на экране и называется *разрешающей способностью* монитора.

В современных видеоадаптерах большая часть графических функций реализуется непосредственно в видеоадаптере на аппаратном уровне, что позволяет разгрузить ЦП, но требует использования в видеоадаптере специализированного процессора — графического или видеопроцессора. Функциональные возможности и сложность современных графических процессоров, как правило, существенно превосходят возможности ЦП. Например, графические процессоры ATI Radeon X1800 XT и NVIDIA GeForce 7800 GTX содержат 320 и 302 млн транзисторов соответственно. В их структуру входит несколько десятков специализированных процессоров, выполняющих операции арифметики с плавающей запятой. В то же время двухядерный процессор Intel Pentium Extreme Edition 840 содержит всего два процессора и 230 млн транзисторов, из которых большая часть приходится на кэш-память второго уровня.

Почти все 3Б-модели, используемые в самых различных логотипах, рекламных роликах и кинофильмах, создаются на специальных рабочих станциях, оснащенных несколькими мощными процессорами и специализированными 3D-ускорителями. Яркий пример таких рабочих станций — машины, производимые компанией SGI (Silicon Graphics Inc), являющейся также создателем стандарта OpenGL. В данной области этим рабочим станциям SGI нет равных, но подавляющее большинство профессионалов используют персональные компьютеры благодаря их невысокой по сравнению со специализированными станциями стоимости, а также большому количеству написанного программного обеспече-

ния и более удобному интерфейсу. В связи с этим все видеоадаптеры, выпускаемые для ПК, можно разделить на три больших класса:

- бюджетные офисные видеокарты для широкого применения;
- игровые видеокарты, поскольку компьютерные игры — наиболее массовая область их применения;
- профессиональные видеокарты для выполнения сложных профессиональных работ.

Профессиональные карты отличаются от игровых тем, что они предназначены для быстрого и качественного отображения трехмерных сцен с большим количеством объектов сложной формы, набором различных источников освещения и большим числом разнообразных текстур, т. е. в этом случае очень важна вычислительная мощность графического процессора и различные технологии повышения качества визуализации. Примером профессиональных графических карт могут служить карты семейств Quadro и nForce Professional фирмы Nvidia и карты семейства FireGL фирмы ATI. Обычно профессиональные графические карты в архитектурном плане практически аналогичны игровым и содержат те же графические процессоры. Основные отличия заключаются в том, что профессиональные карты обычно ориентированы на применение в качестве API OpenGL, снабжаются специальными наборами драйверов (часто ориентированными на конкретные приложения) и высокой ценой. Далее в основном рассматриваются особенности видеоадаптеров первых двух классов как наиболее массовых и часто используемых в качестве замены профессиональных из-за высокой цены последних.

Современные графические адаптеры используют последние достижения трехмерной компьютерной графики, реализуемые на аппаратном уровне и программным способом. Даже краткое перечисление характеристик, функций и поддерживаемых технологий в спецификации видеоадаптера иногда занимает несколько страниц плотного текста. Для оценки способности видеоадаптера с помощью известного набора тестирующих программ приводится внушительный список параметров, в котором трудно разобраться, не владея базовыми понятиями в области КГ. В связи с этим ниже излагаются основы устройства и принципы работы элементов графической подсистемы на популярном уровне, достаточном для самостоятельного изучения.

Параметры видеоадаптеров. Современные видеоадаптеры являются достаточно сложными устройствами и характеризуются большим числом параметров и характеристик, многие из которых оценивают качество отображаемого изображения и являются неформальными. Кроме того, в зависимости от области применения к качеству отображения графики на экране монитора предъявляются различные требования. Однако можно выделить ряд числовых параметров, позволяющих оценить графические возможности видеоадаптера и сравнить их различные модели.

Емкость графической памяти или видеопамати. Определяет сложность обрабатываемых изображений и параметры качества и скорости отображения. Качество изображения на экране монитора характеризуется совокупностью

взаимосвязанных параметров: *максимальная разрешающая способность*, *максимальное число отображаемых цветов* или *разрядность кодирования цвета пиксела*, *максимальная частота кадровой развертки*. Эти параметры задаются в окнах настройки ОС и в некоторой степени зависят от емкости графической памяти. Однако перечисленные параметры определяют только объем буфера кадра, в котором непосредственно формируется изображение, выводимое на экран монитора, а в современных видеоадаптерах в видеопамяти кроме буфера кадра размещается много других данных: буфер глубины (*Z*), данные вершин, текстуры и др. В настоящее время емкость видеопамяти достигает 1024 Мбайт, а значения указанных параметров часто превышают возможности мониторов.

Тактовая частота процессора. Обычно тактовая частота процессора существенно ниже тактовой частоты ЦП и составляет 400...625 МГц.

Тип графической памяти, разрядность шины и тактовая частота памяти. В современных видеоадаптерах используется синхронная динамическая память (SDRAM) типа DDR3 и DDR2 с разрядностью шины памяти от 64 до 256 разрядов и тактовой частотой до 1,7 ГГц.

Максимальная пропускная способность шины памяти. Определяет скорость отображения и скорость работы графического процессора. В современных видеопроцессорах максимальная пропускная способность шины памяти может достигать 50 Гбайт/с.

Тип графического интерфейса. Все современные карты используют один из двух интерфейсов либо параллельный AGP, либо более современный, последовательный PCI Express.

Число пиксельных конвейеров. Пиксельные конвейеры построены на специализированных процессорах и используются для выполнения *пиксельных шейдеров* — специальных программ обработки пикселей изображения. В современных видеопроцессорах (ATI RADEON X1900 (R580)) содержится до 48 (16х3) пиксельных конвейеров.

Число блоков текстурирования. Блоки (процессоры) текстурирования выполняют операции с текстурами для повышения реалистичности изображения на экране монитора. *Текстура* — плоское или трехмерное изображение элементарного участка реальной поверхности, которое может накладываться на трехмерные объекты с учетом их формы, положения и уровня детализации (LOD level). В современных видеопроцессорах (NVIDIA GeForce 7800 GTX) содержится до 24 текстурных блоков.

Число вершинных конвейеров. В данном случае под вершиной понимают вершину многоугольника (полигона) и точку в трехмерном пространстве, положение которой определяется координатами *X*, *Y* и *Z* и переменной *W* (учитывающей перспективную проекцию). В графических процессорах для персональных компьютеров при описании трехмерных объектов используются полигональные модели, состоящие из простейших многоугольников — треугольников, положение которых в пространстве определяется положением их вершин. Вершинные конвейеры построены на специализированных вершинных процессорах и используются для вы-

полнения *вершинных шейдеров* — специальных программ для обработки координат вершин и произвольных точек треугольников при перемещении трехмерных объектов, удаления невидимых граней объектов, сортировки вершин и т. п. В современных видеопроцессорах (NVIDIA GeForce 7800 GTX и ATI RADEON X1800 (R520)/X 1900 (R580)) содержится до восьми вершинных конвейеров.

Скорость обработки полигонов. Предельное значение скорости обработки полигонов определяется числом вершинных конвейеров и тактовой частотой работы графического процессора. Реальная скорость зависит от ряда факторов и от особенностей используемого программного обеспечения и решаемых задач. В современных видеоадаптерах высокого уровня этот параметр достигает значений в 400 млн треугольников в секунду и выше.

Скорость закрашивания пикселей. Максимальное значение скорости закрашивания пикселей определяется числом пиксельных конвейеров и тактовой частотой работы графического процессора. Реальная скорость зависит от многих факторов и в современных видеоадаптерах высокого уровня этот параметр достигает 6...7 тыс. мегапикселей в секунду.

Кроме перечисленных параметров широко используются и другие параметры, измеряемые при сравнении видеоадаптеров с использованием различных тестовых задач.

Устройство графического адаптера. Графический адаптер состоит из следующих функциональных блоков (рис. 2.16):

- графического процессора;
- 2D-ускорителя (часто входит в графический процессор);
- процессора обработки видеосигналов (часто входит в графический процессор);
- видео BIOS;
- видеопамати;
- цифроаналогового преобразователя (RAMDAC), цифрового видеовыхода (DVI) и других интерфейсов (часто входят в графический процессор);
- интерфейсов сопряжения с чипсетом системной платы;
- аналоговых элементов.

Графический процессор — GPU (Graphic Processing Unit). Непрерывный рост требований к качеству изображения привел к созданию специализированной графической многопроцессорной системы, которая обычно называется графическим процессором и занимается исключительно расчетом и формированием изображения, снимая эти функции с ЦП. Современные графические процессоры по сложности не уступают ЦП, более того, часто в них используются более эффективные технологии по сравнению с ЦП. Современные графические процессоры включают в себя комплекс разнообразных функциональных устройств, которые раньше выполнялись в виде отдельных микросхем, что позволяет повысить их производительность. Как и ЦП, графические процессоры характеризуются сложной внутренней архитектурой, рабочей частотой графического ядра, технологическими нормами, по которым изготовлена микросхема, и другими параметрами.

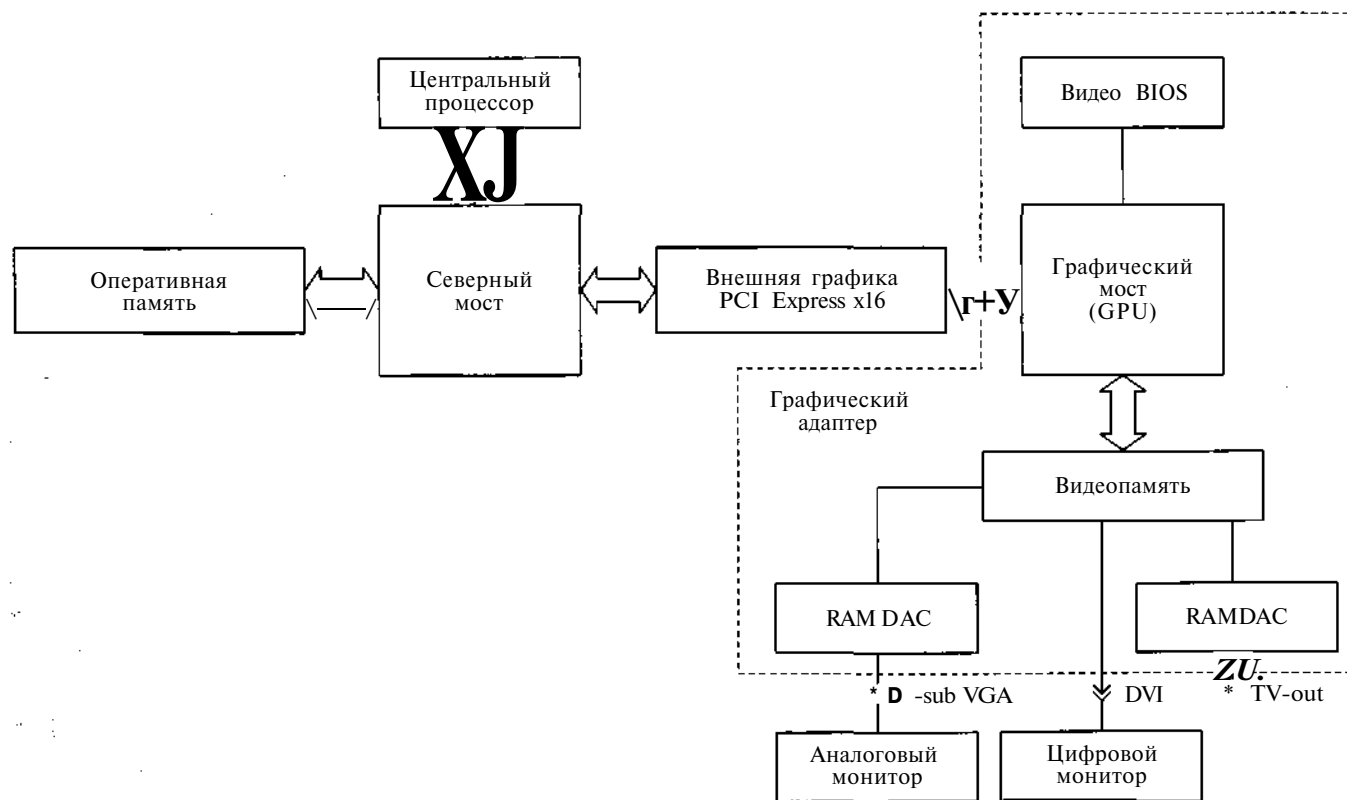


Рис. 2.16. Структурная схема графического адаптера

Видео BIOS. В микросхеме BIOS, установленной на плате видеоадаптера, хранятся программы, обеспечивающие инициализацию видеокарты, поддержку простейшего интерфейса пользователя, базовые компоненты драйвера и другие необходимые компоненты.

Видеопамять. Важную роль в повышении производительности видеоадаптера играют характеристики видеопамяти, определяемые ее типом, частотой работы, величиной задержек, шириной шины памяти. ЦП компьютера направляет данные в видеопамять, а графический процессор видеокарты считывает оттуда информацию. Кроме того, в видеопамяти хранится кадровый буфер и промежуточные данные, необходимые графическому процессору. Современные видеокарты высокого уровня оснащаются 4-канальной памятью типа GDDR3 объемом до 1024 Мбайт и рабочей частотой до 1700 МГц, шиной шириной в 256 (64x4) разрядов. Средние значения задержек видеопамяти DDR достигают 1,3 нс. Объем видеопамяти, установленной на карте, важен в первую очередь для обработки трехмерных изображений с текстурами высокого разрешения при большой глубине цвета. Большая емкость памяти необходима для быстрого вывода трехмерных изображений с высоким разрешением, поскольку в видеопамять при этом загружается огромный объем дополнительной информации, прежде всего массивы текстур.

RAMDAC. Графический процессор, получив информацию об изображении из видеопамяти, обрабатывает ее и передает либо в цифроаналоговый преобразователь (RAMDAC) для вывода на аналоговый монитор, либо в схему формирования цифрового сигнала TDMS (а через нее на цифровой видеовыход DVI) для вывода на цифровой монитор. RAMDAC состоит из памяти с произвольным доступом (RAM) и цифроаналогового преобразователя (DAC — Digital to Analog Converter). В RAMDAC обычно используется статическая память, по быстродействию близкая к кэш-памяти процессоров. Цифроаналоговый преобразователь содержит три параллельных канала — по одному на каждый цвет.

Схемотехника RAMDAC развивается очень быстро, сегодня стандартные RAMDAC обеспечивают разрешение и частоту развертки при 32-битном цвете, превышающие возможности современных мониторов. Кроме того, в них поддерживается режим Direct Color, т. е. возможность прямого доступа к элементам DAC. Это позволяет создавать независимые таблицы для каждого из трех основных цветов, что дает возможность корректировать искажения яркости цвета на электронно-лучевой трубке (ЭЛТ) монитора, вызванные нелинейной зависимостью между подаваемым на ЭЛТ напряжением и яркостью. Эта операция называется гамма-коррекцией. С ее помощью можно регулировать яркость в диапазоне от 0,3 до 4,0, где 1,0 — значение по умолчанию.

Качество получаемого изображения в решающей степени зависит от таких характеристик RAMDAC, как его частота, разрядность, время переключения сигнала с уровня черного цвета на уровень белого и обратно, варианта исполнения (внешний или внутренний). Частота RAMDAC говорит о том, какое максимальное разрешение и при какой частоте кадровой развертки сможет поддержи-

вать видеоадаптер. RAMDAC в современных графических процессорах, как правило, работают с тактовой частотой, равной 400 МГц.

Разрядность RAMDAC показывает, какое число цветов способен воспроизвести видеоадаптер. Большинство микросхем поддерживает представление 8 бит на каждый канал цвета, что обеспечивает отображение около 16,7 млн цветов. За счет гамма-коррекции исходное цветовое пространство расширяется. Современные RAMDAC поддерживают целочисленный тип данных **RGBA** (10:10:10:2) в буфере кадра для более качественного рендеринга с разрядностью 10 бит по каждому каналу, охватывающему более 1 млрд цветов.

Следует обратить внимание на то, как именно выполнен модуль RAMDAC на видеокарте — внутренним или внешним. Обычно в современных графических процессорах он размещается на одном кристалле с видеопроцессором. Многие современные видеокарты поддерживают одновременную работу с двумя мониторами, поэтому в такие карты устанавливаются по два RAMDAC и соответственно по два разъема для подключения монитора. В подавляющем большинстве видеокарт имеется также выход на телевизор (TV-out), позволяющий просматривать мультимедийные программы или фильмы на телевизионном экране. Телевизионный сигнал обычно формируется отдельной специализированной микросхемой, получающей данные от RAMDAC. Большинство современных видеоадаптеров имеют выход **DVI** (часто два) для подключения цифрового монитора. Преимущество цифрового интерфейса перед RAMDAC заключается в том, что при выводе изображения не выполняются цифроаналоговые преобразования изначально цифрового сигнала, что теоретически обеспечивает лучшее качество. На практике разница незаметна: современные видеоадаптеры выдают идеальную картинку и на аналоговые, и на цифровые устройства.

Интерфейс. Интерфейс видеокарты обеспечивает сопряжение с северным мостом чипсета. Рост производительности видеоадаптеров потребовал разработки и внедрения специализированного интерфейса AGP (Accelerated Graphic Port — ускоренный графический порт), который обеспечил приоритетный доступ видеоадаптера к системной памяти и пиковую пропускную способность шины 2133 Мбайт/с (версия AGP8X). В настоящее время осуществляется переход на последовательный интерфейс PCI Express. В современных компьютерах для подключения графических карт используется версия PCI Express x16 с максимальной пропускной способностью шины 4000 Мбайт/с в обоих направлениях.

Аналоговые компоненты. Аналоговые компоненты видеокарты обеспечивают стабильное питание микросхем и формирование сигналов нужной формы на аналоговых выходах (на монитор и телевизор). Видеокарты высокой производительности имеют высокое энергопотребление, часто потребляемая мощность превышает 100 Вт и поэтому их снабжают специальным дополнительным разъемом питания. Кроме того, высокая потребляемая мощность требует эффективной системы охлаждения и системы контроля температуры графического процессора, которые по сложности не уступают соответствующим системам ЦП. Высокая частота аналоговых сигналов на выходах к монитору и к телевизору

требует тщательного согласования с приемниками, а также защиты от помех. Качество выходных сигналов видеокарты в значительной мере определяется качеством используемых аналоговых компонентов и технологическим уровнем производства видеокарт.

Технологии SLI и CrossFire. Возможность параллельной обработки графической информации на уровне видеоадаптеров или одновременное использование двух видеоадаптеров для повышения производительности была реализована впервые компанией 3dfx в 1998 г. в видеокарте 3dfx Voodoo2 Graphics. Эта технология получила название SLI. В то время аббревиатура расшифровывалась как Scan Line Interleaving (чересстрочное сканирование кадров). С помощью этой технологии для увеличения производительности можно было объединять видеоускорители в 3Б-приложениях. Подключение производилось по следующему принципу: монитор подключался к ускорителю, а специальный кабель соединял ускоритель с 2Б-видеокартой. Возможности SLI позволяли соединять видеоускоритель с таким же ускорителем посредством плоского кабеля, похожего на ATA- или FDD-шлейф. Таким образом, вся видеосистема занимала три слота PCI, что было не очень удобно, но (по заявлениям компании 3dfx) позволяло увеличить производительность вдвое. Однако реальный прирост производительности составлял 20...50 %.

Принцип работы Voodoo2 SLI заключался в том, что адаптеры вычисляли строки изображения поочередно. Каждый ускоритель записывал свои строки в видеопамять и через свой цифроаналоговый преобразователь выводил их на монитор. Если компоненты видеокарт отличались по характеристикам, могли возникать ошибки в отображении картинки.

Длительный период времени видеокарте Voodoo2 Graphics в режиме SLI по производительности не было равных. Однако с появлением новых чипов nVidia RivaTNT и ATI Rage и шины AGP, а также объединением 2О-ускорителя и видеокарты в одном устройстве ускорители Voodoo2 Graphics стали пользоваться все меньшей популярностью и вскоре исчезли с рынка.

Технология параллельного использования двух видеоадаптеров была возрождена компанией nVidia после разработки шины PCI Express. Число разъемов с этой шиной на современных системных платах, как правило, превышает 2 (в отличие от шины AGP). nVidia, возрождая технологию SLI, решила не менять ее название. Изменилась только расшифровка аббревиатуры. Теперь SLI расшифровывается как Scalable Link Interface (масштабируемый соединительный интерфейс). Для функционирования технологии nVidia SLI необходимо наличие двух видеокарт с поддержкой SLI, системной платы с чипсетом, поддерживающим SLI, можно (но не обязательно) использовать переходную плату SLI-bridge для связи видеокарт и соответствующий драйвер.

Технология SLI поддерживает два режима работы пары видеокарт: Split Frame Rendering (SFR) и Alternate Frame Rendering (AFR). В режиме SFR происходит разделение кадра на две части, за рендеринг каждой из которых отвечает отдельный видеоадаптер. При этом кадр разделяется динамически в зависимости

от сложности сцены. Такой режим позволяет добиться максимальной производительности за счет равномерной загрузки каждой видеокарты. Метод разделения называется Symmetric Multi-Rendering with Dynamic Load Balancing (SMR), т. е. симметричный мультирендеринг с динамическим распределением нагрузки. В режиме AFR каждый адаптер обрабатывает свой кадр. ЦП перенаправляет ведущей карте запрос на обработку первого кадра и сразу же отправляет ведомой карте запрос на обработку второго кадра. Таким образом, одна видеокарта обрабатывает нечетные кадры, а вторая — четные.

Позднее компания nVidia на базе графического процессора nVidia GeForce 7900 GTX/GT (G71) разработала следующий вариант технологии SLI — для четырех чипов. Разумеется, такие решения существовали и раньше, например в промышленных симуляторах для обучения пилотов и в графических суперкомпьютерах. Однако эта технология впервые реализована на обычном PC на базе, вставляемой в один слот двухчиповой карты на базе GeForce от nVidia. Карта с двумя чипами сама по себе представляет SLI-решение. На системную плату, поддерживающую SLI, можно устанавливать две такие двухчиповые карты. Таким образом, компания nVidia предлагает два варианта — одиночное SLI-решение на двух чипах внутри одной карты и SLI-решение на четырех чипах (Quad-SLI путем установки второй подобной карты). Очевидно, что производительность четырехпроцессорной конфигурации будет высокой, а при некоторых условиях и очень высокой.

Выпуск подобной двухчиповой графической карты стал возможным только благодаря низкому энергопотреблению и тепловыделению, а также низкой себестоимости чипа. Карта состоит из базовой и дополнительной плат, занимает ширину двух слотов и при работе в режиме Quad-SLI связывается двумя каналами с соседней двухчиповой картой. Таким образом, получаем топологию квадрата — оба чипа карты связаны друг с другом и каждый связан с еще одним чипом соседней карты. Кроме того, на каждой двухчиповой карте установлен мост PCI-E x16, осуществляющий арбитраж и доступ к обоим ускорителям со стороны системы.

Режимы работы двух видеокарт аналогичны предыдущим. Есть три режима совместной работы: AFR (чередование расчета кадров между ускорителями), зональный рендеринг (разделение экрана на четыре зоны) и SLI-AA — использование ускорителей для расчета разных AA семплов в пределах одного пиксела. Кроме того, логичным становится комбинирование режимов, например, 2xAFR от двух двухзональных кадров (чередование кадров, каждый из которых построен SLI-методом разделения зон) или зональное разделение 2xSLI-AA и т. д. Комбинаций может быть много, никаких новых архитектурных изменений для этого не нужно, в технологии SLI уже заложены различные возможности, а сочетанием управляет драйвер.

Позднее аналогичная технология, получившая название CrossFire, была представлена и компанией ATI. В отличие от SLI, где обе видеокарты должны поддерживать технологию SLI, CrossFire позволяет объединять видеокарты, из которых только одна должна быть выполнена с поддержкой этой технологии,

вторая же может быть любой. Кроме того, в технологии CrossFire применяется внешнее соединение видеокарт при помощи специального кабеля.

Помимо метода, аналогичного SMR, технология CrossFire предполагает применение еще трех способов рендеринга: SuperTiling, в котором экран делится на множество квадратов и каждая карта обсчитывает половину этих квадратов; чередование кадров (Alternate frame rendering), при котором одна карта отвечает за рендеринг одного кадра, а другая — за рендеринг следующего кадра; и наконец, так называемый Super AA (суперсглаживание), обеспечивающий полное экранное сглаживание в режиме до 14х.

Основным достоинством технологий SLI и CrossFire является увеличение производительности в 3D-приложениях. Во многих играх увеличение производительности достигает 60...70 %. В тестовых приложениях это значение иногда доходит и до 100 %. Из недостатков рассматриваемых технологий следует выделить высокую стоимость видеосистемы, включая мощный блок питания (от 400 Вт) и системную плату на специальном чипсете.

Графические процессоры. Архитектура современных графических процессоров является многопроцессорной, она содержит набор относительно простых специализированных процессоров и АЛУ, ориентирована на полигональное представление трехмерной графики, используемой в современных компьютерных играх, и опирается на следующие основные свойства такого представления:

- значительная доля арифметических операций над векторными данными в графических алгоритмах с незначительным количеством логических операций;
- возможность эффективного распараллеливания процессов обработки графических объектов благодаря их взаимной независимости;
- потоковый характер построения изображения, что позволяет организовать графический конвейер.

Исходные данные, которыми оперируют современные графические процессоры (параметры вершин, матрицы преобразования, значения цвета и т. п.), организованы в виде векторов и соответственно большинство операций, выполняемых графическим процессором, являются векторными. Графические векторы, как правило, имеют четвертый порядок, т. е. содержат четыре числа, например три цветных компонента (R, G, B) и степень прозрачности (альфа-канал). Поэтому графические процессоры содержат векторные АЛУ, выполняющие операции с четырьмя компонентами того или иного формата, с произвольной перестановкой компонентов перед вычислениями, с возможностью выполнять разные операции по схеме 3+1 компонентов или даже менее часто востребованной, но все равно потенциально выгодной — 2+2 (две операции над двумя компонентами).

Операции со значениями цвета и прозрачности — чисто арифметические. Логически данные друг от друга не зависят, поэтому их можно обрабатывать параллельно. Для этого достаточно иметь одно векторное АЛУ и общий блок управляющей логики, обеспечивающий произвольную перестановку компонентов перед вычислениями. В реальных задачах часто встречаются ситуации, когда

необходимо обработать векторы второго порядка или скалярные величины (особенно это касается пиксельных конвейеров и пиксельных алгоритмов). В этом случае вычисления оптимизируются по схеме 2+2.

Особенность графических алгоритмов состоит в том, что объекты, обрабатываемые в графическом конвейере, как правило, не зависят друг от друга, при этом возможно распараллеливание обработки данных на нескольких уровнях. Например, при обработке вершин треугольника все три вершины будут обработаны по одному и тому же алгоритму, и более того, совершенно не важен порядок их обработки. Поэтому можно обрабатывать сразу несколько вершин параллельно, так как современные графические процессоры содержат группу вершинных процессоров (8 в nVidia GeForce 7800 GTX). Обработка пикселей еще лучше поддается распараллеливанию. Как следствие происходит рост числа пиксельных конвейеров в архитектуре графического процессора (48 в ATI RADEON X1900 XTX/XT (R580)), т. е. мощность графического процессора обычно наращивается путем увеличения числа вершинных и пиксельных конвейеров.

С внедрением DirectX 9.0c (пиксельные и вершинные шейдеры версии 3.0) в графических алгоритмах появилась возможность динамического управления вычислениями (циклы и ветвления, вызов подпрограмм, предикаты, возврат и другие операции). Характер и порядок обработки данных может зависеть от их исходных значений. Поэтому возникла необходимость создания полноценных параллельных конвейеров, каждый из которых оснащен управляющей логикой обработки динамических ветвлений. Все данные, поступающие в конвейеры графического процессора, организованы как однородные потоки данных. Следовательно, они могут быть предварительно подготовлены для обработки, т. е. выбраны из памяти, помещены в кэш-память и организованы в очереди. Первоначально эти возможности были реализованы в графических процессорах компании NVidia, а затем и в процессорах компании ATI.

Архитектура графического процессора (GPU). Структурная схема графического процессора GeForce 7800 GTX компании nVidia представлена на рис. 2.17.

ЦП компьютера, подготавливая полигональное описание трехмерной сцены, создает поток параметров вершин полигонов (треугольников), описывающих поверхности трехмерных объектов, присутствующих в кадре. Блок выборки геометрии графического процессора извлекает из ОП или видеопамяти геометрические данные и направляет их в кэш-память вершин. GPU поколения DirectX 9 поддерживают несколько вариантов структуры хранения геометрических данных в видеопамяти, учитывающих особенности полигональных моделей. Кроме того, аппаратно поддерживается гибкий формат данных: для каждой вершины могут храниться не только обычные параметры (координаты или вектор нормали), но и любые другие наборы допустимых скалярных или векторных величин.

Технологии памяти развиваются не так стремительно, однако потоковая природа графических алгоритмов позволяет обеспечить современные графические процессоры данными. В графический процессор поступает много разных данных, из него выходит только результирующее изображение. При этом все

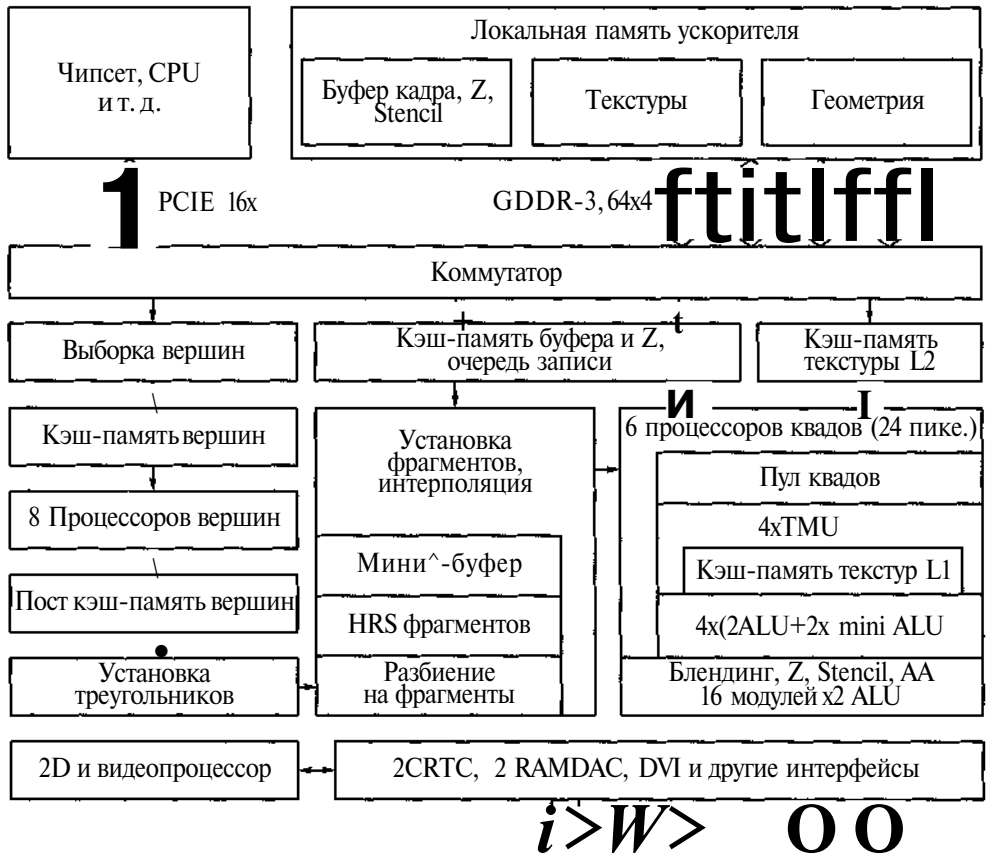


Рис. 2.17. Структурная схема графического процессора GeForce 7800 GTX компании nVidia

поступающие данные по своей сути — потоки, они считываются последовательно или почти последовательно. Таким образом, потоки данных могут быть кэшированы, выбраны предварительно, помещены в очереди, для того чтобы подсистема памяти не простаивала и работала эффективно. В современных графических процессорах практически нет произвольного доступа к памяти, что существенно снижает эффективность кэширования. Поэтому в отличие от обычных процессоров кэш-памяти GPU относительно малы, разделены и работают, как правило, только на чтение. Это позволяет сделать их особенно эффективными — даже кэш-память буфера кадра можно разбить на две части, одна из которых работает только на чтение, а вторая является обычной очередью записи. При этом поддерживается работа с несколькими потоками данных, когда часть атрибутов вершины хранится в одном массиве данных, а другая часть — в другом. Как правило, выборка из памяти происходит одновременно несколькими потоками (обычно по 4 каналам по 64 разряда в каждом). Далее каждая из вершин попадает в вершинный процессор.

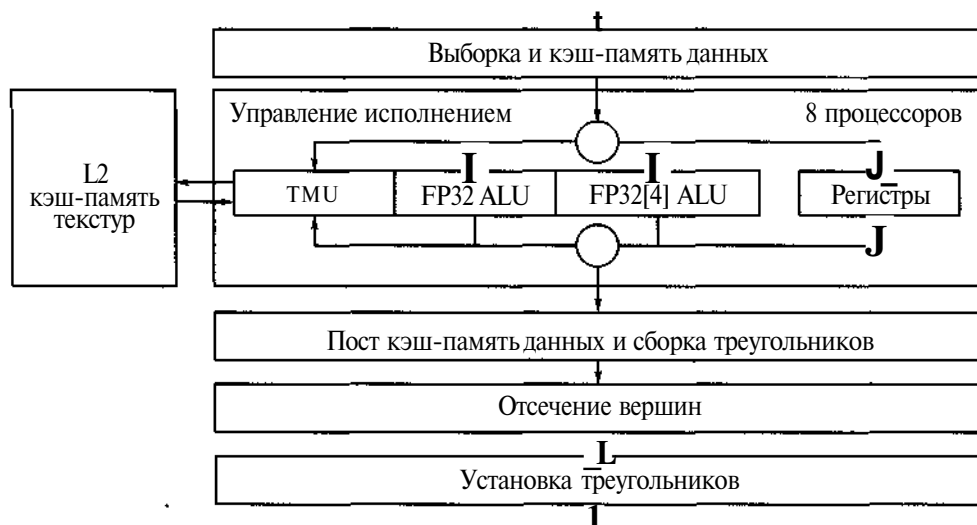


Рис. 2.18. Структурная схема вершинного процессора для GPU nVidia GeForce 7800 GTX

Вершинный процессор. В современных графических процессорах блоки обработки вершинных программ соответствуют спецификации Microsoft DirectX 9.0c, т. е. поддерживают вершинные шейдеры (Vertex Shader, VS) версии 3.0. Производительность и число вершинных блоков растут с развитием архитектуры графических процессоров. Структура вершинного процессора компании nVidia представлена на рис. 2.18.

Вершинные процессоры позволяют специальным программам (вершинным шейдерам) обрабатывать каждую вершину объекта, выполняя трансформации, расчет освещенности и другие операции. Собственно, графический процессор содержит восемь независимых вершинных процессоров, каждый из которых исполняет свои команды и имеет собственное устройство управления, т. е. разные процессоры могут одновременно исполнять различные участки программ над разными вершинами. За один такт вершинный процессор может выполнить одну векторную операцию (до четырех компонентов в формате с плавающей точкой FP32), одну скалярную операцию в формате FP32 и осуществить один доступ к текстуре (TMU). Поддерживаются целочисленные и плавающие форматы текстур и мип-мэппинг. Вершинные процессоры поддерживают статические ветвления, а также динамический контроль выполнения (Flow Control): циклы и ветвления, вызов подпрограмм, предикаты, возврат и другие операции.

Из вершинного процессора вершины, обработанные вершинным шейдером с учетом преобразований и освещения, передаются в небольшой промежуточный пост кэш-вершин. Далее осуществляется сборка примитивов, т. е. вершины группируются согласно с используемой топологией и отправляются в блок установки треугольников, где происходит предварительная подготовка данных, не-

обходимых для закраски всего треугольника. Здесь же производится отбрасывание невидимых и повернутых к наблюдателю обратной стороной треугольников.

Затем треугольник разбивается на фрагменты, часть которых признается невидимыми и отбрасывается в ходе предварительного теста глубины (Z-теста) на уровне фрагментов (технология удаления невидимых поверхностей HSR (Hidden Surface Remove)). Как правило, конечным результатом этого процесса являются видимые (или частично видимые) фрагменты размером 2x2 пиксела, называемые квадрами и подлежащие закрашке. Именно такие фрагменты наиболее удобны для быстрой закраски пикселей (по нескольким, в основном математическим причинам, связанным с интерполяцией текстурных координат).

В современных процессорах закрашка осуществляется в два этапа и на двух уровнях — блоков размером 4x4 пиксела (удобно для работы с буфером глубины — Z-буфером) и квадов размером 2x2 (квадами осуществляется закрашка в пиксельном процессоре). Вначале треугольник полностью покрывается блоками и, если даже один пиксел блока принадлежит треугольнику, он считается кандидатом на закрашку. Далее из рассмотрения удаляются все полностью невидимые блоки. Остальные блоки разбиваются на квады и для каждого из них вычисляются Z-координаты. Сравниваются значения глубины и отбрасываются полностью невидимые квады, а остальные отправляются на установку и закрашку в пиксельный процессор. При этом квады дополняются вычисленными значениями глубины и специальной битовой маской, определяющей видимые и невидимые пиксели квадра. В результате этого процесса в среднем половина пикселей будет отброшена еще до закрашки.

Затем квады отправляются на установку фрагментов. Здесь для каждого из них вычисляются необходимые далее параметры: текстурные координаты, МГР-уровень, векторы и установочные данные для анизотропной фильтрации и т. д. По мере роста сложности пиксельных шейдеров растет число передаваемых и интерполируемых для каждой точки параметров. Интерполяция требует сложных вычислительных операций, но использование квадов позволяет существенно повысить эффективность этого процесса за счет определения базовых параметров для одного пиксела, а для остальных — векторов преобразований. После установки и интерполяции параметров происходит закрашка фрагментов в пиксельном процессоре.

Пиксельный процессор. После расчета значений цвета в пиксельном процессоре происходит смешивание значений с уже существующими в буфере кадра или блендинга (если включен соответствующий режим) или просто запись результирующих значений цвета и глубины в буфер кадра. На этом этапе возможно выполнение дополнительных операций: гамма-коррекция, вычисление самого дальнего значения глубины блока 4x4 для пересылки в иерархический буфер глубины, сжатие Z-координат и т. д.

Структура пиксельного процессора компании nVidia представлена на рис. 2.19.

В пиксельном процессоре в обработке находится одновременно несколько сотен квадов. Для процессора квад представляет собой структуру данных, содержащую для каждого из четырех пикселей следующую информацию:

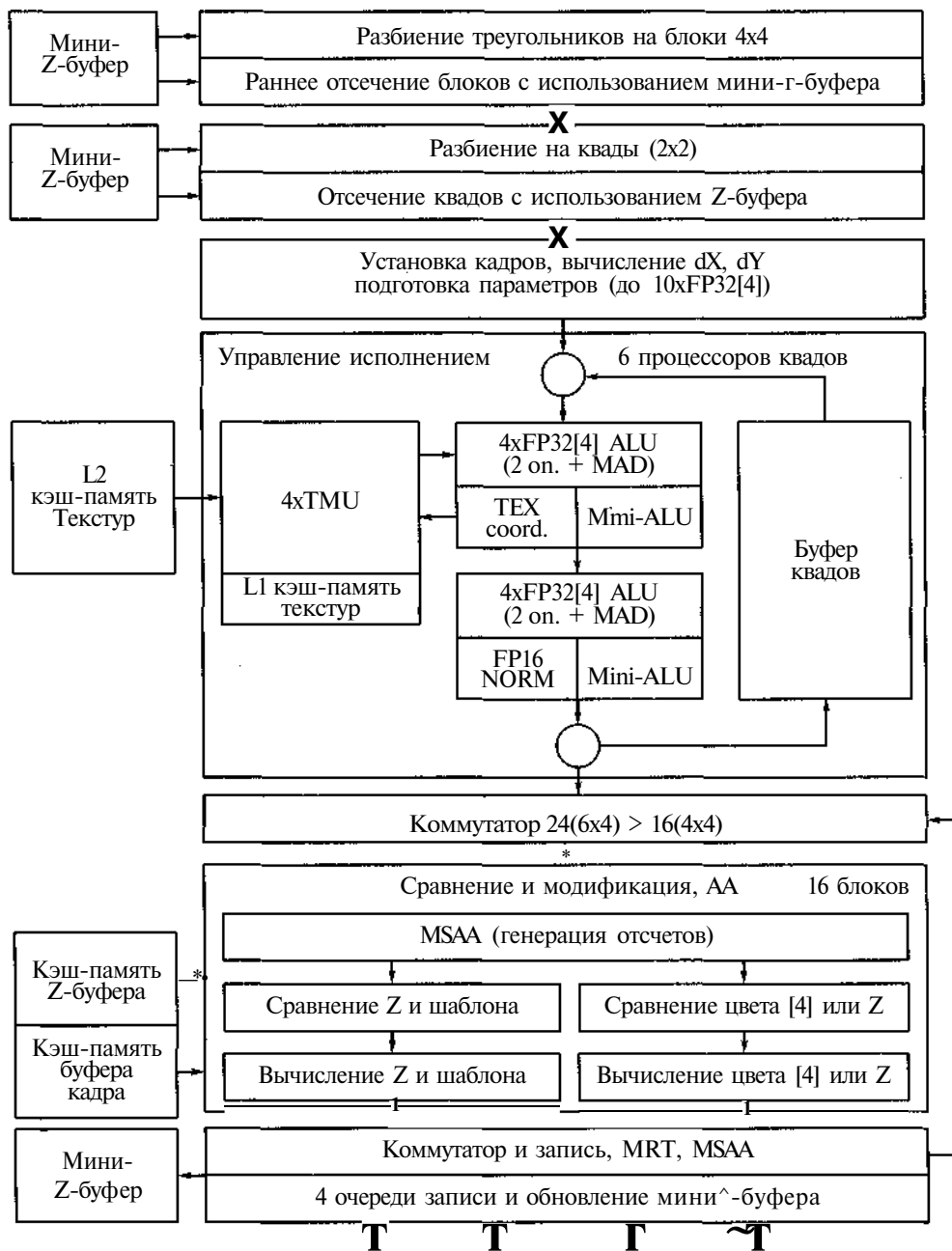


Рис. 2.19. Структурная схема пиксельного процессора для GPU nVidia GeForce 7800 GTX

- флаг активности данного пикселя (поскольку не все точки квада могут быть видимыми);
- флаг значения предиката для данного пикселя;
- значение Z и, возможно, значения буфера шаблонов;
- два векторных временных регистра FP32[4], которые могут быть разделены на четыре FP16[4].

Все обрабатываемые квады по очереди проходят через длинный пиксельный конвейер, состоящий из АЛУ, двух текстурных модулей и затем еще двух АЛУ. Длина конвейера составляет более двухсот тактов. Большая часть конвейера (около 170 тактов) приходится на операции выборки и фильтрации текстур, выполняемые в текстурных блоках. В нормальном режиме работы конвейер способен выдавать по кваду за один такт.

Всего в графическом процессоре содержится шесть независимых процессоров квадов. В каждом процессоре квадов содержится по два полных векторных ALU (способных выполнять две разные операции над четырьмя компонентами), дополненных двумя скалярными mini-ALU для параллельного исполнения простых операций. В результате ALU могут выполнять MAD-операции (одновременное умножение и сложение), часто встречающиеся в типовых пиксельных шейдерах. Кроме того, в каждом процессоре квадов содержится по четыре текстурных модуля (TMU) — по одному модулю на пиксел, собственная кэш-память первого уровня, фильтрация текстур с целочисленным или FP16 форматом компонентов, до четырех компонентов включительно (FP16[4]). TMU поддерживают анизотропную фильтрацию с соотношением сторон до 16:1 включительно (так называемые 16x) и позволяют выполнять все виды фильтрации с плавающими форматами текстур. За массивом из шести процессоров квадов следует коммутатор, который перераспределяет рассчитанные квады по 16 блокам генерации глубины, сглаживания (AA — Antialiasing) и блендинга.

Встроенный ускоритель графики. Большинство пользователей знает, что неотъемлемой частью компьютерной системы служит графический адаптер (видеоадаптер). Традиционно считается, что графический адаптер должен быть представлен отдельной видеокартой, занимающей специальный слот. Необходимо отметить, что видеокарта начального уровня (не самого последнего поколения) имеет определенную стоимость. Видеоадаптер, встроенный в чипсет системной платы, традиционно рассматривается как неполноценное решение, уступающее даже самой дешевой видеокарте.

Однако возможности современных встроенных ускорителей графики непрерывно растут. В первую очередь следует обратить внимание на качество 2Б-графики. Именно оно в значительной мере влияет на удовлетворенность пользователя видеоподсистемой. Ведь и ОС, и основная доля деловых приложений работают в 2О-режиме. Только игры и специальные программы трехмерного моделирования реально используют 3О-возможности видеокарты.

В современных чипсетах ситуация изменилась радикально. Например, чипсет Intel 945G содержит графическое ядро Intel Graphics Media Accelerator 950,

которое имеет RAMDAC с рабочей частотой 400 МГц. Чуть меньше (350 МГц) частота RAMDAC встроенного ускорителя чипсета nForce2 IGP компании nVidia. Эти значения полностью соответствуют требованиям к качественному воспроизведению 2D-графики с разрешением до 1920x1440 точек.

Таким образом, качество и производительность в 2D-приложениях у встроенных ускорителей и видеокарт практически идентичны. Этому в значительной мере способствовало внедрение новых типов оперативной памяти: DDR SDRAM и DDR2 SDRAM, поскольку встроенный ускоритель использует для своей работы часть ОП. При этом графический ускоритель даже в самых тяжелых условиях занимает около 10 % пропускной способности шины памяти, что практически незаметно при решении реальных задач.

Слабым местом встроенных ускорителей является работа со сложной 3D-графикой. Встроенное графическое ядро неплохо справится с простыми трехмерными задачами, например воспроизведением объектов VRML в Интернете. Имея чипсет с современным графическим ядром, можно запустить и сложную трехмерную игру, но обеспечить комфортный игровой процесс вряд ли будет возможно. Очевидно, что не следует решать на встроенном ускорителе задачи моделирования и рендеринга в 3D-Studio MAX. Встроенные графические ускорители обычно уступают типовым видеокартам в функциональности. Как правило, видеокарты оснащают выходом TV-out, цифровым интерфейсом DVI, выходом на второй монитор, а встроенное графическое ядро часто лишено таких функциональных элементов.

Встроенные графические ускорители современного уровня пока редки (Intel GMA 950, ATI RadeonXpress 200). Они могут работать с 2D-графикой без всяких конфликтов. Для трехмерных игр последнего поколения и сложных 3D-приложений необходимо устанавливать отдельную видеокарту. Функциональность встроенных ускорителей пока уступает типовым видеокартам. Области применения систем со встроенными графическими ускорителями: офисный компьютер, Интернет-станция, домашний мультимедийный компьютер с пониженными требованиями к комфортности игр, инженерные и научные расчеты, дупечатная подготовка и другие приложения с невысокими требованиями к 3D-графике.

Программные интерфейсы видеоадаптеров. Основой современных видеоадаптеров являются графические процессоры, которые производят разные фирмы, имеют различную архитектуру и системы команд. Для эффективного использования возможностей видеоадаптера необходимо наличие программного интерфейса между его программным обеспечением и системными и прикладными программами. Однако при широкой номенклатуре графических процессоров невозможно написать программу, которая бы одинаково эффективно работала с любой системой команд графического процессора. Поэтому для разработчиков программного обеспечения и создателей графических процессоров необходимо наличие специальных программных средств, обеспечивающих преобразование запросов программ, написанных в командах ЦП, в последовательность команд графического процессора и программную реализацию аппаратных средств, отсутствующих в графическом процессоре.

Основу современных средств интерфейса прикладного программирования API составляют специализированные прикладные программные библиотеки. Использование API позволяет разработчикам программ делать их универсальными, независимыми от низкоуровневых команд конкретного графического процессора. В настоящее время подавляющее большинство прикладных программ, работающих с трехмерными объектами, опираются на одну из двух типовых библиотек — OpenGL или DirectX.

OpenGL. Это графический стандарт в области компьютерной графики. На данный момент он является одним из самых популярных графических стандартов для разработчиков CAD-систем и научных приложений, кроме того, он широко используется и в других областях компьютерной графики, например в компьютерных играх. Еще в 1982 г. в Стенфордском университете была разработана концепция графической машины, на основе которой фирма Silicon Graphics в своей рабочей станции Silicon IRIS реализовала конвейер рендеринга. Таким образом была разработана графическая библиотека IRIS GL. На основе библиотеки IRIS GL в 1992 г. был разработан и утвержден графический стандарт OpenGL. Разработчиками OpenGL являются крупнейшие фирмы-разработчики как оборудования, так и программного обеспечения: Silicon Graphics, Inc., Microsoft, IBM Corporation, Sun Microsystems, Inc., Digital Equipment Corporation (DEC), Evans & Sutherland, Hewlett-Packard Corporation, Intel Corporation и Intergraph Corporation.

OpenGL на русский язык переводится как открытая графическая библиотека (Open Graphics Library), т. е. OpenGL — открытый и мобильный стандарт. Программы, написанные с помощью OpenGL, можно переносить практически на любые платформы, получая при этом одинаковый результат, будь то графическая станция или суперкомпьютер. Базовый набор OpenGL включает в себя около 150 различных команд, с помощью которых реализуют основные функции: определение объектов, указание их местоположения в трехмерном пространстве, установку других параметров (поворот, масштаб), изменение свойств объектов (цвет, текстура, материал), положение наблюдателя.

Несмотря на то что библиотека OpenGL предоставляет все основные возможности для моделирования и воспроизведения трехмерных сцен, некоторые из функций, которые требуются при работе с графикой, отсутствуют в стандартной библиотеке OpenGL. Поэтому для OpenGL были созданы так называемые вспомогательные библиотеки. Одна из вспомогательных библиотек под названием GLAUX была разработана фирмой Microsoft для ОС Windows.

Программы, написанные с помощью OpenGL, можно успешно перенести на такие платформы, как Unix, Linux, SunOS, IRIX, Windows, MacOS и многие другие. OpenGL развивается с помощью создания так называемых расширений — специальных модификаций базовой версии OpenGL, которые добавляют новые возможности и/или расширяют старые. Когда накапливается достаточно таких изменений, консорциум OpenGL выпускает спецификацию новой версии. Последней версией спецификации является версия 2.0.

DirectX. В 1995 г. компания Microsoft представила первую версию библиотеки DirectX (тогда она называлась Game SDK). В 2004 г. вышла девятая версия DirectX 9.0, а следом — модифицированная версия DirectX 9.0с. DirectX — корпоративный стандарт, все права на который принадлежат компании Microsoft. И только Microsoft определяет, что включать в очередную версию API, а какие предложения игнорировать. Более того, DirectX предназначен только для платформ Intel под управлением ОС Windows. В литературе иногда программы и видеоадаптеры разделяют на поколения согласно поддерживаемым версиям DirectX.

Стандарт DirectX включает в себя модули поддержки:

- программирования двумерной графики (модуль DirectDraw);
- создания трехмерной графики (модуль Direct3D);
- работы со звуками и музыкой (модули DirectSound и DirectMusic);
- поддержки устройств ввода (модуль DirectInput);
- разработки сетевых игр (модуль DirectPlay).

Таким образом, DirectX представляет собой набор из нескольких сравнительно независимых API, позволяющих разработчикам игр и других интерактивных приложений получать доступ к специфическим функциям аппаратного обеспечения без необходимости написания аппаратно-зависимого программного кода. Стандарт DirectX основан на наборе интерфейсов Component Object Model (компонентная модель объектов), а объекты COM могут описываться практически любыми языками программирования, например C/C++, Delphi и даже Basic. Популярность DirectX объясняется тем, что его использование обеспечивает потребности разработчиков игр и аппаратных средств: от создания трехмерной графики и пользовательского интерфейса ввода до поддержки сетевых виртуальных миров.

В начале 2007 г. выпущена новая ОС Microsoft Vista (старое название Longhorn). В отличие от Windows XP и Windows 2000, основанных на разных версиях одного ядра и имевших практически идентичную драйверную модель, Vista предлагает существенные изменения не только в интерфейсной части и API, но и в самом сердце системы, в ядре, архитектуре памяти и управлении ресурсами. Новая ОС будет поддерживать два типа драйверов (две драйверные модели), одну, оставленную для совместимости со старыми драйверами, — модель XP/2000 и вторую — новые драйверы, специально разработанные для Vista и последующих версий этой ОС. Эта новая модель (стандарт на драйверы и их взаимодействие с ядром OS и API) называется LDM (Longhorn Driver Model), важнейшей и наиболее интересной ее частью является отвечающая за всю графику LDDM (Longhorn Display Driver Model), т. е. модель дисплейного драйвера Vista. Все принципиально новые графические возможности будут реализовываться с помощью новых LDDM драйверов. Драйверы, построенные по старой модели, могут обеспечить только базовый, уже доступный в XP уровень аппаратной графической поддержки.

Новая драйверная модель предусматривает два уровня — базовый и продвинутый. Базовые драйверы могут быть написаны и для текущего аппаратного обеспечения, не требуя от аппаратуры каких либо функций, специально расчи-

тайных на новые возможности Vista. Эти драйверы будут обеспечивать все минимально необходимое для работы нового API — DX10 и новой драйверной модели, но не всегда оптимально с точки зрения производительности или удобства реализации. Продвинутые драйверы требуют специальной поддержки аппаратными средствами некоторых функций, в первую очередь связанных с управлением ресурсами, виртуальной памятью и конкурентными потоками заданий от приложений к устройствам, т. е. аппаратные средства должны разрабатываться с учетом спецификаций LDM и понимать некоторые специальные системные структуры данных. Тогда новые функции ядра и драйверной модели Vista будут выполняться наиболее быстрым, надежным и эффективным способом.

2.2.2. Технологии ЗБ-графики

ЗБ-конвейер. Современные графические процессоры для ПК работают с так называемой полигональной графической моделью, в которой любой трехмерный объект представляется как набор плоских многоугольников, рано или поздно разбивающихся на простейшие треугольники. Это связано с тем, что большинство алгоритмов закраски изображений требует, чтобы полигоны были плоскими, т. е. чтобы все их вершины лежали в одной плоскости, а треугольник — плоская фигура по определению. Объект задается вершинами, определяющими ключевые точки, и полигонами, которые образованы линиями, соединяющими вершины. Цвет полигонов формируется по специальным алгоритмам закраски, как правило, с наложением заранее нарисованных плоских изображений — *текстур*. Задача графического процессора сводится к тому, чтобы построить и закрасить как можно больше полигонов за единицу времени. В профессиональных ЗЭ-ускорителях иногда используется более совершенный способ затенения (рендеринга) трехмерных сцен — методом обратной трассировки лучей (Ray Tracing), требующим гораздо больших вычислительных ресурсов.

Для отображения треугольников используется метод так называемых однородных координат, опирающийся на матрицы преобразования и проецирования. Расчет положения любой точки трехмерной сцены на плоскости сводится к умножению вектора исходных координат на эти матрицы. Современный ЦП за секунду просчитывает координаты нескольких десятков миллионов вершин. Таким образом, затраты времени на расчет геометрии сцены не составляют для современного ЦП существенных значений. Проблемы начинаются при закраске полигонов с учетом освещенности и наложения текстур. Для сокращения числа закрашиваемых треугольников необходимо также решить проблему удаления невидимых поверхностей, т. е. разделить все полигоны на видимые, закрашиваемые и невидимые. Существуют чисто геометрические подходы к ее решению (например, тайловый подход, tiling), но чаще всего решение этой проблемы совмещают с закраской (техника буфера глубины, Z-buffer).

Z-buffer — часть графической памяти, в которой хранится относительная глубина каждого пиксела (значения Z). Z-buffer определяет, какая из многих пе-

рекрывающихся точек наиболее близка к плоскости наблюдения. Так же, как большее число битов на пиксел для цвета в буфере кадра соответствует большому количеству цветов, доступных в системе изображения, так и количество бит на пиксел в Z-буфере соответствует большому числу элементов. Обычно Z-буфер имеет не менее 16 бит на пиксел для представления информации о глубине.

Реалистичность изображения в трехмерной сцене во многом определяется качеством *текстур* — заранее нарисованных изображений, накладываемых на полигоны. Двухмерные изображения текстур хранятся в памяти компьютера или графического адаптера в одном из пиксельных форматов. Текстуры могут храниться в сжатом виде на дисках компьютера, перед использованием разворачиваются в памяти и могут занимать объем в десятки раз больше первоначального размера. Для каждой вершины указываются ее координаты в плоскости изображения двухмерной текстуры (используются также одномерные и трехмерные текстуры). При расчете цвета конкретной точки полигона учитывается ее расположение относительно вершин треугольника, и точке присваивается цвет, аналогичный цвету соответствующей точки текстуры. Для этого каждой точке экрана, попавшей в треугольник, нужно найти соответствие в текстурных координатах (это достаточно сложный процесс) и провести так называемую выборку из текстуры — вычислить цвет текстуры в полученной точке. Эта задача достаточно сложна, поскольку часто расчетная точка попадает между пикселями изображения текстуры. Поэтому расчеты получаются очень трудоемкими, особенно с учетом наложения нескольких текстур. К тому же текстуры в современных играх часто представлены изображениями высокого разрешения. В итоге достаточно мощный ЦП даже при низком экранном разрешении способен обработать не более десятка кадров в секунду. Поэтому ускорители трехмерной графики в первую очередь были созданы для аппаратного ускорения закраски изображений.

Вычисление цвета точки, попавшей между известными точками текстуры, называется *фильтрацией*. Если используется простейший способ — выбрать точку текстуры, ближайшую к расчетной, — то реализуется поточечная фильтрация. Если вычисляется взвешенное среднее арифметическое значение для четырех ближайших соседних точек — это билинейная фильтрация. Используется также трилинейная фильтрация, когда значение цвета точки вычисляется по восьми соседним точкам. Самым совершенным способом фильтрации является анизотропная (неоднородная по разным направлениям), в которой учитываются и «физические размеры» пиксела, т. е. считается не просто проекция центра пиксела экрана на текстуру, а пиксела целиком. Чем выше кратность анизотропной фильтрации, тем проекция точнее. Использование простых способов фильтрации приводит к тому, что текстуры в определенных ситуациях размываются, теряют четкость. И, напротив, чем качественнее фильтрация, тем более четко прорисовываются объекты.

Объекты в сцене даже с наложенными текстурами выглядят неестественно, если не учитывается их освещенность и различные оптические эффекты. Первые

видеоадаптеры не могли рассчитывать освещенность объектов. Специальные программные модули, используя ЦП, предварительно рассчитывали освещенность элементов сцены на основе одного из методов затенения, или рендеринга, специально разработанных для визуализации трехмерных сцен (см. разд. 4.2). В результате создавались карты освещенности (light maps), которые отображались как соответствующие текстуры на изображении трехмерной сцены. Качество изображения в этом случае было достаточно высоким, однако только для неподвижных объектов и при статических источниках освещения, что не годится для современных игр, отличающихся высокой динамикой. Чтобы снизить нагрузку на ЦП в видеоадаптеры (по мере роста производительности и сложности), было включено устройство геометрических вычислений — так называемый блок Transform & Lightning (T&L). Однако требования к реалистичности сцены непрерывно росли, а метод интерполяции по всему полигону был не в состоянии правдоподобно передать фактуру поверхности. Кроме того, большинство трехмерных объектов строилось из малого числа полигонов с целью экономии вычислительных ресурсов, и поэтому качество изображения было низким вследствие прямолинейности образующих полигоны линий и плоских граней.

Эту проблему решили, применив метод расчета освещенности с учетом вектора нормали к поверхности, т. е. вектора, перпендикулярного к поверхности в данной точке. Задав еще одну текстуру специального вида (карту нормалей) и модифицировав алгоритм расчета цвета точки, можно радикально улучшить внешний вид моделей. Подобный метод называют Bump Mapping (отображение выпуклостей или шероховатостей). Существует несколько общих типов Bump Mapping: Emboss Bump Mapping, Dot3 Bump Mapping, Environment Mapped Bump Mapping (EMBM) и True, Reflective Bump Mapping. Самая эффективная методика из перечисленных использует Dot3 Bump Mapping. Для полноценной реализации Bump Mapping требует программирования пиксельных конвейеров — перехода от интерполяции и выборки из текстур к вычислениям по формулам определения цвета каждого пикселя объекта (а не только вершин). Так впервые появились пиксельные шейдеры — последовательности кодов графического процессора, позволяющие программировать его пиксельные конвейеры. Шейдеры заметно повысили реалистичность объектов.

Шейдеры. Шейдером в широком смысле называется программа для визуального определения поверхности объекта. Это может быть описание освещения, текстурирования, постобработки и т. п. Программируемые шейдеры были впервые представлены в RenderMan компании Pixar, там определены несколько типов шейдеров: light source shaders, surface shaders, displacement shaders, volume shaders, imager shaders. Эти шейдеры чаще всего программно выполняются ЦП и не имеют полной аппаратной реализации. Затем шейдеры были адаптированы для графических приложений реального времени. Шейдеры для первых графических адаптеров писались на так называемом assembly shader language, который близок к ассемблеру для ЦП, но не привязан к конкретному GPU. Его низкий уровень доставляет определенные сложности для понимания кода и программи-

рования. В настоящее время используются специальные языки программирования высокого уровня для создания шейдеров. Компания nVidia давно предлагает С-компилятор шейдеров — Cg, а Microsoft включила в состав DirectX 9.0c стандартный язык — High-Level Shader Language (HLSL).

В целом шейдеры добавили к графическому конвейеру множество новых возможностей по трансформации и освещению вершин и индивидуальной обработке пикселей, необходимых для разработчиков конкретных приложений. И все-таки, возможности аппаратных шейдеров до сих пор не раскрыты полностью и, по-видимому, с ростом их функциональностей в каждом новом поколении графических ускорителей скоро будет достигнут уровень тех самых шейдеров RenderMan, которые когда-то казались недостижимыми для игровых видеоускорителей. Пока в шейдерных моделях реального времени (в определении DirectX 9 API), поддерживаемых на сегодняшний день видеоадаптерами, определено лишь два типа шейдеров: Vertex Shader (вершинный шейдер) и Pixel Shader (пиксельный шейдер). В API DirectX 10 для ОС Vista к ним добавлен Geometry Shader (геометрический шейдер).

Вершинные шейдеры — программы, исполняемые видеопроцессорами, которые производят математические операции с вершинами полигонов, относящихся к полигональной модели 3D-объекта, иначе говоря, они предоставляют возможность выполнять программируемые алгоритмы по изменению параметров вершин и их освещению (естественное развитие блока T&L). Каждая вершина определяется набором атрибутов: координаты, цвет, нормаль, текстурные координаты и т. п. В процессе работы вершинные шейдеры в зависимости от алгоритмов изменяют эти данные, например вычисляя и записывая новые координаты и/или цвет, т. е. входными данными вершинного шейдера является информация об одной вершине геометрической модели, обрабатываемой в данный момент. Обычно это пространственные координаты, нормаль, компоненты цвета и текстурные координаты. Результаты выполнения шейдера служат входными для дальнейшей части конвейера, затем растеризатор делает линейную интерполяцию входных данных для поверхности треугольника и для каждого пикселя исполняет соответствующий пиксельный шейдер.

До появления современных видеопроцессоров программисты либо использовали собственные программы и алгоритмы, изменяющие параметры вершин, которые исполнялись ЦП (программный T&L), либо фиксированные алгоритмы в видеопроцессорах с аппаратной реализацией трансформации и освещения (аппаратный T&L). Первая же шейдерная модель DirectX означала большой шаг вперед от фиксированных функций по трансформации и освещению вершин к полностью программируемым алгоритмам. В современных видеопроцессорах с вершинами при помощи вершинных шейдеров можно делать очень много сложных геометрических преобразований. Особое внимание вершинным шейдерам уделяется при использовании профессиональных видеокарт.

Возможности шейдеров росли от версии к версии. Начиная с версии 3.0, (включены в API DirectX 9c) шейдеры почти не ограничивают методы программирования, позволяя использовать в вычислениях текстуры и задавать уникаль-

ные свойства вершин, т. е. вершинный блок графического процессора сейчас мало чем уступает по функциональности ЦП.

Пиксельные шейдеры — программы расчета значений цвета и яркости отдельных пикселей. Задачи вершинных шейдеров при необходимости можно решать на ЦП. С геометрическими расчетами мощный процессор справится без проблем. Но при выполнении пиксельного шейдера даже сверхмощный CPU задумается надолго, поскольку здесь происходит интенсивный обмен с памятью и сложные расчеты с использованием арифметики с плавающей точкой. Пиксельные шейдеры — это программы, выполняемые после растеризации для каждого пиксела изображения. Они производят выборку из текстур и/или математические операции над цветом и значением глубины (Z-buffer) пикселей. Все инструкции пиксельного шейдера выполняются попиксельно после того, как операции с трансформированием и освещением геометрии завершены. Пиксельный шейдер в итоге своей работы выдает конечное значение цвета пиксела и Z-значение для последующего этапа графического конвейера, блендинга. Наиболее простой пример пиксельного шейдера — мультитекстурирование, являющееся смешением нескольких текстур (diffuse и lightmap, например), и наложение результата вычислений на фрагмент.

До появления видеопроцессоров с аппаратной поддержкой пиксельных шейдеров у разработчиков были лишь возможности по обычному мультитекстурированию и альфа-блендингу, что существенно ограничивало возможности по многим визуальным эффектам и не позволяло делать многое из того, что сейчас доступно. И если с геометрией еще что-то можно было сделать программно, то с пикселями — нет. Ранние версии DirectX (до 7.0 включительно) были ориентированы на повершинные расчеты и предлагали крайне ограниченную функциональность попиксельных расчетов. Пиксельные шейдеры сделали возможным расчет освещения любых поверхностей попиксельно, используя заданные разработчиками материалы. Пиксельные шейдеры версии 1.1 (в понимании DirectX) уже могли не только делать мультитекстурирование, но и многое другое. Однако большинство игр, использующих шейдеры версии SM1, просто использовали традиционное мультитекстурирование на большинстве поверхностей, выполняя более сложные пиксельные шейдеры лишь на части поверхностей, для создания разнообразных спецэффектов (общеизвестно, что вода до сих пор является наиболее частым примером использования пиксельных шейдеров в играх). Обработкой пиксельных шейдеров практически всегда занимаются пиксельные конвейеры графического процессора. Использование пиксельных шейдеров позволило реализовать расчет освещенности объектов методом Фонга, обеспечивающим достаточно высокую реалистичность. С появлением шейдеров версии SM3 и поддерживающих их видеопроцессоров возможности пиксельных шейдеров возросли уже до того, чтобы с их помощью делать даже трассировку лучей (raytracing), пусть и с некоторыми ограничениями.

Технологии наложения и обработки текстур. Современные приложения трехмерной графики позволяют создавать для различных программ изображения

объектов с высокой степенью реалистичности. Большую роль в процессе получения изображений с высоким уровнем реалистичности играет наложение текстур на трехмерные модели объектов. Посредством наложения текстур отображаются не только свойства поверхности объекта, но и учитываются различные оптические эффекты, позволяющие значительно повысить реалистичность изображения.

Texture Mapping (наложение текстур). Традиционно термином «texture mapping» в трехмерном моделировании называют процесс наложения изображения текстуры на объект (текстура как бы натягивается на объект) для придания ему соответствующего внешнего вида. Наложение текстур без каких-либо эффектов отличается простотой реализации, обработка не требует значительных вычислительных мощностей, возможно использование и отображение практически любых изображений.

MIP mapping (MIP-текстурирование). Multum in Parvo (с лат. — много в одном) — метод улучшения качества текстурных изображений при помощи текстур с разным разрешением для различных объектов одного и того же изображения в зависимости от их размера и глубины. Таким образом, в памяти хранятся несколько копий изображения текстуры в различных разрешениях. В результате этого изображение остается качественным при приближении к объекту и при удалении от него. При использовании этого метода формируется изображение высокого разрешения для близко расположенных объектов и изображение низкого разрешения при удалении объектов. MIP mapping снижает мерцание и «зашумленность» изображения, возникающие при texture mapping, повышает эффективность использования текстур.

Суть MIP-текстурирования заключается в предварительном или динамическом создании набора текстур с различным разрешением и уровнем детализации на основе базовой текстуры максимального разрешения. Цвет каждого текселя (элемент текстуры — определенный пиксел в текстуре) текстуры с более низкой детализацией выбирается путем аппроксимации цвета четырех ближайших текселей текстуры предыдущего (более высокого) уровня детализации. При построении трехмерной сцены определяется удаленность полигона от картинной плоскости и соответственно этому значению накладывается текстура с заданным разрешением. В настоящее время для выбора конкретного уровня детализации используют в основном динамический расчет.

Расчет уровня MIP можно вести либо для полигона в целом (по усредненному значению для трех вершин), либо для каждого пикселя. Последний метод дает более качественный результат и позволяет избежать некоторых дефектов изображения, но потребляет много вычислительных ресурсов. Кроме того, в зависимости от вычисленного уровня к текстурам применяют различные фильтры, например для сглаживания ребер.

Procedural Textures (процедурные текстуры). Эти текстуры создаются на основе аналитической математической модели. Такие текстуры не занимают в видеопамати места, они генерируются пиксельным шейдером «на лету», каждый

их элемент (тексель) получается в результате исполнения соответствующих команд шейдера. Наиболее часто встречающиеся процедурные текстуры: разные виды шума (например, fractal noise), дерево, вода, лава, дым, мрамор, огонь, т. е. те, которые сравнительно просто можно описать математически. Процедурные текстуры также позволяют использовать анимированные текстуры при помощи всего лишь небольшой модификации математических формул. Например, изображения облаков, созданные подобным образом, выглядят вполне правдоподобно и в динамике, и в статике.

Преимущества процедурных текстур также включают в себя неограниченный уровень детализации каждой текстуры. При этом не нужна пикселизация, текстура всегда генерируется под необходимый для ее отображения размер. Большой интерес представляет анимированный Normal Mapping, с его помощью можно сделать волны на воде без применения предварительно просчитанных анимированных текстур. Еще одно достоинство таких текстур состоит в том, что чем больше их применяется в продукте, тем меньше работы для художников (правда, больше для программистов) над созданием обычных текстур.

Bump Mapping/Specular Bump Mapping (бамп мэппинг). Это техника моделирования неровностей или микрорельефа на плоской поверхности без больших вычислительных затрат и изменения геометрии. Для каждого пиксела поверхности выполняется вычисление освещенности, исходя из значений в специальной карте нормалей, называемой normal map. Это обычно 8-битная черно-белая текстура и значения цвета текстуры не накладываются как обычные текстуры, а используются для описания неровности поверхности. Значение цвета каждого текселя определяет высоту соответствующей точки рельефа, большие значения означают большую высоту над исходной поверхностью, или наоборот.

Степень освещенности точки зависит от угла падения лучей света. Чем меньше угол между нормалью и лучом света, тем больше освещенность точки поверхности, т. е. если взять ровную поверхность, то нормали в каждой ее точке будут одинаковыми и освещенность также будет одинаковой. Если поверхность неровная, то нормали в каждой точке будут разными и освещенность будет различная — в одной точке она будет больше, в другой меньше. Отсюда и принцип bump mapping — для моделирования неровностей в разных точках полигона задаются нормали к поверхности, которые учитываются при попиксельном вычислении освещенности. В результате получается более натуральное изображение поверхности, bump mapping придает поверхности большую детализацию, такую, как неровности на кирпиче, поры на коже и на других поверхностях без увеличения геометрической сложности модели, так как расчеты ведутся на пиксельном уровне. Причем при изменении положения источника света корректно изменяется и освещенность этих неровностей.

Конечно, расчет освещенности в вершинах намного проще с точки зрения вычислений, но слишком нереалистично это выглядит, особенно при малом числе полигонов в геометрии. Интерполяция цвета для каждого пиксела не может воспроизвести большие, чем рассчитанные значения для вершин, т. е. пиксели в

середине треугольника не могут быть ярче, чем фрагменты возле вершины. Следовательно, области с резким изменением освещения, такие как блики и источники света, очень близко расположенные к поверхности, будут отображаться неправильно, что особенно заметно в динамике. Частично проблема может быть решена путем увеличения геометрической сложности модели, ее разбиением на большее количество вершин и треугольников, но оптимальным вариантом является попиксельный расчет освещенности.

Необходимо также напомнить о составляющих освещенности. Цвет точки поверхности рассчитывается как сумма *ambient*, *diffuse* и *specular* составляющих от всех источников света в сцене. Вклад в это значение каждого источника света зависит от расстояния между источником света и точкой на поверхности.

Равномерная (*ambient*) составляющая освещения — аппроксимация глобального освещения, фоновое освещение для каждой точки сцены, при котором все точки освещаются одинаково и освещенность не зависит от других факторов.

Рассеянная, или диффузная (*diffuse*), составляющая освещения зависит от положения источника освещения и от нормали поверхности. Эта составляющая освещения разная для каждой вершины объекта, что придает им объем. Свет уже не заполняет поверхность ровным, одинаковым оттенком.

Бликовая (*specular*) составляющая освещения проявляется в бликах отражения лучей света от поверхности. Для ее расчета помимо вектора положения источника света и нормали необходимы еще два вектора: вектор направления взгляда и вектор отражения. Бликовая модель освещения впервые была предложена Фонгом (Phong Bui-Tong). Блики существенно увеличивают реалистичность изображения, поскольку практически все реальные поверхности в той или иной степени отражают свет, поэтому бликовая составляющая очень важна, особенно при моделировании динамики, так как по бликам сразу видно изменение положения камеры или самого объекта. Существуют другие более сложные способы вычисления этой составляющей, учитывающие распределение энергии света, его поглощение материалами и рассеивание в виде диффузной составляющей.

В ранних видеокартах (на базе чипов nVidia Riva TNT) использовались некоторые виды *bump mapping* (*Emboss Bump Mapping*), однако методы их реализации в то время были крайне примитивны и широкого применения не получили. Следующим известным видом был *Environment Mapped Bump Mapping* (EMBM), затем появился *Normal Mapping*, видеопроцессоры того времени требовали три прохода для реализации алгоритма, так как они были ограничены одновременным использованием только двух текстур. Начиная с версии процессора NV20 (GeForce3) появилась возможность выполнять алгоритм за один проход, используя пиксельные шейдеры. Позднее стали применять более эффективные методы.

Normal Mapping (нормалмэппинг). Это улучшенная разновидность техники *bump mapping*, описанной выше, ее расширенная версия. *Bump mapping* был раз-

работай Блинном (Blinn) еще в 1978 г. Нормали поверхности при этом методе наложения рельефа изменяются на основе информации из карты нормалей (normal map). В то время как bump mapping всего лишь изменяет существующую нормаль для точек поверхности, normal mapping полностью заменяет нормали путем выборки их значений из специально подготовленной карты нормалей (normal map). Эти карты обычно являются текстурами с сохраненными в них, заранее просчитанными значениями нормалей, представленными в виде компонентов цвета RGB (существуют и специальные форматы для карт нормалей, в том числе со сжатием), в отличие от 8-битных черно-белых карт высот в bump mapping.

Как и bump mapping, normal mapping является достаточно простым методом добавления детализации к моделям сравнительно низкой геометрической сложности без использования сложной реальной геометрии, только более продвинутый. Одно из наиболее интересных применений этого метода — существенное увеличение детализации моделей с малым числом полигонов при помощи карт нормалей, полученных обработкой такой же модели высокой геометрической сложности. Карты нормалей содержат более подробное описание поверхности по сравнению с bump mapping и позволяют представлять более сложные формы.

Карты нормалей по сравнению с простым использованием большого количества полигонов представляют более эффективный способ для хранения подробных данных о поверхностях. Однако их существенным ограничением является то, что они не очень хорошо подходят для крупных деталей, поскольку normal mapping на самом деле не добавляет полигонов и не изменяет форму объекта, он только создает видимость этого. Это всего лишь моделирование деталей на основе расчета освещенности на пиксельном уровне. На крайних полигонах объекта и при больших углах наклона поверхности это очень хорошо заметно. Поэтому наиболее целесообразно применение normal mapping для создания модели с малым числом полигонов, достаточно детализированной для того, чтобы сохранялась основная форма объекта, и использование карты нормалей для добавления более мелких деталей.

Карты нормалей обычно создаются на основе двух версий модели: с малым и большим числом полигонов. Модель с малым числом полигонов состоит из основных форм объекта и имеет простое геометрическое описание, а модель с большим числом полигонов содержит все необходимое для максимальной детализации. Затем при помощи специальных утилит карты сравниваются друг с другом, рассчитывается разница и сохраняется в текстуре, называемой картой нормалей. При ее создании дополнительно можно использовать и bump map для очень мелких деталей, которые нельзя смоделировать даже в модели с большим числом полигонов (поры кожи, различные мелкие углубления и т. п.).

Карты нормалей представляются в виде обычных RGB текстур, где компоненты цвета R, G и B (от 0 до 1) интерпретируются как координаты X, Y и Z. Каждый тексель в карте нормалей представляется как нормаль точки поверхности. Карты нормалей бывают двух видов: с координатами в глобальной системе ко-

ординат или в локальной системе координат треугольника. Чаще применяется второй вид карт.

Современные приложения реального времени часто проигрывают предварительно просчитанной анимации по качеству изображения. Это касается прежде всего качества моделирования освещения и геометрической сложности сцен. Количество вершин и треугольников, рассчитываемых в реальном времени, ограничено, поэтому очень важны методы, позволяющие снизить сложность геометрии. До normal mapping было разработано несколько таких методов, но модели с малым числом полигонов, даже при использовании bump mapping, изображаются хуже более сложных моделей. Normal mapping, несмотря на присущие ему недостатки, заметно улучшает итоговое качество рендеринга, сохраняя низкую геометрическую сложность моделей. Сейчас данный метод весьма популярен и используется во многих компьютерных играх благодаря высокому результирующему качеству изображения и снижению требований к геометрической сложности моделей.

У метода normal mapping есть один недостаток — увеличение сложности текстур. Карта нормалей сильно влияет на то, как будет выглядеть объект, и она должна быть достаточно высокого разрешения, поэтому требования к видеопамяти и ее пропускной способности удваиваются (в случае несжатых карт нормалей). Однако пропускная способность видеопамати и ее объем непрерывно растут, разработаны методы сжатия специально для карт нормалей, поэтому эти ограничения практически не являются существенными. Гораздо важнее эффект, который дает метод normal mapping, позволяя использовать модели со сравнительно малым числом полигонов, снижая требования к памяти для хранения геометрических данных и повышая производительность.

Displacement Mapping (карта смещения). Наложение карт смещения является методом добавления деталей к трехмерным объектам. В bump mapping и других попиксельных методах картами высот корректно моделируется только освещенность точки, но не изменяется ее положение в пространстве. Это дает лишь иллюзию увеличения сложности поверхности, а карты смещения позволяют получить сложные 3D-объекты из вершин и полигонов без ограничений, присущих попиксельным методам. Этот метод изменяет положение вершин треугольников, сдвигая их по нормали на величину, исходя из значений в картах смещения.

Карта смещения — обычно черно-белая текстура, и значения в ней используются для определения высоты каждой точки поверхности объекта подобно bump map. Часто карты смещения применяются (в этом случае они называются и картами высот) для создания земной поверхности с холмами и впадинами. Поскольку рельеф местности описывается двухмерной картой смещения, его относительно легко деформировать при необходимости, так как это потребует всего лишь модификации карты смещения и рендеринга поверхности на ее основе в следующем кадре.

Большим преимуществом наложения карт смещения является не просто возможность добавления деталей к поверхности, а практически полное создание

объекта. При этом модель объекта с малым числом полигонов разбивается (тесселируется) на большее количество вершин и полигонов. Вершины, полученные в результате тесселяции, затем смещаются по нормали, исходя из значения, прочитанного в карте смещения. В итоге из простого объекта, используя соответствующую карту смещения, получаем сложный ЗБ-объект. Число треугольников, созданных при тесселяции, должно быть достаточно большим для того, чтобы передать все детали, задаваемые картой смещений. Иногда дополнительные треугольники создаются автоматически, используя N-патчи или другие методы. Карты смещения лучше использовать совместно с методом bump mapping для создания мелких деталей, где достаточно правильного попиксельного расчета освещенности.

Наложение карт смещения впервые появилось в DirectX 9.0. Это была первая версия данного API, которая поддерживала метод Displacement Mapping. В DX9 поддерживаются два типа наложения карт смещения: filtered и presampled. Filtered метод отличается тем, что позволяет использовать мип-уровни для карт смещения и применять для них трилинейную фильтрацию. В этом методе мип-уровень карты смещения выбирается для каждой вершины на основе расстояния от вершины до камеры, т. е. уровень детализации выбирается автоматически. Таким образом достигается почти равномерное разбиение сцены, когда треугольники имеют примерно одинаковый размер.

Наложение карт смещения можно, по существу, считать методом сжатия геометрии. Использование карт смещения снижает объем памяти, требуемый для определенной детализации 3D-модели. Громоздкие геометрические данные замещаются простыми двухмерными текстурами смещения, обычно 8-битными или 16-битными. Это снижает требования к объему памяти и пропускной способности, необходимой для доставки геометрических данных видеопроцессору, а эти ограничения являются одними из основных для современных систем. Или же при равных требованиях к пропускной способности и объему памяти наложение карт смещения позволяет отображать намного более сложные геометрические 3D-модели. Применение моделей меньшей сложности, когда вместо сотен тысяч или миллионов треугольников используют десятки тысяч, позволяет ускорить их анимацию или улучшить, применив более сложные комплексные алгоритмы и методы вроде имитации тканей (cloth simulation).

Другое преимущество карт смещения состоит в том, что их применение превращает сложные полигональные трехмерные сетки в несколько двухмерных текстур, которые проще поддаются обработке. Кроме того, вместо сравнительно сложных алгоритмов сжатия трехмерных сеток можно применять обычные методы сжатия текстур, даже подобные JPEG. А для процедурного создания моделей 3D-объектов можно использовать обычные алгоритмы для двухмерных текстур.

Однако карты смещения в некоторых ситуациях использовать нельзя. Например, гладкие объекты, не содержащие большого количества тонких деталей, будут лучше представлены в виде стандартных полигональных сеток или иных поверхностей более высокого уровня. Кроме того, очень сложные модели, такие

как деревья или растения, также нелегко представить картами смещения. Существуют также проблемы их применения — это почти всегда требует специализированных утилит, поскольку достаточно сложно непосредственно создавать карты смещения (за исключением простых объектов, вроде ландшафта). Многие проблемы и ограничения, присущие картам смещения, аналогичны проблемам наложения карт нормалей, поскольку эти два метода представляют собой два разных подхода к реализации похожей идеи.

Parallax Mapping/Offset Mapping. После метода normal mapping, представленного в 1984 г., был разработан метод рельефного текстурирования (Relief Texture Mapping), представленный Olivera и Bishop в 1999 г. Этот метод для наложения текстур основан на информации о глубине. Метод не нашел применения в играх, но его идея способствовала продолжению работ над методом parallax mapping и его улучшением. Kaneko в 2001 г. представил метод parallax mapping, который стал первым эффективным методом для попиксельного отображения эффекта параллакса. В 2004 г. Welsh продемонстрировал применение метода parallax mapping на программируемых видеопроцессорах. У этого метода много различных названий: Parallax Mapping, Offset Mapping, Virtual Displacement Mapping, Per-Pixel Displacement Mapping. Далее применяется первое и основное название.

Parallax Mapping — одна из альтернатив методам bump mapping и normal mapping, которая дает еще большее представление о деталях поверхности, более натуралистичное отображение 3D-поверхностей также без слишком больших потерь производительности. Этот метод похож одновременно на наложение карт смещения и normal mapping (нечто среднее между ними). Метод также предназначен для отображения большего количества деталей поверхности, чем имеется в исходной геометрической модели. Отличие состоит лишь в том, что данный метод искажает наложение текстуры, изменяя текстурные координаты так, что поверхность выглядит выпуклой, хотя в реальности она плоская и не изменяется. Иными словами, метод parallax mapping — метод аппроксимации эффекта смещения точек поверхности в зависимости от изменения точки зрения.

В этом методе сдвигаются текстурные координаты (поэтому его иногда называют offset mapping) так, чтобы поверхность выглядела более объемной. Идея метода состоит в том, чтобы возвращать текстурные координаты той точки, где видовой вектор пересекает поверхность. Это требует просчета лучей (рейтрейсинг) для карты высот, но если она не имеет сильно изменяющихся значений (гладкая или плавная), то можно обойтись аппроксимацией. Данный метод эффективен для поверхностей с плавно изменяющимися высотами, без просчета пересечений и больших значений смещения. Подобный простой алгоритм отличается от normal mapping всего тремя инструкциями пиксельного шейдера: две математические инструкции и одна дополнительная выборка из текстуры. После того, как вычислена новая текстурная координата, она используется дальше для чтения других текстурных слоев: базовой текстуры, карты нормалей и т. п. Метод parallax mapping на современных видеопроцессорах почти также эффекти-

вен, как обычное наложение текстур, а в результате получается более реалистичное отображение поверхности по сравнению с методом normal mapping.

Однако использование parallax mapping ограничено картами высот с небольшой разницей значений. Резкие неровности обрабатываются алгоритмом некорректно, появляются различные артефакты, «плавание» текстур и пр. Было разработано несколько модифицированных методов для улучшения техники parallax mapping. Несколько исследователей (Yerex, Donnelly, Tatarchuk, Policarpo) предложили новые методы, улучшающие начальный алгоритм. Почти все идеи этих методов основаны на трассировке лучей в пиксельном шейдере для определения пересечений деталей поверхностей друг другом. Модифицированные методы получили несколько разных названий: Parallax Mapping with Occlusion, Parallax Mapping with Distance Functions, Parallax Occlusion Mapping. Для краткости будем их все называть Parallax Occlusion Mapping.

Методы Parallax Occlusion Mapping включают еще и трассировку лучей для определения высот и учета видимости текселов. Поскольку при взгляде под углом к поверхности тексели загораживают друг друга, то учитывая это, можно добавить к эффекту параллакса больше глубины. Получаемое изображение становится реалистичнее и такие улучшенные методы можно применять для более глубокого рельефа, он отлично подходит для изображения кирпичных и каменных стен, мостовой и подобных поверхностей. Главное отличие метода Parallax Mapping от метода Displacement Mapping состоит в том, что все расчеты являются попиксельными, а не повершинными. Именно поэтому метод имеет названия Virtual Displacement Mapping и Per-Pixel Displacement Mapping.

Метод Parallax Mapping позволяет эффективно отображать детализированные поверхности без миллионов вершин и треугольников, которые потребовались бы при геометрической реализации. При этом сохраняется высокая детализация (кроме силуэтов/граней) и значительно упрощаются расчеты анимации. Этот метод проще, чем метод реальной геометрии, в нем используется значительно меньшее количество полигонов, особенно в случаях с очень мелкими деталями.

Дополнительное преимущество метода состоит в том, что карты высот могут динамически изменяться (поверхность воды с волнами, дырки от пуль в стенах и многое другое). К недостаткам данного метода можно отнести отсутствие геометрически правильных силуэтов (краев объекта), поскольку алгоритм попиксельный и не является настоящим Displacement Mapping. Однако он снижает требования к производительности в результате уменьшения затрат на трансформацию, освещение и анимацию геометрии и экономит видеопамять, необходимую для хранения больших объемов геометрических данных.

Композитные текстуры. Текстуры высокого разрешения занимают огромное место в памяти. Например, текстура размером 2048 x 2048 пикселей при глубине цветности 32 бита достигает объема 16 Мбайт. Очевидно, что затраты видеопамати при использовании таких текстур будут очень большие, и придется хранить часть текстур в ОП, что дополнительно нагружает шину видеоадаптера, занятую передачей данных для обсчета полигонов.

Для решения этой проблемы была предложена технология так называемых композитных текстур, или текстур с детализацией. При таком подходе требуется создать всего две текстуры: базовую и детальную. Базовая текстура содержит основные элементы и как бы создает общий фон, детальная текстура — лишь мелкие элементы, необходимые при рассмотрении объекта вблизи. Обе текстуры смешиваются, причем степень их взаимовлияния определяется исходя из расстояния до плоскости проецирования.

Алгоритм наложения текстур может реализовываться по-разному в зависимости от выбора разработчиков. Например, часто используют смешение с помощью альфа-канала или мультитекстурирования. В результате на большом расстоянии видна только базовая текстура, а по мере приближения объекта к картинной плоскости начинает проявляться детальная текстура. Пошаговый метод реализации основан на механизме MIP mapping и мультитекстурировании, т. е. с помощью механизма LOD определяется уровень MIP-каскада для выборки соответствующей детальной структуры (напомним, что базовая текстура остается неизменной). Затем детальная структура накладывается на базовую — и результат отправляется в буфер кадра. Очевидно, что при такой методике уровень детализации меняется дискретно, скачками. Будет ли это заметно пользователю, зависит от искусства разработчиков.

Смешение с помощью альфа-канала позволяет организовать динамическое изменение уровня детализации. Реализовать этот алгоритм достаточно просто, построив функцию зависимости степени прозрачности детальной текстуры от уровня LOD (Level-of-Detail — уровень детализации) и направления движения объекта (к картинной плоскости или в глубину).

Трехмерные текстуры. Элемент трехмерной текстуры представляет собой некоторый кубический объем. Каждая точка внутри текстурного куба имеет присвоенный ей цвет. Конечно, на самом деле трехмерный объект в компьютерной графике состоит из конечного числа точек (текселов), и потому его удобно представлять как конечный набор слоев (плоских текстур), текселы которых образуют узловые точки трехмерной текстуры.

Полигону, на который накладывается трехмерная текстура, присваиваются такие локальные координаты вершин, чтобы он оказался сечением куба. При этом не обязательно, чтобы полигон целиком помещался в объем куба. Обычно точке полигона вне куба присваивается цвет ближайшей граничной точки трехмерной текстуры.

Поскольку полигон может занять по отношению к кубу любое положение, лишь бы выполнялось сечение, возможны варианты, когда нельзя однозначно соотнести точку на полигоне и тексел трехмерной текстуры. В этом случае применяют либо прямое присваивание цвета ближайшей узловой точке, либо трилинейную фильтрацию, т. е. линейную аппроксимацию по восьми ближайшим узловым точкам.

Важным свойством трехмерных текстур является способность реалистично имитировать материалы с характерной внутренней структурой. Например, для

моделирования деревянной конструкции достаточно создать ее полигональную модель, затем разработать трехмерную текстуру, имитирующую рисунок деревянной поверхности и присвоить ее полигонам. При этом необходимость в рисовании текстур для каждого полигона и соединении их на границах уже отпадает. Кроме этого трехмерные текстуры позволяют реалистично имитировать спецэффекты, природные явления и даже движение тел. На самом деле, если поместить полигон последовательно с заданным сдвигом в трехмерную текстуру и также последовательно его отображать, меняя только одну координату, получится эффект анимации. Подобным методом можно имитировать огонь, блики на поверхности воды и металлов, множество других эффектов. Особое место занимает возможность правдоподобной имитации объемных эффектов: дыма, взрывов, локального тумана достаточно простыми способами по сравнению с технологией систем частиц.

Методы фильтрации текстур. Вычисление цвета точки полигона, попавшей между жирными точками текстуры, называется фильтрацией. Очевидно, что цвет пиксела проекции трехмерного объекта на экран должен однозначно определяться цветом тексела соответствующей текстуры. Однако это правило действует только в простых случаях при направлениях проецирования, близких к нормали, и соответствующем удалении от картинной плоскости. Дело в том, что как пиксел, так и тексел только в математическом смысле именуется точками. Физически они имеют вполне конкретные размеры и выглядят как окружности, диаметр которых зависит от разрешения экрана монитора и текстуры соответственно. Как только угол проецирования выходит за определенные рамки, бывает, что на один пиксел проецируются два и более тексела, так что проекция становится близкой к овалу. Если же объект очень близок к плоскости проецирования, может произойти обратное — один тексел будет проецироваться на несколько пикселей.

В простейшем случае для улучшения качества изображения применяют точечную выборку, т. е. цвет пиксела определяется по цвету тексела, расположенного в центре образованной его проекцией фигуры. Конечно, результат выглядит достаточно грубым, так как не учитываются ни цвета других пикселей, входящих в проецируемую фигуру, ни ее форма. Если такая текстура наложена на объект, уходящий в глубину сцены, эффект перспективы оказывается сильно искаженным.

Билинейная фильтрация. Метод расчета цвета по среднему арифметическому четырех ближайших соседей точки называют билинейной фильтрацией. В этом случае считается, что проекция представляет собой круг, а цвет пиксела рассчитывается путем аппроксимации цветов четырех текселей, как бы образующих данный круг.

При сильном приближении объекта к плоскости проецирования бывает, что в круг попадает меньше четырех текселей, тогда изображение выглядит чрезмерно размытым. Если же плоскость полигона повернута относительно плоскости проецирования, круг уже не соответствует реальной форме проекции (овалу,

эллипсу или иной фигуре) и эффект перспективы хотя и присутствует (все-таки сказывается аппроксимация по четырем точкам), но все равно существенно искажен. К тому же для определения цвета одного пиксела требуется считывать цвета четырех текселов из памяти, где хранятся текстуры, что увеличивает нагрузку на шину памяти.

Трилинейная фильтрация. Этот метод представляет собой комбинацию технологий MIP mapping и билинейной фильтрации (билинейной по текстуре и линейной по текстурам). В технологии MIP mapping применяют текстуры с разной степенью детализации (и разным разрешением) в зависимости от удаленности полигона от плоскости проецирования. При трилинейной фильтрации берутся две соседние текстуры, одна из которых содержит текселы, попадающие в проекцию, а другая является ближайшей к ней по удаленности, и к каждой применяют билинейную фильтрацию. В итоге аппроксимация цвета проводится уже по восьми текселам и результат выглядит ближе к реальности, так как текстуры заранее обчислены для определенных расстояний. Чтобы не было видно резких скачков при переходах от одного MIP-уровня к другому, видеопроцессор определяет линейную комбинацию цветов, вычисленных по ближним и по дальним текстурам.

Анизотропная фильтрация. Самым эффективным методом фильтрации является анизотропная (неоднородная по разным направлениям). Существуют различные алгоритмы анизотропной фильтрации, суть которых состоит в возможно более точном учете формы проекции при различном положении текстурированного полигона по отношению к проецируемой плоскости, т. е. вокруг центра проекции строится виртуальный куб из наложенных друг на друга текселов текстур разного уровня детализации, которые теоретически пересекает проекция.

Внутри куба плоскость проекции может располагаться как угодно — в идеале будут учтены все точки, попадающие в проекцию. В зависимости от размера грани куба может быть обчислено от 8 до 32 текселов для определения цвета единственного пиксела. Результат действительно близок к фотореалистичному, но и расход ресурсов GPU и видеопамати очень велик.

Environment Map (EM). Environment Map — карта окружающей среды (иногда ее называют картой отражения) — служит для отражения в объекте свойств окружающего пространства. Карты отражения либо создаются заранее при разработке игры (обычно используются сферические карты среды), либо в ходе построения трехмерной сцены (кубические карты среды).

Визуальные свойства карт окружающей среды можно сделать зависимыми от направления линии визирования (что соответствует правилам физической оптики). В целях ускорения обработки трехмерной сцены разработчики иногда не предусматривают изменения карты среды в зависимости от направления линии визирования.

Более качественным является метод создания кубических карт окружающей среды в ходе построения трехмерной сцены. По сути дела, для этого необходимо предварительно выполнить рендеринг сцены относительно объекта, к которому

применяется технология EMBM. Затем попиксельное состояние окружения записывается в кубическую (Cube Map) текстурную карту, которая и накладывается на объект.

Сжатие текстур. Повышение разрешения текстур до 2048x2048 точек при 32-битном цветовом охвате вызвало существенное возрастание требований к объему памяти, выделяемой для хранения текстурных карт. Ведь на основе базовой текстуры, согласно многим технологиям, генерируются другие уровни текстурных карт. Кроме того, часто применяют и прочие карты: освещенности, рельефа, окружающей среды и т. д. Очевидно, что хранить большой объем данных в обычном виде становится нерациональным. Разработчики видеоадаптеров предлагают использовать различные методы сжатия данных для текстурных карт. Здесь существенны два требования: сжатие практически без потери качества и обработка данных «на лету» в процессе преобразования и наложения текстур.

Первой по-настоящему общепризнанной технологией сжатия текстур стала технология S3TC (S3 Texture Compression). В зависимости от характера текстуры степень сжатия достигает 6:1. Распаковка текстур производится «на лету» и выполняется как программными, так и аппаратными средствами. Это обстоятельство привлекло интерес фирмы Microsoft, и она лицензировала новую технологию у фирмы S3, включив механизм сжатия текстур в библиотеку DirectX (технология получила обозначение DXTC).

Вскоре появились технологии сжатия, альтернативные S3TC — технология FXT1 от фирмы 3Dfx, объявленная открытой. Ее преимущества заключаются в большем коэффициенте сжатия, достигающем 8:1, и меньшей потере качества при компрессии текстур.

Несколько особняком стоит технология VTC (Volume Texture Compression) от фирмы nVidia. Она ориентирована исключительно на сжатие трехмерных текстур. Известно, что 3D-текстуры являются самыми требовательными к объему видеопамати, однако взамен они обеспечивают великолепное качество изображения, близкое к фотореалистичному.

Методы улучшения изображения. Дефекты изображения, возникающие при рендеринге трехмерной сцены, носят различный характер. Обычно пространственные искажения выражаются в ступенчатости прямолинейных границ объектов, потере мелких деталей изображения, появлении муара (регулярной структуры на изображении, не предусмотренной разработчиками), искажении текстур. Особым видом являются дефекты, связанные с непреднамеренной анимацией сцены, например мерцание объектов, обусловленное постоянными переключениями между ступенями LOD.

Технологии устранения ступенчатости прямолинейных границ объектов получили название антиалиасинг, или сглаживание. Пространственные дефекты сглаживаются либо локальными, либо глобальными методами. Глобальные методы применяют ко всей области картинной плоскости. До недавнего времени основной в 3D-ускорителях фирмы ATI была технология сглаживания дефектов сцены (FSAA — Full scene anti-aliasing), использующая метод масштабирования

изображения. Видеоадаптер в этом случае выполняет рендеринг сцены с разрешением, превосходящим необходимое экранное разрешение, а затем производится уменьшение изображения, при этом каждый пиксел обрабатывается на основе маски, соответствующей коэффициенту масштабирования. Например, при маске 2×2 и экранном разрешении 800×600 точек необходимо выполнить рендеринг сцены в разрешении 1600×1200 . Затем каждому пикселу в реальном изображении будет присвоен цвет, получившийся в результате смешения цветов соседних пикселов маски 2×2 в виртуальном разрешении. При этом сглаживаются не только ступеньки, улучшается и цвет всех текстур, переходы становятся более плавными. Однако использование этого способа снижает производительность минимум в 4 раза. Метод антиалиасинга, описанный выше, многим знаком как суперсэмплинг (supersampling, SSAA), точнее, Ordered Grid Supersampling, OGSS. *Сэмпл* — избранный участок изображения, содержащий 1, 4, 16 и т. д. пикселов, используемый для обработки объекта и сглаживания. Существует еще один метод суперсэмплинга — Rotated Grid Supersampling, RGSS. При его применении рассчитывается n версий изображения, смещенных на небольшое расстояние относительно исходного изображения в различных направлениях, а потом цвета субпикселов дочерних изображений, которые имеют одинаковые координаты, смешиваются для получения цвета итогового пиксела.

Локальные методы применяют к границам объектов, т. е. текселы текстуры, являющиеся краевыми в полигоне, обрабатываются таким образом, чтобы исключить ступенчатость. Широко распространенной является технология усреднения по площади. Для этого определяется весовое соотношение текселов, чьи проекции пришлись на данный пиксел. Затем их цвета смешиваются в соответствии с весовыми коэффициентами и присваиваются данному пикселу. Такой способ сглаживания обычно называют мультисэмплинг (multisampling, MSAA). Это тот же суперсэмплинг, только применяемый для обработки границ полигонов, а не всего изображения. Действительно, антиалиасинг нужен для сглаживания краев полигонов, так зачем производить ту же самую работу на поверхности каждого полигона? Смысл мультисэмплинга (раньше этот метод назывался HRAA — High Resolution Antialiasing) состоит в том, что не нужно обрабатывать все до единого пиксела на экране, ведь подавляющее большинство из них находится внутри полигона, а не на границе. Конечно, отказ от сглаживания всей картинке ухудшает качество вывода текстур, но с этой проблемой должна бороться билинейная, трилинейная и анизотропная фильтрации. Технически мультисэмплинг происходит так. Видеопроцессор находит пикселы, расположенные на границе полигонов. В зависимости от уровня мультисэмплинга процессор делит пикселы на 2, 4, 8 и т. д. субпикселов и усредняет цвета каждого из них. В итоге наблюдается не такое сильное падение производительности, как при обычном суперсэмплинге, поскольку, во-первых, процессор сэмплирует только грани между полигонами и, во-вторых, он использует не новую текстуру для каждого пиксела, а одну и ту же. Особенность реализации мультисэмплинга у каждого из производителей видеокарт состоит в том, как именно процессор выбирает пикселы и разбивает их на субпикселы.

Современные графические процессоры компании nVidia основаны на технологии улучшения качества изображения Intellisample 4.0, которая поддерживает два режима сглаживания — адаптивного суперсэмплинга и адаптивного мультисэмплинга с учетом прозрачности, которые повышают качество и скорость сглаживания. Режимы адаптивного суперсэмплинга и мультисэмплинга с учетом прозрачности используют дополнительные тексельные сэмплы и этапы сглаживания для улучшения качества объектов, состоящих из тонких линий, таких как звенья цепи, деревья и растительность. Такие типы объектов обычно строятся на базе очень простых моделей полигонов (или даже на одном полигоне). Сложность результирующего изображения (группы ветвей или растений) зависит от текстуры, накладываемой на полигон. Традиционный вид сглаживания не давал таких результатов, так как края ветвей или растений обычно находятся внутри проецируемой текстуры. К пикселям внутри полигона не применяются существующие на сегодня методы сглаживания.

Адаптивный мультисэмплинг с учетом прозрачности также повышает качество сглаживания даже при более высоких скоростях, так как один тексельный сэмпл используется для расчета окружающих субпиксельных значений. Хотя адаптивный мультисэмплинг с учетом прозрачности не обеспечивает такое высокое качество, как метод суперсэмплинга, его повышенная эффективность компенсирует качество изображения высокой производительностью. Визуальные улучшения адаптивного суперсэмплинга очевидны по сравнению с традиционными методами суперсэмплинга/мультисэмплинга.

Компания ATI пошла другим путем. Вместо того чтобы бороться с некачественным сглаживанием путем наращивания количества сэмплов, она решила исказить решетку, по которой размещаются субпиксели, эта технология получила название Smooth Vision. Такой случайный отбор местоположения сэмпла позволяет намного лучше сгладить картинку, при этом сохраняя вполне приемлемый уровень производительности.

В графическом процессоре R420 компании ATI реализован новый алгоритм сглаживания, называемый Temporal Antialiasing (временное сглаживание). Суть технологии заключается в улучшении качества сглаживания благодаря особенностям человеческого зрения, которое обладает свойством инерционности. При рендеринге сцены положение субпикселей меняется так, что на каждом четном и нечетном кадре субпиксели смещаются относительно начального положения. В итоге глаз усредняет два соседних кадра и создается впечатление, что использовано в 2 раза больше сэмплов, чем на самом деле. Существенным ограничением в применении алгоритма является обязательность включения вертикальной синхронизации с монитором. Если в игре частота кадров падает ниже частоты синхронизации, алгоритм Temporal Antialiasing отключается и применяется обычное сглаживание.

После появления технологий SLI и Crossfire возродился интерес и к суперсэмплингу. Теперь у обоих производителей доступны режимы, в которых традиционный сегодня мультисэмплинг комбинирован с простым суперсэмплингом 2х, что позволяет улучшить качество текстур и сгладить изображения не только

на гранях полигонов, но и внутри них (например, текстуру с надписью наподобие тех же вывесок в любых играх). Эти режимы объединены в один — SLI AA и Crossfire Super AA.

Postprocessing (постобработка). В широком смысле, постобработка — все то, что происходит после основных действий по построению изображения. Иначе говоря, постобработка — любые изменения изображения после его рендеринга. Постобработка представляет собой набор средств для создания специальных визуальных эффектов. Их создание производится уже после того, как основная работа по визуализации сцены выполнена, т. е. при создании эффектов постобработки используется готовое растровое изображение.

Существует много разных возможностей по обработке изображения после его рендеринга. В графических редакторах имеется множество так называемых графических фильтров, или постфильтров: blur, edge detection, sharpen, noise, smooth, emboss и др. В применении к 3D-рендерингу в реальном времени это происходит следующим образом: выполняется рендеринг всей сцены в специальную область (render target) и после основного рендеринга это изображение дополнительно обрабатывается при помощи пиксельных шейдеров и только потом выводится на экран. Из эффектов постобработки в играх чаще всего используют Bloom, Motion Blur, Depth Of Field. Существует и множество других постэффектов: noise, flare, distortion, sepia и др.

High Dynamic Range (HDR). В применении к 3D-графике — это рендеринг в широком динамическом диапазоне. Суть HDR заключается в описании интенсивности и цвета реальными физическими величинами. Чаще других для описания изображения используется цветовая модель RGB (см. гл. 1), когда все цвета представляются как сумма основных цветов: красного, зеленого и синего, с разной интенсивностью в виде возможных целочисленных значений от 0 до 255 для каждого, при 8-разрядном кодировании цвета. Отношение максимальной интенсивности к минимальной, доступной для отображения конкретной моделью или устройством, называется динамическим диапазоном. Так, динамический диапазон модели RGB составляет 256:1 или 100:1 cd/m² (два порядка). Общепринятое название этой модели описания цвета и интенсивности — Low Dynamic Range (LDR).

Возможных значений модели LDR для всех случаев явно недостаточно, человек способен воспринимать гораздо больший динамический диапазон, особенно при малой интенсивности света, а модель RGB для этого слишком ограничена даже при большой интенсивности. Динамический диапазон зрения человека составляет от 10⁻⁶ до 10⁸ cd/m², т.е. 100000000000000:1 (14 порядков). Одновременно весь диапазон человек видеть не может, но диапазон, видимый глазом в каждый момент времени, примерно равен 10000:1 (четырем порядкам). Зрение приспосабливается к значениям из другой части диапазона освещенности постепенно, при помощи так называемой адаптации, как, например, при выключении света в комнате в темное время суток — сначала глаза видят очень мало, но со временем адаптируются к изменившимся условиям освещения и видят уже намного лучше. То же самое происходит и при обратной смене темной среды на ярко освещенную.

Таким образом, динамического диапазона цветовой модели RGB недостаточно для представления изображений, которые человек способен видеть в реальности, эта модель значительно уменьшает возможные значения интенсивности света в верхней и нижней частях диапазона. Самый распространенный пример, приводимый в материалах по HDR, — изображение затемненного помещения с окном на яркую улицу в солнечный день. При использовании модели RGB можно получить или нормальное отображение того, что находится за окном, или только того, что внутри помещения. Значения больше 100 cd/m^2 в LDR вообще обрезаются, это является причиной того, что при 3D-рендеринге трудно правильно отображать яркие источники света, направленные непосредственно в камеру.

Сами устройства отображения данных пока что серьезно улучшить нельзя. В этом случае целесообразно отказаться от модели LDR при расчетах, а использовать реальные физические величины интенсивности и цвета (или линейно пропорциональные), а на монитор выводить максимум того, что он сможет. Суть представления HDR состоит в использовании значений интенсивности и цвета в реальных физических величинах или линейно пропорциональных и в том, чтобы использовать не целые числа, а числа с плавающей точкой с большой точностью (например, 16 или 32 бит). Это позволит преодолеть ограничения модели RGB, а динамический диапазон изображения намного увеличится. Затем любое HDR изображение можно отобразить на любом средстве отображения (например, на RGB-мониторе) с максимально возможным качеством для него при помощи специальных алгоритмов *tone mapping*.

HDR-рендеринг позволяет изменять экспозицию уже после рендеринга изображения, имитировать эффект адаптации человеческого зрения (перемещение из ярких открытых пространств в темные помещения, и наоборот), а также выполнять физически правильное освещение. Он является унифицированным решением для применения эффектов постобработки (*glare*, *flares*, *bloom*, *motion blur*). Алгоритмы обработки изображения, цветокоррекцию, гамма-коррекцию, *motion blur*, *bloom* и другие методы постобработки качественнее выполнять в HDR-представлении.

В приложениях 3D-рендеринга реального времени HDR-рендеринг начали использовать не так давно, поскольку это требует вычислений и поддержки *render target* в форматах с плавающей точкой, которые стали доступны только в видеопроцессорах с поддержкой DirectX 9. Обычная последовательность HDR-рендеринга в играх такова: рендеринг сцены в буфер формата с плавающей точкой, постобработка изображения в расширенном цветовом диапазоне (изменение контраста и яркости, цветового баланса, эффекты *glare* и *motion blur*, *lens flare* и подобные), применение *tone mapping* для вывода итоговой HDR-картинки на LDR-устройство отображения. HDR-рендеринг очень полезен для комплексной постобработки более высокого качества по сравнению с обычными методами.

2.2.3. Последовательность работы графического конвейера

Совокупность аппаратных и программных средств обработки трехмерных сцен образует графический конвейер, конечным результатом работы которого является кадр, отображаемый на экране монитора. Рассмотрим последовательные этапы работы такого конвейера.

Большинство приложений трехмерной графики при построении трехмерных сцен придерживаются определенной последовательности действий, в совокупности составляющей ЗБ-конвейер (рис. 2.20). Итогом работы ЗБ-конвейера является получение результирующего изображения в буфере кадра и его отображение на экране монитора компьютера. Группу операций, выполняющих обособленные промежуточные действия, принято называть этапом, или стадией ЗБ-конвейера. Описываемая ниже последовательность операций является в некоторой степени общепринятой в современных графических подсистемах. При конкретной реализации на программном и аппаратном уровнях могут появляться существенные отличия, однако смысловое содержание блоков практически не меняется. На сегодняшний день процесс визуализации трехмерной сцены на экране монитора в общих чертах выглядит следующим образом.

Этап 1. На этом этапе решаются задачи создания модели трехмерной сцены, которую необходимо отобразить. В их число входят определение перечня объектов сцены, их положения и ориентации в объеме сцены, состояния объектов, назначения материалов объектов, задание положения источников освещения, их типа и параметров и т. п. С каждым объектом в сцене (например, при создании игр) может быть связан некоторый набор геометрических моделей, т. е. объект один (например, некий монстр), но с ним сопоставляются разные геометрические модели в зависимости от его состояния — жив, ранен, убит, трансформировался в другой объект и т. д. Практически все задачи первого этапа решает ЦП. Результаты его работы пересылаются в графический процессор по шине графического адаптера с помощью драйвера.

Однако чем выше требования к качеству отображения трехмерной сцены и чем сложнее сцена, тем больше объем передаваемых данных и тем выше требования к пропускной способности шины графического адаптера. Одним из основных критериев реалистичности отображения сцены является сложность представленных в ней моделей. Обычно считается, что начиная с 500 000 треугольников в сцене, можно обеспечить качество отображения, близкое к фотореалистичному. Объем данных для одного треугольника в этом случае близок к 1 Кбайту, а общий поток данных превысит 2,5 Гбайт/с. Поэтому для современных графических адаптеров характерна миграция с шины AGP на шину PCI Express 16x с пропускной способностью до 8 Гбайт/с.

Кроме того, путем использования трехмерных поверхностей высокого порядка можно добиться улучшения формы трехмерных полигональных моделей и снижения потока передаваемых данных. В этом случае исходный объект можно представить меньшим числом элементов, а детализацию выполнить средствами графического адаптера путем разбиения граней на большее число треугольников



Рис. 2.20. Этапы работы графического конвейера

(тесселяция). В библиотеке DirectX 8 появились средства описания поверхностей высокого порядка с помощью набора контрольных точек *{Patches}*.

Этап 2. Выполняется построение геометрических моделей объектов трехмерной сцены. Внешний вид объекта формируется с помощью полигональной модели. Чаще всего в роли полигонов используются треугольники, поскольку треугольник — это простейшая плоская фигура, однозначно располагаемая в трехмерном пространстве. Как правило, применительно к 3D-графике в компьютерных играх термины «полигон» и «треугольник» являются синонимами. Современные графические процессоры умеют выполнять дополнительные операции, например тесселяцию *{Tessellation}*, т. е. разделение исходных треугольников на более мелкие. Некоторые графические процессоры могут аппаратно обрабатывать геометрические модели, построенные на основе параметрических поверхностей (механизм *RT-Patches*). Часть графических процессоров может превращать плоские треугольники в трехмерные поверхности путем «выдавливания» в третье измерение (механизм *N-Patches*).

Данные вершин выбираются из памяти и попадают в предварительный кэш вершин. Современные видеоадаптеры поддерживают несколько топологий хранения геометрии (треугольников) в памяти. Кроме того, поддерживается аппаратно-гибкий формат данных — для каждой вершины могут храниться не только классические данные, такие как координаты или вектор ее нормали, но и любые другие наборы атрибутов допустимых скалярных или векторных типов. Существует аппаратная поддержка работы с несколькими потоками данных — когда часть атрибутов вершины хранится в одном массиве данных, а другая часть — в другом. В таком случае выборка из памяти должна идти в несколько потоков. Всем этим и занимается блок выборки геометрии.

Этап 3. Итоговый результат операций второго этапа (каждая вершина) пересылается в вершинный процессор геометрического процессора для обработки вершинным шейдером. Каждая вершина полигона полностью описывается набором атрибутов: трехмерные координаты, нормаль, цвет, текстурные координаты и др. При обработке вершины вершинным шейдером графический процессор не имеет какой-либо информации о соседних вершинах треугольника. Нормаль (вектор, перпендикулярный к моделируемой поверхности) также является атрибутом вершины, но не треугольника. Вершинные процессоры последовательно обрабатывают каждую вершину объекта, выполняя межкадровую интерполяцию вершин (*Key Frame Interpolation*), наложение вершин (*Vertex Blending*) с использованием более чем четырех матриц преобразования, искажение свойств вершин каким-либо параметрическим объектом и другие трансформации, а также расчет освещенности, используя сложные модели освещения, учитывающие свойства материала объектов, и другие операции. Содержание операций преобразования и расчета освещенности объектов — *Transform & Lightning (T&L)* — можно динамически изменять посредством вершинных шейдеров. Таким образом, в современном персональном компьютере многоядерный ЦП дополняется еще более сложным программируемым графическим процессором, содержащим несколько десятков ядер.

С практической точки зрения нерационально все объекты трехмерной сцены отображать с максимальной детализацией, так как зрение человека не может воспринимать мелкие детали удаленных объектов. К тому же непрерывная обработка нескольких десятков или сотен тысяч полигонов, составляющих объект, независимо от его удаления приводит к дополнительной и бесполезной нагрузке графического процессора. Изменить эту ситуацию можно, определяя в некоторые моменты времени так называемый уровень детализации LOD (*Level of Detail*) для всех объектов в трехмерной сцене. Расчет уровня детализации (LOD) в 3D-приложениях позволяет снизить сложность рендеринга кадра за счет уменьшения общего количества полигонов, текстур и иных ресурсов в сцене. Метод особенно эффективен, если количество объектов в сцене велико и они расположены на разных расстояниях от камеры. Существуют статический и динамический подходы к управлению детализацией. В первом случае заранее создаются упрощенные варианты максимально детализованного объекта. В тех-

нологии динамического или непрерывного управления детализацией используют различные алгоритмы, позволяющие плавно регулировать число полигонов в объекте в зависимости от расстояния до картинной плоскости.

У каждого метода есть свои преимущества и недостатки. Статическое управление иногда вызывает эффект дергания изображения при смене детализации объекта. Если уровень детализации близок к граничному значению, может возникнуть циклическая смена моделей с разным уровнем детализации. К тому же необходимо обсчитывать несколько разных моделей для одного объекта. Динамическое управление детализацией потребляет значительные вычислительные ресурсы, требует непрерывного пересчета не только координат вершин треугольников, но и параметров освещенности текстур. При частом переключении между уровнями наблюдается эффект волнистости поверхности — форма объекта непрерывно «плывет», что при моделировании объектов неживой природы выглядит особенно нереально. Необходимо отметить, что уровень детализации не обязательно относится только к геометрии, метод может применяться и для экономии других ресурсов: при текстурировании (хотя видеопроцессоры и так используют мипмэппинг, иногда есть смысл менять текстуры мгновенно на другие, с иной детализацией), методы освещения (близкие объекты освещаются по сложному алгоритму, а дальние — по простому), методы текстурирования (на ближних поверхностях используется сложный параллаксмэппинг, а на дальних — нормалмэппинг) и т. п.

Этап 4. После вершинного процессора трансформированные и освещенные, т. е. уже обработанные вершинным шейдером, вершины попадают в небольшой (порядка 32 вершин в современных архитектурах) промежуточный буфер. Он называется «Post T&L буфер вершин». Этот буфер играет двоякую роль, во-первых, служит для накопления результатов, готовых отправиться на последующие стадии конвейера, и таким образом уменьшает вероятность потенциальных простоев блоков графического процессора в ожидании данных, а во-вторых, позволяет избежать повторного трансформирования и обработки шейдером вершины, если она будет повторно использована в скором времени. Например, это часто происходит с соседними треугольниками, несмотря на наличие разных подходов к описанию геометрии, практически в каждом из них одна вершина будет использоваться несколько раз.

Этап 5. Вершины объединяются в соответствии с выбранной топологией (обычно по три), в соответствии с полигонами (треугольниками), к которым они принадлежат, и отправляются в блок установки треугольников, где происходит предварительная подготовка данных, необходимых для закраски всего полигона. Поскольку треугольники часто имеют общие вершины, обычно требуется обработка лишь одной вершины для каждого нового треугольника. Недостающие вершины считываются из Post T&L буфера вершин. Здесь же производится отбрасывание невидимых (вышедших за экран или заданных плоскостей отсечения) и повернутых к нам обратной стороной (если такая опция задействована) треугольников.

Этап 6. Треугольник разбивается на фрагменты, часть которых признается невидимыми и отбрасывается в ходе предварительного теста глубины на уровне фрагментов. Результатом этого процесса являются видимые (или частично видимые) фрагменты (обычно квады), подлежащие закраске. Затем сравниваются значения глубины и отбрасываются полностью невидимые квады (если такие будут), а остальные отправляются на установку и закраску в пиксельный процессор. При этом они снабжаются вычисленными значениями глубины и специальной битовой маской, которая говорит, какие из пикселей квадра видимы, а какие следует игнорировать (поскольку часть квадра может оказаться за гранью треугольника, а другая часть пикселей может быть невидима из-за того, что не пройдет тест глубины).

В среднем, как минимум, половина пикселей отбрасывается еще до закраски, но это зависит от структуры сцены. Поэтому производительность проверки и вычисления глубины должна быть выше максимальной производительности закраски.

Этап 7. При установке фрагментов для каждого из квадов вычисляются параметры, такие как текстурные координаты, MIP-уровень, векторы и установочные параметры анизотропии и т. д., необходимые для дальнейших расчетов. Именно здесь вычисляются только базовые значения параметров на весь блок и специальные коэффициенты dx и dy — их предполагаемое изменение по горизонтали и вертикали квадра. А затем из этого набора параметров получаются все четыре значения. По мере роста сложности пиксельных шейдеров увеличивается и число передаваемых и интерполируемых для каждой точки параметров. Число интерполяторов в видеопроцессоре не может быть очень большим, поскольку это достаточно длительная операция, и интерполяция типичного набора параметров может занимать более одного такта, замедляя закраску. Использование квадов позволяет существенно поднять эффективность этого процесса.

В практических реализациях процесс интерполяции параметров может частично или полностью проходить в пиксельном процессоре, используя его специальные или общие ресурсы (ALU).

Этап 8. После установки и интерполяции параметров происходит закраска фрагментов. Все обрабатываемые квады по очереди проходят через длинный конвейер, состоящий из ALU, затем двух текстурных модулей и затем еще двух ALU. Длина конвейера (и соответственно время прохода квадра через него) составляет более двух сотен тактов, большая часть которых (около 170) приходится на операцию выборки и фильтрацию текстур и скрыта в текстурных блоках. В нормальном режиме работы конвейер способен выдавать по квадрату за такт. Затем, если двух выборок текстур и трех операций было недостаточно, квадрат может отправиться по кругу, через буфер квадов, повторно на вход этого конвейера.

Этап 9. После того как значения цвета были рассчитаны, если включен соответствующий режим, в пиксельном процессоре происходит смешивание (блендинг). Под этим понимается комбинирование двух или более текстур с использованием некоторого базиса пикселей, которое определяется коэффициен-

том смешивания (blend factor) и режимом смешивания (blend mode). Если цвет пиксела представлен в формате RGBA, возможно выполнение смешивания с учетом альфа-канала (alpha blending). Этот метод позволяет моделировать прозрачность объектов 3D-сцены для создания оптических эффектов типа дыма, стекла или воды. При отсутствии блендинга выполняется просто запись результирующих значений цвета и глубины в буфер кадра. На этом этапе может происходить несколько дополнительных операций, таких как: гамма-коррекция или вычисление самого большого значения глубины всего блока 4x4 для корректного обновления мини-буфера глубины, сжатие Z-координат и т. д.

Этап 10. После того как изображение построено, может быть произведен дополнительный проход для усреднения результатов полноэкранного сглаживания изображения, иногда этот процесс сразу совмещается с выводом на экран.

Перечисленные этапы в конкретных графических процессорах могут быть переставлены, разделены, объединены, выполняться в несколько проходов, однако их физический смысл остается неизменным. Технологически некоторые элементы этапов или этапы целиком могут быть выполнены различными способами. Вариант реализации зависит от особенностей приложения и видеоадаптера.

2.2.4. Поколения графических процессоров

Первое поколение (1995-1997). Первое поколение графических карт представлено чипами, которые могут использоваться и на шине PCI, и на шине AGP, т. е. их производительность не превосходит пропускной способности шины PCI, и потому вариант AGP ничем не лучше. Среди карт первого поколения можно выделить модели Voodoo Graphics и Voodoo Rush компании 3Dfx, Riva 128 и Riva 128ZX компании nVidia.

Второе поколение (1997-1999). Второе поколение охватывает широкий круг видеокарт, которые нормально работают только на шине AGP, так как их производительность в принципе превышает возможности шины PCI. Ко второму поколению относятся чипсеты 3Dfx Voodoo2, 3Dfx Voodoo Banshee, nVidia Riva TNT, Matrox G200, S3 Savage 3D, i740 и более поздние 3Dfx Voodoo3, 3Dfx VSA-100, nVidia Riva TNT2, Matrox G400, S3 Savage4, ATI Rage128.

У карт второго поколения аппаратные конвейеры могут одновременно обрабатывать две текстуры, они поддерживают до 64 Мбайт видеопамати, часто поддерживается 32-битный цвет. Повышенная частота RAMDAC обеспечивает комфортную работу в высоких разрешениях экрана монитора. Глубина Z-буфера составляет 24...32 бит. Стандартом считается аппаратная поддержка мультитекстурирования, анизотропной фильтрации и прочих современных технологий.

Третье поколение (1999-2002). Третье поколение представлено видеоускорителями DirectX 7, оснащенными принципиально новым элементом — геометрическим процессором. Тем самым значительная часть расчетов геометрических преобразований и параметров освещения снимается с ЦП компьютерной системы.

Это позволило значительно ускорить обработку трехмерных сцен. Однако следует подчеркнуть, что разработчики программ должны специально предусмотреть поддержку новых возможностей в своих приложениях. Согласование параметров аппаратных средств и программного кода стало возможным благодаря принятию типового API DirectX 7, разработанного компанией Microsoft. С этого времени поколения графических процессоров принято различать по способности аппаратно реализовать функции какой-либо версии DirectX.

Четвертое поколение (2001-2005). Отличительной чертой видеокарт поколения DirectX 8 стало появление программируемого блока обработки атрибутов вершин (процессора вершин). Программы обработки (вершинные шейдеры) сначала выполняли геометрические операции, затем могли работать с цветом вершин и прозрачностью. Подобный блок для расчета цвета пикселей на основе пиксельных шейдеров стал частью пиксельного конвейера.

Пятое поколение (2004 - н. вр.). Графические ускорители с полностью программируемым графическим процессором относятся к поколению DirectX 9. Благодаря их появлению разработчики программ получили возможность описывать способы обработки графики с помощью команд, похожих на операторы языков программирования высокого уровня, например C++. В частности, компания nVidia даже разработала язык Cg (C Graphics) для программирования своих графических процессоров. Поддержка программируемых графических процессоров предусмотрена в API DirectX 9 различных версий. В настоящее время последней является версия DirectX 9.0с, поддерживающая вершинные и пиксельные шейдеры версии 3.0.

2.2.5. Мониторы

Основными устройствами оперативного вывода графической информации (и не только графической) в современных ЭВМ являются различные мониторы. Наиболее распространенным в течение многих лет мониторам на ЭЛТ, производство которых постоянно сокращается, пришли на смену мониторы на жидкокристаллических панелях. Тем не менее мониторы на ЭЛТ до сих пор остаются непревзойденными в основных требованиях, которые предъявляются к устройствам отображения красочной информации — цветопередаче, скорости реакции точки, углом обзора. По этой причине ниже рассматриваются не только наиболее популярные сегодня мониторы на жидкокристаллических панелях или перспективные технологии, но и мониторы на ЭЛТ.

Мониторы на ЭЛТ. Всем знакомы мониторы с ЭЛТ. Эта технология отработана до тонкостей многолетним производством, и до сих пор выпускается много мониторов на ее основе. Работают такие устройства на ЭЛТ (CRT — Cathode Ray Tube), подобной кинескопу обычного домашнего телевизора. Как и телевизоры, мониторы выпускают с плоским экраном и традиционным, т. е. выпуклым. В отличие от традиционного монитора с плоским экраном обычно дает более четкое изображение без искажений.

Параметры монитора определяются характеристиками и качеством элементов, управляющих видеотрактом и в основном характеристиками ЭЛТ. Иногда на основе ЭЛТ одной модели производители выпускают мониторы разных ценовых категорий, меняя лишь их электронные блоки. В свою очередь, параметры ЭЛТ во многом зависят от выбранной технологии производства. Причем сложность современных технологий производства ЭЛТ такова, что освоить их, а затем и продолжить исследования могут только крупные производители, среди которых Hitachi, Mitsubishi, Sony, Toshiba, Panasonic/Matsushita, Samsung, LG-Philips.

Принципиально конструкция ЭЛТ для монитора совпадает с конструкцией телевизионного кинескопа. В горловине стеклянной колбы, дно которой покрыто слоем люминофора, установлена электронная пушка, испускающая поток электронов. Этот поток отклоняется электромагнитным полем отклоняющей системы в нужном направлении и затем, проходя через теньевую маску, установленную перед дном колбы, попадает на люминофор, вызывая его свечение.

В цветных мониторах для формирования изображения применяют отдельные пушки для каждого из основных цветов (Red — красный, Green — зеленый, Blue — синий), а слой люминофора составляют из близко расположенных точек группами по три (также в сочетании Red, Green, Blue — RGB) точек цветного люминофора. Для точного попадания в заданную точку люминофора необходимо электронный луч сфокусировать до заданных размеров в плоскости экрана. Это осуществляется установкой после пушек специального фокусирующего электрода и теньевой маски, имеющей отверстия с размерами, близкими к размерам отдельной точки люминофора, непосредственно перед люминофорным покрытием. В зависимости от типа маски и характера отверстий различают три основные технологии: трехточечная (дельтавидная) теньевая маска (Dot-Trio shadow-mask); апертурная решетка (Aperture-grille); щелевая маска (Slot-mask).

Каждая технология имеет свои преимущества и свои недостатки, поэтому среди изготовителей, специалистов и пользователей есть сторонники и противники того или иного варианта. Поверхность экрана ЭЛТ на основе теньевой маски имеет форму сферы очень большого радиуса. В мониторах с апертурной решеткой поверхность экрана представляет собой часть цилиндра также большого радиуса.

Трехточечная теньевая маска. Эта технология относится к проверенным техническим решениям. Физически она представляет собой перфорированный металлический лист, расположенный перед люминофором. Расстояние между группами соседних точек таково, что маскируются все паразитные излучения, обеспечивается попадание луча от каждой электронной точки в требуемую точку люминофора (с учетом допуска). За счет улучшения систем управления отклоняющей системой удается выпускать трубки с практически плоской поверхностью экрана (так называемого типа FST). Традиционно считается, что мониторы с теньевой маской лучше воспроизводят текст, имеют высокую контрастность, хорошие показатели стоимость-эффективность. К недостаткам обычно относят

пониженную точность цветопередачи и меньшую яркость, которые сведены к минимуму в современных моделях трубок от известных производителей.

Апертурная решетка. ЭЛТ с апертурной решеткой была разработана фирмой Sony. Выполненные по этой технологии ЭЛТ известны на рынке под именами, содержащими характерное окончание *tron* (Trinitron, DiamondTron). Маска представляет собой тонкую фольгу, в которой выполнены частые вертикальные отверстия (апертурная решетка). Поперек размещают две-три нити, обеспечивающие жесткость конструкции. Соответственно и люминофор на дне колбы располагается в виде вертикальных чередующихся полосок разных цветов. Особенности технологии позволяют увеличить процент электронов, попадающих на люминофор, и добиться лучшей яркости изображения. Использование более темного стекла компенсирует недостаток контрастности. Мониторы с трубками на основе апертурной решетки традиционно привлекают специалистов для работы с графикой, требующей ярких и чистых цветов. Однако некоторые профессионалы считают недостатком сравнительно невысокую контрастность и наличие на экране тени от поперечных проволочных нитей.

Щелевая маска. Последней технологией, разработанной фирмой NEC, была технология щелевой маски. Под торговой маркой ChromaClear были выпущены ЭЛТ, в которых тeneвая маска образована продольными щелями. Соседние триады рядов таких щелей смещены по вертикали, образуя решетку с расположением элементов в шахматном порядке. По сути дела, в технологии щелевой маски удалось совместить достоинства предыдущих конструкций, почти избавившись от их недостатков. Специалисты признают, что решение NEC является универсальным для всех групп пользователей.

Параметры мониторов на ЭЛТ. Мониторы на ЭЛТ обычно оцениваются следующими параметрами: размером экрана по диагонали в дюймах, шагом точек, диапазоном частот горизонтальной и вертикальной развертки, полосой пропускания видеотракта, габаритными размерами, весом, потребляемой мощностью и поддерживаемыми интерфейсами и режимами работы. Кроме того, приводятся стандарты качества, требованиям которых удовлетворяет монитор.

Традиционно количественным выражением качества изготовления маски и люминофора служит минимальный размер точки на экране кинескопа. Раньше под этим понимался минимальный диаметр точки. Для современных ЭЛТ с тeneвой маской используют понятие «шаг точек» (Dot Pitch), для апертурной решетки — «шаг полос» (Strip или Grille Pitch).

В ЭЛТ с тeneвой маской принято измерять расстояния (т. е. шаг) между двумя соседними точками люминофора по диагонали. Для апертурной решетки и щелевой маски расстояние меряют по горизонтали. Примерное соотношение таково: 0,27 мм Dot Pitch эквивалентно 0,25 мм Strip Pitch. Нормальным для маски сегодня считается шаг 0,28 мм, качественные мониторы имеют шаг 0,25 мм, профессиональные мониторы с апертурной решеткой — 0,22...0,24 мм. Величина шага заметно сказывается на качестве изображения. Поэтому для графических работ следует выбирать мониторы с шагом не более 0,25 мм. В трубках с

плоским экраном иногда используют маски (решетки) с переменным шагом от центра к краям, дабы обеспечить неизменность пропорций изображения. В противном случае картинка на экране будет выглядеть вогнутой.

Внедрение новых технологий в систему управления видеотрактом монитора и в принципы изготовления ЭЛТ позволило приступить к выпуску изделий с плоским экраном и укороченной трубкой. Модификации с плоским экраном известны под разными торговыми марками — PanaFlat (Panasonic), Flatron (LG Electronics), FD Trinitron (Sony) и др. Уменьшение длины трубки достигнуто за счет увеличения угла отклонения лучей электронной пушки с 90 до 100°. У таких мониторов значительно меньше размер по глубине, т. е. глубина укороченного 19-дюймового монитора равна глубине обычного 17-дюймового монитора.

Важным элементом монитора является его видеоусилитель. Полоса пропускания (верхняя частота) видеоусилителя определяет возможности монитора по максимальному разрешению и частоте кадровой развертки. Видеоусилитель должен иметь такую полосу пропускания, чтобы можно было обеспечить передачу сигналов, генерируемых видеоадаптером, без искажений. Минимально необходимую полосу пропускания достаточно просто рассчитать по необходимым параметрам разрешения. Так, полосы пропускания монитора около 250 МГц достаточно для получения разрешения экрана 1600 x 1200 точек при кадровой частоте 100 Гц.

Начиная с разрешения 1280 x 1024 точек при кадровой частоте 85 Гц и выше для снижения искажений сигналов раньше монитор и видеоадаптер соединяли экранированным коаксиальным кабелем и разъемами BNC. По коаксиальному кабелю через разъемы BNC на монитор поступают отдельные сигналы цветности (R, G, B), вертикальной и горизонтальной синхронизации. В отличие от обычного VGA-кабеля отдельные кабели не влияют друг на друга, и потому изображение получалось четким. В современных мониторах появилась возможность помимо аналогового интерфейса VGA использовать цифровой интерфейс DVI, которым снабжаются также многие видеоадаптеры. Digital Visual Interface (DVI) был разработан группой Digital Display Working (DDWG), включающей достаточно много компаний. Хотя этот стандарт не был принят VESA, DVI имеет очень хорошую перспективу благодаря применению цифрового протокола передачи — TMDS (PanelLink). По сравнению с другими вариантами цифрового интерфейса P&D и DFP, имеющими только один линк, DVI имеет второй линк, который удваивает максимальную пропускную способность. Это позволяет использовать разрешение выше 1280x1024. Дополнительное преимущество DVI заключается в возможности передачи аналогового сигнала, что позволяет подключать LCD- и ЭЛТ-мониторы, использующие обычный вход VGA.

К достоинствам мониторов на ЭЛТ можно отнести качественное изображение при работе с разными разрешениями. Нижней планки в разрешении практически нет, можно использовать даже 320 x 240. Кроме того, мониторы на ЭЛТ более качественно передают цвета (исходя из восприятия их человеком), имеют значительно больший угол обзора, более высокую скорость смены изображения и более качественное сглаживание прямолинейных границ.

Необходимо также учитывать вопросы безопасности использования мониторов, поскольку пользователь, напрягая свое зрение, наблюдает изображение на экране монитора и находится от него на расстоянии 40...50 см, подвергаясь воздействию тех или иных излучений в течение длительного времени. Мониторы на основе ЭЛТ являются наиболее опасными — они излучают широчайший спектр электромагнитного излучения, издают шум, не говоря уже о вреде для глаз при низком качестве изображения.

Каждая страна предъявляет свои требования к безопасности монитора. В России контролирующим органом, который проводит испытания различной аппаратуры с точки зрения безопасности ее использования, является «РосТест». Сертификаты таких организаций говорят об уровне безопасности использования данной техники.

В Швеции были разработаны стандарты безопасности для вычислительной техники — ТСО, которые были приняты во многих странах мира. Первый стандарт обнародован в 1992 г., в нем приведены нормы допустимых электромагнитных излучений, уровня пожарной и электрической безопасности именно монитора. В 1995 г. были разработаны стандарты ко всей персональной ЭВМ и касались они эргономики, экологии, энергосбережения, уровня шума и тепловыделения; требования к излучениям остались прежними, но применялись уже не только к монитору, но и ко всем составляющим ПЭВМ. ТСО'99 предъявил более жесткие требования, чем ТСО'95, практически по всем пунктам. Современные мониторы должны отвечать требованиям ТСО'03, которые не столь значительно отличаются от стандарта ТСО'99.

Жидкокристаллические мониторы. В мониторах на плоских панелях, используемых в ЭВМ, применяются различные технологии получения изображения: на жидких кристаллах (LCD), на плазменных (PDP) или светодиодных элементах (OLED) и др. Ниже рассматриваются только технологии, реально используемые при производстве мониторов.

В основе работы мониторов на LCD лежат оптические свойства жидких кристаллов (ЖК), первые упоминания о которых относятся к 1888 г., когда австрийский ботаник Ф. Райницер обнаружил эти удивительные структуры в ходе своих экспериментов. Однако термин «жидкий кристалл» был предложен немецким физиком О. Леманном, который исследовал их электромагнитные и оптические свойства. По своей природе ЖК представляют собой переходное состояние вещества между твердым и жидким состояниями, при котором сохраняется кристаллическая структура молекул и в то же время обеспечивается текучесть. Под действием электрического поля молекулы ЖК могут изменять свою ориентацию и вследствие этого изменять свойства светового луча, проходящего сквозь них.

Жидкие кристаллы не являются светоизлучающими приборами, они управляют световым потоком от ламп подсветки, проходящим через кристаллы или падающим на них. ЖК-панель имеет многослойную структуру, в которой ключевую роль играют две стеклянные подложки и находящийся между ними слой

ЖК. На подложках проделаны параллельные бороздки, определяющие ориентацию ЖК. Бороздки двух подложек ортогональны друг к другу и на пересечении бороздок образуются ячейки с ЖК. При этом продольные оси молекул самого верхнего слоя ЖК будут расположены под прямым углом по отношению к осям молекул из нижнего слоя. Между этими двумя крайними положениями образуется своеобразная молекулярная спираль с промежуточными ориентациями молекул, которая и дала название технологии — *twisted nematic* (закрученные нематические). Причем каждая ячейка с ЖК располагается между контактами тонкопленочного транзистора (TFT). Именно эта основа TFT и меняет напряжение, под воздействием которого изменяется плоскость поляризации жидкости. За самой пластиной с ЖК расположены одна или две мощные флуоресцентные лампы подсветки и специальные материалы, или световоды, для равномерного распределения освещения по плоскости экрана. Меняя угол поляризации, ЖК изменяют интенсивность проходящего света или прекращают его прохождение в каждой конкретной точке. Каждая такая ячейка соответствует одной точке на экране монохромной ЖК-панели. Сумма всех точек (пикселей) на панели и является максимальным разрешением панели. Цветное изображение образуется, как и в мониторах на ЭЛТ, сочетанием трех основных цветов: красного, синего и зеленого. Эту задачу решает панель цветного фильтра.

Поворот плоскости поляризации светового луча незаметен для глаза, поэтому на панели устанавливают несколько поляризационных фильтров, подложки также являются поляризационными фильтрами. Они пропускают только компоненту светового потока, у которой ось поляризации соответствует заданной, т. е. при прохождении поляризатора степень ослабления светового потока зависит от угла между его плоскостью поляризации и осью поляризатора. В отсутствие напряжения на сегменте углы поляризации света после прохождения ЖК-ячеек и второй подложки совпадают и потому пиксел выглядит прозрачным.

В присутствии электрического поля поворот вектора поляризации происходит на меньший угол, тем самым вторая подложка становится лишь частично прозрачной для светового излучения. Если разность потенциалов такова, что поворота плоскости поляризации в ЖК-ячейках не происходит, то световой луч полностью поглощается вторым поляризатором и экран при освещении на про-свет выглядит черным.

Для управления свойствами ячеек к ним подключают электроды, создающие разные электрические поля в отдельных ячейках экрана. В активной матрице (Active Matrix) каждая ячейка панели подключена к собственному управляющему элементу. В качестве управляющих элементов используются тонкопленочные транзисторы (TFT — Thin Film Transistor), образующие матрицу из строк и столбцов соответственно ячейкам экрана. Активная матрица имеет высокую яркость и большие углы обзора (120... 160°) без ущерба для качества изображения. Время реакции дисплея с активной матрицей в лучших образцах составляет до 8... 10 мс (для пассивной матрицы — около 300 мс). Яркость отдельного элемен-

та изображения остается неизменной весь период демонстрации. Именно поэтому для ЖК-мониторов достаточной считается частота регенерации 60 Гц. ЖК-кристаллы типа Super Twisted Nematic имеют увеличенный с 90 до 270° торсионный угол (угол кручения) ориентации, что обеспечивает лучшую контрастность изображения при увеличении размеров монитора.

Технология *TN+Film* является самой распространенной технологией производства ЖК-матриц, что обусловлено низкой стоимостью производства при приемлемом качестве изображения. Принцип действия TN+Film матрицы основан на использовании в ЖК-ячейках закрученных нематических кристаллов (TN — Twisted Nematic) и покрытием матрицы специальной пленкой с высоким показателем преломления для расширения угла обзора в горизонтальной плоскости. TN+Film-матрицы отличаются невысоким качеством изображения, и одним из путей его повышения стала замена матового покрытия на глянцевое. Родоначальником этой идеи выступила компания Toshiba, технология называлась CASV (Clear Advanced Super View). Сегодня этот термин практически не используется, но подобные технологии есть практически у любого производителя ЖК-матриц. Так, Toshiba сейчас предлагает TruBrite, Sony — X-Brite и X-Black, ASUS — ACE View, IBM — FlexView, Fujitsu — CrystalView. Что интересно, все эти технологии имеют общую идею, но могут использоваться по отношению к различным типам матриц, что приводит к разным результатам.

В чем же причина того, что все производители стали менять свои матовые матрицы на глянцевые? Основной особенностью матовой матрицы является то, что ее коэффициент отражения (или другими словами, усиления) равен единице. Это означает, что, к примеру, яркость изображения никоим образом не усиливается и напрямую зависит от яркостных возможностей непосредственно матрицы. Но если же установить между матрицей и глазами пользователя глянцевую пленку, имеющую коэффициент отражения больше единицы, то в соответствии с законами физики будет создаваться впечатление более яркой и контрастной картинки.

Например, технология компании Sony X-Brite, название которой X-Brite (extended Brite (от bright) — расширенная яркость), реализует это усовершенствование. Формально дисплей с такой матрицей кажется более четким, ярким и контрастным, но только при хорошем освещении. Это субъективное ощущение основано оно на особенностях зрения человека. Относительно технологии TN+Film с X-Brite необходимо отметить, что высокий уровень черного цвета здесь не уменьшился, и это сразу видно при слабом внешнем освещении. Правда, есть и позитивное улучшение — отсутствие рассеяния проходящего света. Но с другой стороны, это приводит к тому, что глянцевая матрица в отличие от матовых матриц отражает любые внешние яркие объекты, как в зеркале. Матрицы на базе этой технологии имеют наименьшее время отклика и приемлемые углы обзора при самой низкой стоимости. Однако данной технологии присущи и существенные недостатки: плохая цветопередача, низкая контрастность, высокий уровень черного цвета и недостаточные углы обзора. Мониторы с использо-

ванием TN+Film-матриц можно рекомендовать для просмотра видеофильмов и компьютерных игр.

Технология IPS (In-Plane Switching) была разработана компаниями NEC и Hitachi, известна также под названием SuperTFT. В соответствии с этой технологией управляющие электроды расположены на одной подложке, а молекулы ЖК поворачиваются единой плоскостью без скручивания в спираль. Матрицы, изготовленные по этой технологии, имеют углы обзора 170° в обоих направлениях, что обусловлено более точным механизмом управления ориентацией ЖК, поскольку они располагаются параллельно друг другу. К тому же обеспечивается более высокая яркость и контрастность до 300:1. Но при этом IPS-матрицы отличаются большим энергопотреблением, значительным временем отклика и очень высокой ценой. Однако на сегодняшний день им нет равных по качеству цветопередачи, и мониторы с этими матрицами можно рекомендовать для работы дизайнеров и обработки цифровых фотографий, где необходима высококачественная визуализация статичных изображений.

Технология MVA (Multi-Domain Vertical Alignment), совместившая особенности технологий, описанных выше, была разработана компанией Fujitsu в 1996 г. В данной технологии электроды размещены на обеих подложках, а сами подложки имеют выступы, разделяющие ЖК на области — домены. Все домены переключаются одновременно, но продольные молекулы в них поворачиваются в противоположных направлениях. За счет усовершенствования способа размещения ЖК (они расположены параллельно друг другу, но под прямым углом к поляризационному фильтру) получился некий промежуточный вариант, который обладает достоинствами IPS-матриц (высокие яркость и контрастность до 500:1, еще большие углы обзора), а время отклика близко к значениям TN+Film-матриц. Модификацию этой технологии под названием PVA, которая отличается еще большей яркостью и контрастностью, использует компания Samsung. Аналогичная разработка под названием ASV (Advanced Super View) есть и у компании Sharp.

Что касается будущего ЖК-матриц, то наиболее перспективной считается технология LTPS (Low Temperature Poly Silicon — низкотемпературный поликристаллический кремний), которая по всем параметрам превосходит все ныне существующие технологии, но очень дорогая, трудоемкая и время ее массового использования еще не определено.

Параметры плоскостельных мониторов. Важнейшим параметром плоскостельных мониторов является стандартное (иногда называемое максимальным) разрешение. Оно соответствует числу пикселей по горизонтали и вертикали. Именно в стандартном разрешении ЖК-монитор воспроизводит изображение наиболее качественно. Разрешение определяется размером ячеек и диагональю панели. Сейчас производятся панели с ячейками размером 0,248...0,3 мм и, как правило, длина диагонали экрана определяет его стандартное разрешение. Так, для ЖК-монитора с длиной диагонали равной 15", стандартное разрешение равно 1024x768, а для 17" — 1280 x 1024. Для монитора на ЭЛТ можно установить разрешение больше стандартного (рекомендуемого) для данного размера диаго-

нали экрана, а на ЖК-мониторе — нельзя. Как правило, в ЖК-мониторах предусмотрена возможность использовать разрешение более низкое, чем стандартное. Обычно применяют метод растяжения (Expansion), основанный на интерполяции изображения с низким разрешением на всю площадь экрана. Очевидно, что при этом существенно снизится качество изображения, поскольку интерполяция ухудшает резкость изображения и вносит цветовые искажения.

Яркость и контрастность определяют комфортность работы с ЖК-монитором. Средним считается значение яркости $250...350$ кд/м², качественные панели поддерживают более высокие значения. Контрастность ЖК-монитора определяется отношением яркости самого яркого белого и самого темного черного цветов. Хорошим контрастным соотношением считается 350:1. Цветовой охват современных ЖК-панелей достигает 16,7 млн цветов. Угол обзора (по вертикали и горизонтали) характеризует зону восприятия изображения на экране без существенных искажений. Нормальным считается угол обзора по горизонтали $150...160^\circ$, по вертикали $120...130^\circ$.

Слабым местом ЖК-мониторов остается время отклика (скорость переключения между режимами черный — белый — черный), которое составляет 15...30 мс. Этот параметр характеризует максимальное быстродействие, а в режимах пониженной яркости (менее 100 %) оно увеличивается в 5-7 раз, что приводит к смазыванию быстро меняющихся изображений.

Таким образом, к достоинствам ЖК-мониторов можно отнести малую глубину, действительно плоское изображение (без геометрических искажений), высокие значения яркости, низкое энергопотребление, отсутствие электромагнитных излучений. Однако им присущи четыре существенных недостатка: высокая цена искажения в цветопередаче, единственный режим разрешения, обеспечивающий хорошее качество, малые углы комфортного обзора.

Плазменные мониторы. Основой для создания плазменных экранных матриц (Plasma Display Panels) стали процессы, происходящие в обычных лампах дневного освещения. Плазменные мониторы состоят из полой стеклянной панели, заполненной газом. На поверхность внутренней стороны стенок выведены микроскопические электроды, образующие две симметричные матрицы, а снаружи эта конструкция покрыта слоем люминофора. Когда на контакты подается ток, между ними возникает крошечный разряд, который заставляет светиться (в ультрафиолетовой части спектра) располагающиеся рядом молекулы газа. Возникшее свечение освещает участок люминофора, как и в обычных мониторах на ЭЛТ.

Панели, изготовленные по этой технологии, отличаются высокой яркостью и контрастностью, а также малым временем отклика; угол обзора плазменных панелей практически равен 180° , а толщина — менее 10 см. Однако в качестве монитора плазменные панели используются достаточно редко, хотя и являются идеальным средством отображения коллективного пользования в учебном классе или в домашнем кинотеатре. Это связано с техническими особенностями плазменных панелей. Во-первых, габариты — при толщине в 8...10 см минимальная длина диагонали плазменной панели с приемлемым разрешением

равна 32". Себестоимость 19"-панели немногим меньше 40", поэтому ни один производитель не решился на выпуск панелей даже с диагональю, равной 21". Во-вторых, сравнительно небольшой срок службы плазменной панели. Недаром на обратной стороне многих моделей установлен почасовой счетчик, отсчитывающий суммарное время работы. Причем очень быстро такая панель выгорает, если часто и подолгу просматривать статические изображения. И главное обстоятельство, останавливающее рядового покупателя, — очень высокая цена.

Таким образом, плазменные панели практически не имеют конкурентов в сегменте больших диагоналей. Их успешно используют в местах скопления людей: в аэропортах и на железнодорожных вокзалах, в казино и ресторанах; достойно выглядят они и в небольших демонстрационных залах компаний, но для использования в качестве мониторов пока не пригодны.

Мониторы OLED и DEL. Большой интерес к средствам отображения графической информации постоянно стимулирует интенсивные научные исследования в этой области. В результате этого каждые 3-4 месяца объявляется о новой разработке, готовой перевернуть «мониторный мир», появляются новые технологии, которых становится все больше и больше. Из этого множества новинок наиболее близкими к массовому производству являются технологии OLED и DEL.

Органический электролюминесцентный монитор OLED представляет собой монолитный тонкопленочный полупроводниковый прибор, который излучает свет, когда к нему приложено напряжение. OLED состоит из ряда тонких органических пленок, которые заключены между двумя тонкопленочными проводниками. Рабочее напряжение OLED составляет всего лишь 3...10 В. Цвет, эффективность и интенсивность излучения приборов OLED зависят от использованных органических материалов, которыми определяется многообразие воспроизводимых дисплеем цветов. Сегодня основное внимание "разработчиков приборов OLED направлено на создание материалов для полноцветных приборов OLED. В приборах OLED используются два класса органических материалов: микромолекулы (sm-OLED) и полимеры (PLED). Эти две системы имеют несколько различий. Сегодня мониторы sm-OLED опережают мониторы PLED по эффективности и сроку службы.

Таким образом, OLED — тонкопленочное устройство со светоизлучающей поверхностью, которая образована множеством одновременно излучающих свет ячеек на одной подложке. Причем эти ячейки могут быть изготовлены либо методом напыления, либо методом струйной печати. Для создания дисплея с произвольным структурированием можно применить обычную литографию. Другими словами, OLED имеют значительные преимущества в технологии формирования структуры.

Собственно говоря, органические светодиодные панели OLED (Organic Light Emitting Diode) уже применяются в производстве CD- и MP3-плееров, а также мобильных телефонов. В этом активное участие принимает компания Samsung. Основанная на светодиодной технологии, OLED устраняет большин-

ство недостатков ЖК-панелей. Как сообщают разработчики, угол обзора в OLED-панелях составляет более 160° , а время отклика достигает 10 мс, что возможно лишь в самых совершенных ЖК-мониторах. Яркость и контрастность этих устройств превышает показатели мониторов на ЭЛТ. Толщина и энергопотребление OLED-панелей ощутимо меньше, что делает их привлекательными для применения в портативной технике: КПК, ноутбуках и т. п. Кроме того, эти приборы могут выдерживать немалые механические и температурные нагрузки, а стоимость производства ниже, чем для LCD.

Почти одновременно с запуском OLED-панелей в массовое производство было объявлено о завершающей стадии разработки еще более интересной технологии — DEL (Dielectric ElectroLumesceny), работа которой основана на особенностях люминесцентного фосфора, излучающего свет под воздействием электромагнитного поля. Контрастность такой панели еще выше, угол обзора уже 170° , а время отклика до 2 мс. Но самое главное — это высокое качество цветопередачи, которое не уступает мониторам на ЭЛТ и позволяет использовать DEL-мониторы в профессиональных целях.

Производители плоских панелей оказались в некоторой растерянности: чему отдать предпочтение, какая технология перспективна и получит большее распространение в ближайшем будущем. Можно предположить, что каждая технология займет свою нишу. Ученые, расхваливая OLED, утверждают, что добьются минимальной толщины и возможности наносить такой дисплей практически на любую поверхность, что в совокупности со способностью работать в сложных условиях позволит в недалеком будущем увидеть его, например на стекле автомобиля, подключенным к бортовому компьютеру или на рукаве обычной куртки, выполняющим сразу несколько функций: часов, мобильного телефона и карманного персонального компьютера. При всех этих невероятных качествах они обещают, что стоимость производства будет на 30..40 % ниже, чем для ЖК-мониторов.

Трехмерные мониторы. Однако, несмотря на удивительные возможности новейших технологий и открывающиеся в связи с этим перспективы, предложен принципиально новый подход — трехмерные или 3D-мониторы. В результате его реализации изображение на вашем мониторе начинает приобретать объем и «выходить» из экрана, причем для этого не нужно надевать очки или принимать галлюциногены. Все это 3D-монитор, передающий трехмерное изображение в двух плоскостях. Основывается это чудо на принципе стереоскопии: угол зрения каждого глаза человека разный. Создавая два минимально отличающихся изображения и располагая их таким образом, чтобы каждый глаз видел свое, инженеры и добились эффекта трехмерности. Зрительного эффекта каждая компания-разработчик достигает своим способом на платформе TFT.

На сегодняшний день уже несколько компаний налаживают выпуск 3D-мониторов. Фирма Sharp представила жидкокристаллический 15" 3D-монитор, а компания Kodak анонсировала свою технологию трехмерного дисплея, превосходящую все известные по яркости. Немецкая компания АСТ Kern уже продает свои трехмерные дисплеи. Однако стоимость 3D-мониторов сегодня в несколько

раз превышает стоимость обычных ЖК-мониторов. Такой монитор незаменим для научных исследований, медицины, проектирования и других задач. Естественно, для новой технологии потребуется иное программное обеспечение, но при развитии рынка 3Б-мониторов оно будет разработано в кратчайшие сроки.

Преимущества и недостатки 3Б-мониторов представлены в табл. 2.2.

Таблица 2.2

Технология	Недостатки	Преимущества
ЭЛТ	Морально устарела, мерцание, масса и габаритные размеры	Большие углы обзора, цветопередача, скорость реакции точки
ЖК	Скорость реакции, цветопередача, углы обзора	Яркость, масса, габаритные размеры, дизайн, перспективы развития, большие диагонали
PDP	Вес, большие размеры точки, небольшой срок службы, высокая цена	Яркость, контрастность, время реакции точки
OLED	Недостаточно отработанная технология, малые диагонали, высокая цена	Время отклика, яркость, контрастность, углы
3D	Высокая цена, технологическая сложность, углы обзора	Улучшенное восприятие изображения

2.2.6. Проекторы

Проектор, подключаемый к компьютеру параллельно с монитором или вместо него, традиционно является основным устройством отображения графической информации коллективного пользования. Имея небольшие габаритные размеры по сравнению с монитором, он превосходит любую плазменную или ЖК-панель по размерам создаваемого изображения. Стоимость проекторов неуклонно снижается, и сегодня можно купить подходящую модель даже для домашнего кинотеатра.

В проекторе нельзя выделить самый главный блок. Все его компоненты оказывают значительное влияние на качество изображения, и все они так или иначе совершенствуются со временем. Наибольшее внимание уделяется технологии формирования изображения, однако качество оптики и электронные системы могут оказать на изображение не меньшее влияние.

Оптические системы в проекторах зачастую достаточно сложны, а качественную оптику в мире производит не так много компаний, и иногда изображение даже в проекторах одного производителя может создаваться объективами с разными логотипами — от Fuji до Carl Zeiss. Электроника приобретает тем большее значение, чем большее разрешение требуется от проектора и в связи с использованием телевидения высокой четкости (HDTV). Важной частью проектора является и лампа, создающая световой поток, равномерный по световым свойствам. При выборе проектора обязательно следует учитывать срок службы лампы и ее высокую стоимость.

Проекторы на ЭЛТ. Технология формирования изображения на ЭЛТ — одна из первых и наиболее отработанная. Но принципы ее работы в проекторах существенно отличаются от мониторов или домашних телевизоров. Во-первых, в проекторе имеется сразу три электронно-лучевых проекционных трубки. Каждая из них отвечает за свой цвет — красный, синий или зеленый, которые и формируют цветное изображение в соответствии с моделью RGB. Нужный цвет обычно формируется цветным фильтром, стоящим за трубкой. Световой поток из трех основных цветов проходит через относительно несложную систему линз и фокусируется на экране, создавая полноцветную картинку. Такие проекторы имеют отличную цветопередачу — за десятилетия технология производства трубок достигла совершенства. Благодаря синтетическому характеру каждого участка изображения на картинке отсутствует видимое зерно. Проекторы на ЭЛТ отлично передают и черный цвет, что сложно обеспечить во многих других системах.

Главными недостатками проекторов на ЭЛТ являются большие габаритные размеры и масса — каждая трубка имеет диаметр более 10 см и требует мощного охлаждения. Кроме того, качественное изображение формируется путем тщательного сведения трех картинок на одном экране, что исключительно сложно в настройке и не позволяет быстро переместить проектор ни на сантиметр после настройки. Цена таких проекторов чрезвычайно высока — выше 10 тыс. долл. Недостатком также является не самая высокая яркость таких систем, поэтому их лучше использовать в затемненных помещениях. Однако для качественного домашнего кинотеатра такие проекторы до сих пор остаются одними из лучших.

Лазерные проекторы. В лазерных проекторах используется способ формирования изображения, подобный используемому в ЭЛТ. Источниками света в них являются три (иногда больше) лазера, яркость излучения которых модулируется в соответствии с видеосигналом каждого цвета. Матрица лазеров формирует три луча красного, синего и зеленого цвета, которые смешиваются при формировании цветного изображения. Изображение создается очень сложной системой фокусировки и развертки, в которой находится специальная система зеркал, осуществляющая развертку по горизонтали и по вертикали. Реалистичное изображение формируется при этом практически на любой, в том числе и неровной, поверхности, а его характеристики достаточно высоки. Начиная с 2000 г., когда началось серийное производство таких проекторов, качество изображения возросло, но все еще остаются проблемы с цветопередачей, хотя изображение и обладает впечатляющими показателями контрастности и яркости.

Лазерные проекторы пока остаются в большей степени дорогими профессиональными инструментами — они имеют большие размеры и потребляют много энергии. Однако их конструкция позволяет разделить излучающую батарею лазеров с большим тепловыделением и проецирующую часть. Кроме того, срок службы лазера заметно превосходит срок службы лампы традиционных проекторов, а энергии при сопоставимых параметрах яркости расходуется меньше. Главным достоинством лазерных проекторов является их способность

создавать изображения на огромных экранах — диагональ экрана может достигать нескольких десятков метров.

Существуют еще и такие малоизвестные устройства, как лазерные ЭЛТ, в которых лазерный луч, падая на люминофор, вызывает яркое свечение экрана, но они мало распространены и находятся на стадии разработки коммерческих прототипов (такие разработки ведутся и в России).

Проекторы на ЖК-матрицах. Одной из самых отработанных является технология, применяющаяся в проекторах — ЖК-матрица, работающая «на просвет». Это самая первая и самая дешевая технология до сих пор остается самой распространенной — проекторы, созданные на основе одной LCD-матрицы, распространены в образовательных учреждениях, в презентационных комнатах при показе статичных слайдов и т. п. В данном случае свет лампы, проходя сквозь LCD-матрицу как через диафильм или киноплёнку, а затем через объектив, пронизывает множество слоев матрицы и цветового фильтра. В готовом изображении, проецируемом на экран, часто присутствует эффект «мозаичности». Кроме того, проблема формирования черного цвета проявляется здесь в полной мере. Поскольку ЖК-матрицы работают на просвет, то создать абсолютно непрозрачный участок в условиях яркого и мощного освещения они попросту не способны. Поэтому часто черный цвет на экране больше похож на серый. По этой же причине ЖК-матрицы с трудом справляются с полутонами — количество градаций серого цвета не так велико, как это необходимо.

Более качественных результатов позволяет добиться технология, в которой вместо одной ЖК-матрицы используются три таких матрицы.

Проекторы на трех ЖК-матрицах. Три ЖК-матрицы позволяют создать изображение гораздо лучшего качества, чем при использовании одной ЖК-матрицы, за счет разделения светового потока и прохождения его только через одну ЖК-панель, а не через три цветовых фильтра последовательно. Это гарантирует большую яркость и лучшее качество картинки, особенно в плане четкости. Система дихроичных зеркал разделяет свет на три составляющих цвета, пропуская каждый через свою ЖК-матрицу, а потом призма собирает все три изображения в одну картинку. Однако и в этом случае сохраняется проблема черного цвета — он опять оказывается скорее серым, чем черным.

Проекторы на трех ЖК-матрицах обладают некоторым преимуществом перед однокристальными DLP-проекторами, в которых цвет создается путем последовательного наложения цветов. В ЗЖК-проекторах цвет создается одновременно и без использования движущихся частей.

Проекторы на микрозеркальной технологии DLP. Самой бурно развивающейся технологией, на которой строятся современные проекторы, можно считать микрозеркальную, или DLP-технологию. При ее использовании свет мощной лампы отражается от специального чипа (DMD — Digital Mirror Device), содержащего тысячи микрозеркал, каждое из которых отвечает за свой пиксел изображения. Матрица с зеркалами очень миниатюрна, обычно около 1", и именно на нее и на систему управления приходится большая часть стоимости

таких проекторов и телевизоров. Каждое из миллионов микрозеркал управляется индивидуально, и в итоге создается очень четкое и ясное изображение, лишенное мерцания и артефактов, присущих ЖК. Разработчиком этой технологии и поставщиком всех DMD-матриц и схем управления ими является американская компания Texas Instruments.

Свет на микрозеркала DMD-матрицы попадает через специальный вращающийся светофильтр, имеющий три или четыре грани. На трехцветном светофильтре они окрашены в красный, зеленый и синий цвета, а на четырехгранном добавлена прозрачная грань, оказывающаяся полезной тогда, когда имеются большие неокрашенные участки изображения. Скорость смены всех сочетаний настолько высока, что человеческим взглядом воспринимается только цельное изображение, очень яркое и четкое. В последнее время приобретают популярность системы, в которых применяется цветовое колесо с шестью или семью сегментами — качество изображения от этого заметно улучшается и пропадает эффект радуги, возникающий на резких цветовых границах изображения.

Пикселизация изображения, присущая ЖК-технологии, присутствует и в DLP-проекторах, хотя и в заметно меньшей степени. Это связано с промежутками между элементами, формирующими пиксел. Если в ЖК-матрице на неработающие участки матрицы между точками, которые не участвуют в формировании изображения, приходится до 30 % площади (в старых матрицах доходило и до 40 %), то при DLP-технологии — не более 10... 15 %. Учитывая, что эта технология работает не на просвет, а на отражение, некоторые проблемы у таких проекторов могут быть с формированием белого цвета, а также с несвоевременным срабатыванием зеркал.

Проекторы и проекционные телевизоры на базе этой технологии наиболее компактны, к тому же позволяют получать световой поток до значения 10 тыс. лм. Существуют разновидности микрозеркальной технологии, несколько отличающиеся по своим принципам от DLP, например iMOD или интерференционные дисплеи, но они пока не отработаны до конца, хотя имеют отличные перспективы. Например, в технологии iMOD отсутствуют цветные фильтры, и она гораздо менее энергоемка.

Проекторы на технологии D-ILA (LCOS). Технология D-ILA (Digital Direct Drive Image Light Amplifier) является коммерческим развитием технологии LCOS (Liquid Crystal on Silicon — ЖК на кремнии) и активно развивается разными производителями, в том числе и компанией JVC, которая выпускает на ее основе проекционные системы. Изображение при использовании этой технологии формируется ЖК, однако работает она не на просвет, как обычные ЖК-матрицы, а на отражение, и иногда технология называется отражающими ЖК-панелями. Главное ее отличие от обычной ЖК-матрицы состоит в том, что вся электроника расположена за слоем ЖК под отражающими электродами, а не между ячейками. Это обеспечивает лучший коэффициент заполнения — изображение формируется на большей площади матрицы, и незадействованной остается минимальная площадь. Световой поток формируется несильным источ-

ником света, а потом усиливается специальной лампой, отчего и происходит название технологии.

В результате граница между пикселями практически незаметна, светоотдача матрицы возрастает, а ее нагрев уменьшается. Теоретически контрастность самой матрицы может достигать 2000:1. Оптическая схема, подобная используемой в обычных ЖК-проекторах, и три матрицы D-ILA позволяют получить полноцветное изображение. Формирование цветов происходит по-разному — так, например, JVC создала голографический фильтр, другие производители используют вращающуюся призму, разделяющую цвета, существуют также трехчиповые системы, в которых нет движущихся частей.

Технология D-ILA сегодня активно развивается, как и технология трех ЖК-матриц, и позволяет получить изображение, по своим характеристикам близкое к получаемому проекторами на ЭЛТ, т. е. хорошо воспринимаемое глазом человека. С черным цветом эти проекторы справляются также отлично, к тому же эта технология позволяет добиваться очень высокого разрешения. До сих пор такие проекторы остаются достаточно тяжелыми и дорогими, однако над началом их производства работают многие компании, и перспективы у этой технологии хорошие.

Вопросы для самоконтроля

1. Какие задачи решают технические средства в составе графической системы?
2. Перечислите основные параметры компьютеров.
3. Назовите основные классы ЭВМ и поясните их различия.
4. Поясните состав и назначение устройств графической рабочей станции.
5. Какова структура ЭВМ? Поясните назначение отдельных устройств.
6. Перечислите основные особенности наборов мультимедийных инструкций центрального процессора.
7. Назовите основные виды запоминающих устройств и поясните их назначение.
8. Поясните, как в ЭВМ организуется обмен с периферийными устройствами?
9. Охарактеризуйте роль и место графической подсистемы в структуре ЭВМ.
10. Перечислите области применения трехмерной КГ.
11. Назовите основные параметры графических адаптеров.
12. Поясните устройство графического адаптера и назначение основных его элементов.
13. Что понимают под технологией SLI и CrossFire?
14. Какова структура графического процессора? Поясните назначение его отдельных блоков.
15. Поясните устройство вершинного процессора и решаемые им задачи.
16. Поясните устройство пиксельного процессора и решаемые им задачи.
17. Что такое встроенный ускоритель графики?
18. Дайте сравнительную характеристику программных интерфейсов видеоадаптеров.
19. Охарактеризуйте основные особенности работы ЗО-конвейера.
20. Что понимают под вершинными и пиксельными шейдерами?
21. Сравните технологии наложения и обработки текстур.

2.2. Графическая подсистема ЭВМ

22. Чем различаются основные методы MIP-mapping?
23. Дайте сравнительную характеристику методов фильтрации текстур.
24. Охарактеризуйте основные способы улучшения изображения.
25. Какова структура и назначение графического конвейера?
26. Перечислите основные задачи, решаемые на различных этапах работы графического конвейера.
27. Сравните поколения графических процессоров.
28. Поясните устройство монитора на ЭЛТ и назначение его отдельных блоков.
29. Назовите особенности устройства плоскopianельных мониторов.
30. Сравните основные технологии формирования изображения, используемые в плоскopianельных мониторах.
31. Дайте сравнительную характеристику плоскopianельных мониторов и мониторов на ЭЛТ.
32. Поясните особенности устройства проекторов.
33. Сравните основные технологии формирования изображения, используемые в проекторах.

3. МАТЕМАТИЧЕСКИЕ МОДЕЛИ ГЕОМЕТРИЧЕСКИХ ОБЪЕКТОВ

Рассматриваются математические модели кривых и поверхностей, нашедшие широкое применение в современных графических пакетах и системах автоматизированного проектирования. Подробно обсуждаются свойства бикубических параметрических кривых и поверхностей, на которых основывается значительная часть компьютерных приложений, связанных с рисованием и черчением. Многие проектные ситуации требуют представления объекта в виде трехмерного тела, у которого в явном виде заданы все внутренние и граничные точки. Твердотельное моделирование — это самостоятельное направление компьютерной графики, которое предлагает средства математического описания трехмерных объектов. В главе описываются самые известные и востребованные твердотельные модели: регулярные булевские операции, граничное представление (B-ger), конструктивная геометрия твердых тел (CSG), а также самые важные основные модели пространственного разбиения.

3.1. Представление кривых и поверхностей

В естественном природном окружении очень редко встречаются геометрические объекты с нерегулярностями и значительными нарушениями гладкости. По этой простой причине большая часть компьютерных графических программ предназначена для создания гладких кривых и поверхностей. Поверхности, порождаемые системами автоматизированного проектирования, компьютерные шрифты и рисунки, анимационные траектории — это все примеры гладких или почти гладких геометрических объектов.

Можно выделить два принципиально разных случая создания геометрических объектов в компьютерной среде: описание существующей формы или поверхности и создание геометрии «с нуля». В первом случае точное математическое описание формы по различным причинам может отсутствовать. В процессе компьютерного моделирования его часто заменяют сочетанием простейших геометрических объектов: плоскостей, сфер, конусов и других элементарных форм, имеющих точное и лаконичное математическое описание.

Во втором случае пользователь создает геометрию объекта, не отталкиваясь от существующего оригинала или прототипа. Современные компьютерные программы предлагают для формообразования множество различных технических

приемов. Это может быть некоторый свободный интерактивный процесс или операции с формальными объектами — в конечном итоге будет создано абстрактное описание, по которому CAD-система или графический редактор синтезируют визуальное представление на финальных этапах процесса проектирования.

Самой простой формой представления поверхностей является полигональная сетка (polygon mesh) — совокупность плоских фрагментов, ограниченных многоугольниками и соединенных между собой. В такой форме естественным образом представляется множество самых разнообразных объектов: здания, мебель, промышленные интерьеры и пр. С некоторыми потерями при помощи полигональных сеток можно описать гладкие поверхности и поверхности природного происхождения.

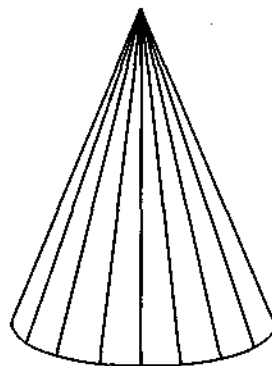


Рис. 3.1. Аппроксимация гладкой поверхности при помощи полигональной сетки

На рис. 3.1 показана аппроксимация поверхности конуса при помощи простой полигональной сетки. Очевидные недостатки такого представления можно отчасти компенсировать посредством уменьшения размеров ячеек сетки. Естественной платой за это решение будет увеличение размеров представления и повышение затрат на вычисления и обработку в компьютере. Кроме того, при значительном увеличении размеров оригинала сеточная структура и изломы в местах сочленения могут стать заметными для наблюдателя.

Параметрическая полиномиальная кривая (parametric polynomial curve) представляет собой геометрическое место точек, которое задается тремя полиномами и параметрами t , x , y , z . Значения этих параметров выбираются таким образом, чтобы трасса кривой пролегла по требуемому пути. В теоретических исследованиях по машинной графике используются различные виды и способы представления кривых. В этой книге рассмотрен только самый употребительный случай, когда кривые задаются полиномами третьей степени. Такие объекты часто называют кубическими кривыми.

Часто используется способ описания поверхностей, полное английское название которого (parametric bivariate polynomial surface patches) переводят на русский как куски полиномов. При этом координаты точек на криволинейной (в общем случае) поверхности задаются при помощи трех двухмерных полиномов — по одному на каждую координату x , y , z . В таком представлении границей фрагментов служат полиномиальные кривые. Очевидно, что для точного приближения сложных криволинейных поверхностей в общем случае понадобится намного меньше полиномиальных кусков, чем полигонов. Однако алгоритмы обработки таких фрагментов технически сложнее, чем для многоугольников. Самый популярный частный случай такого представления использует два кубических полинома, а соответствующие формы называются бикубическими поверхностями (bicubic surfaces).

Квадратичные поверхности (quadric surfaces) задаются уравнением вида $f(x, y, z) = 0$, где f — квадратичный полином от аргументов x, y, z . Это обычный способ описания простейших трехмерных форм: сферы, эллипсоида и цилиндра.

3.1.1. Полигональные сетки

Полигональная сетка представляет собой совокупность вершин, ребер и полигонов, соединенных таким образом, что каждое ребро принадлежит не более чем двум многоугольникам. Ребра ограничиваются двумя вершинами, а полигоны замыкаются цепочкой из последовательно соединенных ребер. Каждое ребро служит границей двух смежных полигонов, а каждая вершина принадлежит, по крайней мере, двум ребрам.

Существует несколько способов описания полигональных сеток со своими достоинствами и недостатками. Часто даже одно графическое приложение использует в своей работе альтернативные способы описания сеток, например одно для внутреннего представления и расчетов, а другое для визуализации.

Основными критериями для оценки эффективности описания полигональных сеток служат время и объем памяти, необходимой для ее хранения. Если оценивать по затратам времени, то наиболее ресурсоемкими будут топологические операции: нахождение общей границы двух многоугольников, поиск соединительной вершины ребер, а также отображение сетки и поиск ошибок сеточного представления. Понятно, что чем проще описываются связи между элементами сеточного представления, тем быстрее будут выполняться основные операции над геометрическими объектами.

3.1.2. Представление полигональных сеток

Рассмотрим несколько простых методов представления полигональных сеток: прямой способ, указатели на список вершин и на список ребер.

Прямой способ описания

Этот способ описания представляет собой точное описание всех многоугольников в виде списка координат вершин:

$$P = ((x_1, y_1, z_1), (x_2, y_2, z_2), \dots, (x_n, y_n, z_n)).$$

Для экономии вычислительных ресурсов вершины перечисляются в том порядке, который они занимают при обходе границы многоугольника. Поэтому каждая пара соседних вершин списка задает граничное ребро, кроме того, ребро существует между первой и последней вершинами списка. Это очень экономная форма хранения для одиночных многоугольников. Для сеток данное утверждение несправедливо, поскольку приходится хранить множественные дубликаты

координат вершин. Более того, для некоторых операций невозможно использовать явное представление смежных ребер и вершин. Например, чтобы перетащить вершину и рассчитать новое положение всех смежных объектов интерактивно, требуется найти все полигоны, которые примыкают к данной вершине. Эта операция требует сравнения координатных троек одного полигона со всеми аналогичными объектами других полигонов. Более эффективным способом реализации этой операции является предварительная сортировка N координатных троек. В самом лучшем случае трудоемкость такой процедуры оценивается выражением $N \log_2 N$. Кроме того, накопление ошибок вычисления может привести к некорректному вычислению координат полигонов и фатальным ошибкам.

Данный способ представления графики влечет за собой ряд проблем в процессе визуализации на экране, печати на принтере и выводе на плоттер или подобных устройствах. Поскольку ребра могут принадлежать двум полигонам, то в общем случае задачу отсечения ребер приходится решать дважды. В процессе рисования и печати объекты с кратной принадлежностью будут прорисовываться несколько раз, если не принять специальных мер. Заметную неустойчивость работы в этом смысле демонстрируют так называемые растеризаторы — устройства, переводящие векторную графику в растровую.

Список вершин

Большие возможности дает способ описания полигональных сеток, называемый списком вершин. В этом случае информация о сетке хранится в виде перечня координат вершин, в котором каждая вершина записывается один раз в виде тройки координат:

$$V = ((x_1, y_1, z_1), \dots, (x_n, y_n, z_n)).$$

Каждый многоугольник задается множеством порядковых номеров вершин в списке вершин. Например четырехугольник, опирающийся на вершины с номерами 3, 7, 12, 35, будет представлен в виде

$$P = (3, 7, 12, 35).$$

На рис. 3.2 приведен пример описания полигональной сетки в виде списка вершин. Эта простейшая сетка состоит из двух полигонов P_1 и P_2 . Первый из них задается вектором $(1, 2, 3)$, второй описывается тройкой $(1, 3, 4)$. Это представление обладает несколькими несомненными преимуществами по сравнению с явным описанием полигональных сеток в виде списка многоугольников. Во-первых, здесь достигается значительная эконо-

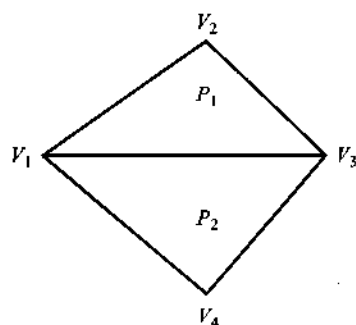


Рис. 3.2. Представление полигональных сеток в виде списка вершин:

$$v = (v_1, v_2, v_3, v_4); \\ P_1 = (1, 2, 3); P_2 = (1, 3, 4)$$

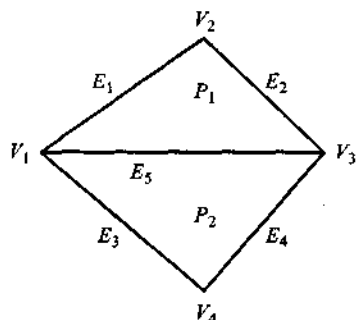


Рис. 3.3. Представление полигональной сетки в виде списка ребер:

$V=(V_u V_v, V_d=(x_u y_i, z_i),$
 $\dots, (x_d, y_d, z_d); E_1=(V_1, V_2, P_1);$
 $E_2=(V_2, V_3, P_1); E_3=(V_3, V_4, P_2);$
 $E_4=(V_4, V_1, P_2); E_5=(V_1, V_3, P_2);$
 $P_1=(E_1, E_2, E_5);$
 $P_2=(E_3, E_4, E_5)$

Список ребер

При задании полигональной сетки в виде списка ребер базовую информацию дает список вершин, но многоугольники описываются ссылками на список ребер, в котором каждое из них упоминается только один раз. В свою очередь, ребра представляются в виде пары граничных вершин и

перечня смежных многоугольников, число которых не может превышать два. В общем случае такое описание имеет следующий вид:

$$P = (E_i, E_j, \dots, E_n),$$

где E_i — описание ребра,

$$E_i = (V_k, V_l, P_h, P_n).$$

Здесь V — имена граничных вершин ребра; P — наименования смежных полигонов или полигона.

На рис. 3.3 показан пример простой полигональной сетки и ее представление в виде списка ребер. Границы многоугольников задаются в виде перечисления всех смежных ребер. Это позволяет избежать проблем избыточной прорисовки и различных трудностей, которые возникают в процессе трансформации избыточных описаний. Упрощается также и визуализация полигонов с заполнением.

В некоторых случаях, например при описании сложных трехмерных текстурированных объектов, отдельные ребра могут быть смежными с более чем двумя полигонами. Данное описание легко модифицируется на этот случай, поскольку позволяет учесть любое число смежных многоугольников:

$$E = (V_1, V_2, P_1, P_2, \dots, P_n).$$

Следует отметить, что ни в одном из рассмотренных описаний (явное описание, список вершин, список ребер) задача определения ребер, смежных задан-

ной вершине, не имеет простого решения. В любом случае для ее решения следует просмотреть все наличные ребра.

Известны модели, где данные о смежности вносятся непосредственно в описание полигональной сетки, что делает такое представление значительно более громоздким. Структура крыльевых ребер, рассмотренная в подразд. 3.4.5, посвященном твердотельному моделированию, дает частичное решение задачи, сохраняя компактность модели.

3.1.3. Согласованность полигональных сеток

Полигональные сетки иногда синтезируются в результате некоторой интерактивной процедуры или динамического взаимодействия компьютера с оператором, например в процессе рисования на графическом планшете. В подобных случаях ошибки представления становятся почти неизбежными. Чтобы добиться согласованности элементов сетки, целесообразно выполнить простые проверки этой структуры: замкнутость всех полигонов, принадлежность ребер, по крайней мере, одному многоугольнику и смежность вершин хотя бы двум ребрам. Кроме этих универсальных ограничений некоторые графические приложения могут накладывать дополнительные требования на структуру и топологию сетки. Это могут быть требования связности, планарности, отсутствие разрывов и пр.

Если сравнить между собой три ранее рассмотренных способа представления полигональных сеток, то окажется, что явное описание лучше всего приспособлено для выполнения проверок на согласованность. Это объясняется высокой избыточностью данной модели.

Существует очень лаконичный программный код, предназначенный для проверки принадлежности ребер полигонам. Достаточно легко устанавливается и ликвидируется избыточность (кратное присутствие ребер и вершин на границе одного многоугольника). Разработаны эффективные процедуры для определения связности и идентификации отверстий.

3.1.4. Уравнения плоскости

В процессе обработки полигональных сеток часто требуется определить уравнение плоскости, на которой лежит выбранный многоугольник или их совокупность. В некоторых случаях это уравнение вытекает в явном виде из самой процедуры построения сетки. В противном случае его можно восстановить по координатам трех точек полигона. Напомним классическое уравнение плоскости:

$$Ax + By + Cz + D = 0.$$

Коэффициенты этого уравнения задают нормаль $[A, B, C]$ к данной плоскости. Точки P_1, P_2, P_3 , принадлежащие плоскости, позволяя вычислить нормаль как векторное произведение $P_1P_2 \times P_1P_3$ (или $P_2P_3 \times P_2P_1$ и т. д.), где P_i — радиус-векторы, соединяющие начало координат с точками плоскости.

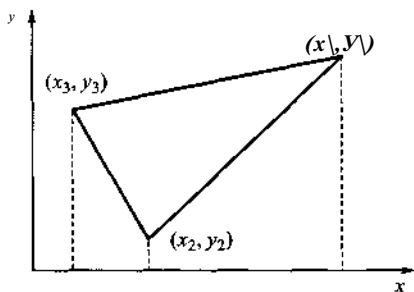


Рис. 3.4. Вычисление коэффициента уравнения плоскости по проекции:

$$C = -\frac{1}{2}((y_1 + y_2)(x_2 - x_1) + (y_2 + y_3)(x_3 - x_2) + (y_3 + y_1)(x_1 - x_3))$$

Очевидно, что коэффициенты A , B , C уравнения плоскости зависят от проекций полигона на координатные плоскости (x, y) , (x, z) , (y, z) . Например, если полигон расположен параллельно плоскости (x, y) , то $A = B = 0$, а его проекции на координатные плоскости (x, z) , (y, z) равны нулю.

Пусть известна проекция полигона на координатную плоскость (x, y) . По этим данным можно определить коэффициент C уравнения плоскости:

$$C = -\frac{1}{2} \sum_{i=1}^n (y_i + y_{i+1})(x_{i+1} - x_i).$$

Легко заметить, что это уравнение описывает алгебраическую сумму площадей трапеций, образованных вершинами и их проекциями на ось x . По аналогичным зависимостям рассчитываются остальные коэффициенты уравнения плоскости A и B .

На рис. 3.4 показан простой пример вычисления коэффициента уравнения плоскости по известной проекции многоугольника.

Описанный подход обладает рядом несомненных преимуществ. Самое главное, что он не предъявляет особых требований к выбору точек. Они могут иметь произвольное размещение в пространстве, даже быть компланарными или коллинеарными.

Полученное уравнение плоскости позволяет точно оценить степень некомпланарности многоугольника. Для этого достаточно вычислить кратчайшее расстояние между несущей плоскостью и каждой вершиной полигона:

$$d_i = \frac{Ax + By + Cz + D}{\sqrt{A^2 + B^2 + C^2}},$$

где x, y, z — координаты вершины.

Если векторное произведение равно нулю, то выбранные три точки коллинеарны, т. е. не могут определить положение плоскости. Вместо этой выборки в принципе может быть использована любая другая совокупность точек. Свободный член уравнения D может быть определен после нахождения нормали $[A, B, C]$ и подстановки в искомое уравнение любой из трех точек.

Если для задания полигона используется более трех точек, то они могут оказаться некомпланарны. Это может быть вызвано ошибками вычисления или выбранным способом генерации многоугольника. В этом случае следует использовать другую технику для нахождения плоскости, которая наилучшим образом приближает заданное множество точек многоугольника.

Расстояние, рассчитанное по этой формуле, может быть положительным или отрицательным в зависимости от положения точки. Если точка принадлежит несущей плоскости, то $d = 0$. Если по условиям задачи требуется определить только положение точки, для этого не требуется вычислять корень и выполнять деление. Знак числителя даст точный ответ на поставленный вопрос.

Уравнение несущей плоскости не является единственным. Его можно умножить на любое число k , отличное от нуля. Это меняет вид уравнения, но не меняет положения плоскости. Часто коэффициенты плоскости хранят в нормализованном виде при

$$k = \frac{1}{\sqrt{A^2 + B^2 + C^2}}.$$

Данное решение значительно упрощает вычисление расстояний d , поскольку знаменатель соответствующего выражения становится равным единице.

3.2. Параметрические кубические кривые

Ломаные и полигоны представляют собой важнейшие способы аппроксимации кривых и поверхностей. Чтобы получить высокую точность кусочно-линейной аппроксимации сложных кривых или поверхностей в общем случае требуется большое количество фрагментов. Это влечет необходимость хранения и обработки больших массивов граничных точек, что способно усложнить некоторые интерактивные операции с геометрическими формами. Рассмотрим более эффективный способ представления кусочно-гладких кривых. Далее это представление будет обобщенно на случай пространственных поверхностей. Основная идея состоит в том, чтобы для описания геометрии сложных объектов использовать кривые более высоких порядков, чем линейные функции. В общем случае эти кривые точнее приближают форму и требуют для обработки меньше вычислительных ресурсов.

3.2.1. Основные положения

Существуют три основных подхода к использованию аппроксимационных кривых высоких порядков. Так, можно попытаться найти точные аналитические выражения вида $y = f(x)$ и $z = g(x)$. Этот выбор имеет несколько очевидных недостатков. Во-первых, нельзя однозначно описать замкнутые кривые, например окружности или эллипсы. Такие объекты приходится предварительно разбивать на ряд сегментов и искать описание каждого из них. Во-вторых, полученное описание не обладает инвариантностью относительно поворотов. Чтобы задать повернутую версию кривой следует проделать значительную вычислительную работу, а в общем случае требуется получить новое разбиение кривой на сегмен-

ты. Наконец, большие вычислительные сложности возникают при попытке задать кривые с очень большими углами наклона.

Проблемы иного порядка влечет за собой попытка задать кривую в виде решения неявного уравнения $f(x, y, z) = 0$. Во-первых, выбранное уравнение способно продуцировать несколько решений, в числе которых могут содержаться и паразитные. Во-вторых, существуют кривые, которые не имеют точного аналитического описания или описываются системами уравнений. Простейший пример такого рода — полуокружность. В-третьих, серьезные проблемы возникают в процессе объединения неявно заданных фрагментов кривых. Часто в точках их сочленения невозможно определить точную величину тангенса угла наклона, значение которого используется в процессе решения многих геометрических задач.

Параметрическое задание кривой в виде $x = x(t)$, $y = y(t)$, $z = z(t)$ преодолевает недостатки функционального и неявного способов описания кривых. Вместо кусочно-линейных кривых, рассмотренных в предыдущем разделе, здесь будут использоваться кусочно-полиномиальные кривые. Более точно, если сложная кривая разбивается на несколько сегментов, каждый из которых описывается тремя функциями $x(t)$, $y(t)$, $z(t)$, являющимися кубическими полиномами от параметра t . В современной практике геометрических вычислений кубические полиномы используются чаще всего. Полиномы низших порядков не обеспечивают достаточной точности аппроксимации кривых, а многочлены более высоких порядков порождают осцилляции формы, что приводит к росту вычислительных ресурсов. Никакие другие полиномы невысокой степени не дают возможности выполнить точную интерполяцию криволинейных фрагментов по значениям конечных точек и их производных. Кроме того, параметрические кубические кривые — это кривые самого низкого порядка, которые могут занимать произвольное положение (не лежать на плоскости) в трехмерном пространстве. Действительно, положение кривой, описываемой полиномом второй степени, задается тремя точками, а три точки всегда определяют плоскость.

Для задания кривых более высоких порядков требуется еще больше параметров. В некоторых случаях кривые демонстрируют колебания небольшой амплитуды, расположенные вдоль основной трассы кривой. Несмотря на эти недостатки, такие кривые используются в процессе моделирования поверхностей со сложным поведением, например в автомобилестроении и авиационной промышленности.

Кубический полином, описывающий криволинейный сегмент, можно представить в следующем виде:

$$Q(t) = [x(t), y(t), z(t)],$$

где

$$\begin{aligned} x(t) &= a_x t^3 + b_x t^2 + c_x t + d_x; \\ y(t) &= a_y t^3 + b_y t^2 + c_y t + d_y; \\ z(t) &= a_z t^3 + b_z t^2 + c_z t + d_z; \\ 0 &\leq t \leq 1. \end{aligned} \tag{3.1}$$

Без потери общности можно утверждать, что значения параметра t ограничиваются единичным интервалом.

Введем вектор-строку $T = [t^3, t^2, t, 1]$. Используя это обозначение, исходное выражение можно записать в более компактном матричном виде:

$$Q(t) = [x(t), y(t), z(t)] = TxC,$$

где

$$C = \begin{bmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \\ d_x & d_y & d_z \end{bmatrix} K.$$

Производная от функции $Q(t)$ называется касательным или тангенциальным вектором параметрической кривой. Тангенциальный вектор легко найти прямым дифференцированием выражения $Q(t)$:

$$\begin{aligned} \frac{d}{dt}Q(t) &= \left[\frac{d}{dt}x(t), \frac{d}{dt}y(t), \frac{d}{dt}z(t) \right] = \frac{d}{dt}T \times C = [3t^2, 2t, 1, 0] \times C = \\ &= [3a_x t^2 + 2b_x t + c_x, 3a_y t^2 + 2b_y t + c_y, 3a_z t^2 + 2b_z t + c_z]. \end{aligned}$$

На рис. 3.5 показан пример двух соединенных сегментов параметрической кубической кривой и их полиномы. Пример демонстрирует возможность параметрическим способом задавать многозначные кривые, т. е. такие зависимости, которые могут принимать несколько значений при одном значении аргумента.

Если два сегмента V_1 и V_2 параметрической кривой соединены, то говорят, что кривая принадлежит в данной точке к классу геометрической непрерывности G^0 . Если совпадают направления касательных векторов (необязательно зна-

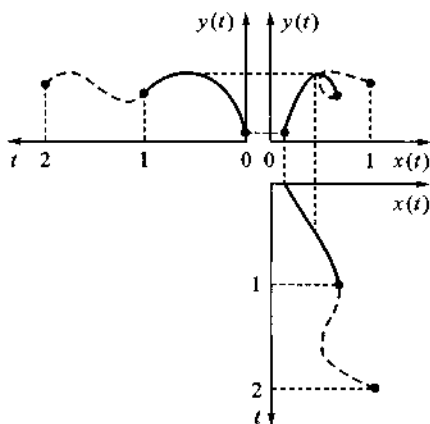


Рис. 3.5. Координаты параметрической кривой и представляющие их полиномы



Рис. 3.6. Соединение фрагментов кривых трех различных типов

C^1 . Наконец, равенство производных n порядка дает основания отнести кривую к классу C^n . Иногда непрерывность типа C_i , $i = 1, 2, \dots$ называют параметрической непрерывностью.

На рис. 3.6 показаны примеры соединений фрагментов кривых. Кусок кривой S соединяется с тремя кривыми C_0 , C_1 , C_2 (нижний индекс обозначает тип соединения).

Очевиден физический смысл касательного вектора $\frac{d}{dt}Q(t)$. Его значение

равно скорости изменения кривой относительно параметра t в данной точке. Вторая производная от $Q(t)$ по параметру t равна ускорению. Если, к примеру, камера движется вдоль параметрической кубической кривой и делает снимки через равные промежутки времени, то касательный вектор задает скорость перемещения камеры вдоль кривой. В точках сочленения фрагментов скорость движения камеры и ее ускорение не должны резко изменяться. В противном случае возможны заметные нарушения плавности съемки. На примере, показанном на рис. 3.6, кривая C_2 обеспечивает намного более высокую плавность и на значительном удалении от точки соединения.

Очевидно, что принадлежность кривой классу C^1 влечет за собой ее принадлежность G^1 . Обратное утверждение не является справедливым в общем случае. Следует отметить, что, опираясь только на визуальную оценку параметрических кривых, часто бывает трудно обнаружить значительную разницу между кривыми классов C^1 и G^1 .

Существует важный частный случай, когда выполнимость условия принадлежности к классу C^1 не влечет за собой принадлежность кривой классу G^1 . Такая ситуация может возникнуть, когда значения касательного вектора равны нулю в точке сочленения для обоих смежных сегментов параметрической кривой. В этом случае значения касательного вектора будут равны, но его направления могут различаться (рис. 3.7). Если представить себе движение камеры вдоль этой кривой, то в точке сочленения скорость ее перемещения будет равна нулю, после перехода на смежный фрагмент она изменит направление движения.

чения), то кривая в точке соединения принадлежит классу G^1 . Длины касательных векторов таких сегментов связаны прямо пропорциональным соотношением $T(V_x) = kT(V_y)$, где k — коэффициент пропорциональности. В системах автоматизированного проектирования во многих операциях используются кривые последнего класса, поскольку они воспринимаются наблюдателем как гладкие.

Если в точке сочленения сегментов совпадают значения и направления касательных векторов, то такая кривая относится к классу

3.2. Параметрические кубические кривые

Графики параметрических кривых заметно отличаются от графиков обычных функций, где независимый аргумент откладывается по одной оси, а зависимый — по другой. Аргумент t параметрических кривых не представлен на их графиках совсем. Это значит, в частности, что невозможно определить касательный вектор непосредственно по графику параметрической кривой.

Проиллюстрируем это утверждение простым примером. Пусть имеется параметрическая кривая $8(0, 0 < t < 1$, а ее касательный вектор равен нулю при $t = 0$. Предположим, что $\psi(0) = 8(0; 0 < t < 1/2$. В этом случае графики параметрических функций $\psi(0)$ и $8(0)$ совпадают. С другой стороны, $\psi'(0) = 28'(0)$. Это значит, две кривые имеют одинаковые графики, но различные касательные векторы. Поэтому принято накладывать на кривые требование принадлежности к классу G^1 . Для гладкого сочленения двух фрагментов часто достаточно совпадения направления касательных векторов, а не их полного равенства.

Как следует из определения, для нахождения кубического полинома (формула (3.1)) требуется установить значения его четырех коэффициентов. Для нахождения неизвестных следует наложить на кривую четыре дополнительных условия. Такими условиями служат значения кубического сегмента на конечных точках, значения касательного вектора и условия связности между соседними сегментами.

Рассмотрим три важнейших типа кривых — многочлены Эрмита, кривые Безье и сплайны различного рода. Первые задаются собственными значениями и величинами касательного вектора, принимаемыми на конечных точках. Для определения кривой Безье требуется задать значения кривой на конечных точках и два промежуточных значения, которые используются для пересчета касательного вектора. Для определения сплайнов должны быть известны значения четырех контрольных точек и, кроме того, могут накладываться различные дополнительные условия, например степень гладкости или поведение кривой вблизи точки сочленения. Важнейшими типами сплайнов, нашедшими широкое применение в машинной графике, являются однородные (uniform) и неоднородные (nonuniform) B-сплайны.

Напомним, что параметрическая кубическая кривая описывается уравнением

$$Q(t) = TxC,$$

где $\langle 2(0 = MO, j(0, z(t)), T = [t^3, t^2, t, 1]$; C — прямоугольная матрица коэффициентов

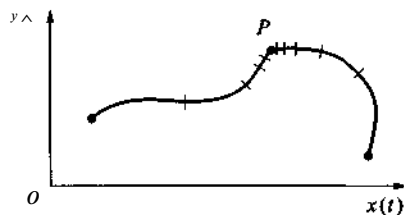


Рис. 3.7. Пример кривой, которая принадлежит классу C^1 , но не входит в класс G^1 .

P — точка сочленения фрагментов (засечками обозначены расстояния, пройденные объектом за равные промежутки)

$$C = \begin{bmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \\ d_x & d_y & d_z \end{bmatrix}.$$

Представим матрицу C в виде $C = MG$, где матрица M — базовая матрица четвертого порядка, а G — вектор-столбец геометрических ограничений, который часто называют геометрическим вектором. Координаты геометрического вектора формируются на основе дополнительной информации о поведении кривой. Это могут быть значения на концевых точках, величины касательного вектора и другие ограничения, определяющие поведение кривой. Обозначим через G_x вектор-столбец, содержащий только x -компоненты геометрического вектора. Аналогичное толкование будут иметь обозначения G_y и G_z . По матрицам M и G строится классификация различных типов кривых.

Запишем выражение параметрической кривой в развернутом виде:

$$Q(t) = [x(t) \ y(t) \ z(t)] = [t^3 \ t^2 \ t \ 1] \times \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ m_{41} & m_{42} & m_{43} & m_{44} \end{bmatrix} \times \begin{bmatrix} G_1 \\ G_2 \\ G_3 \\ G_4 \end{bmatrix}. \quad (3.2)$$

Найдем выражение для координаты $x(t)$

$$\begin{aligned} x(t) &= T_x M x G_x = \\ &= (t^3 m_{11} + t^2 m_{21} + t m_{31} + m_{41}) G_{x1} + (t^3 m_{12} + t^2 m_{22} + t m_{32} + m_{42}) G_{x2} + \\ &+ (t^3 m_{13} + t^2 m_{23} + t m_{33} + m_{43}) G_{x3} + (t^3 m_{14} + t^2 m_{24} + t m_{34} + m_{44}) G_{x4}. \end{aligned} \quad (3.3)$$

Из уравнения (3.3) следует, что кривая представляет собой взвешенную сумму элементов геометрической матрицы. Такой вес имеют кубические полиномы по всем трем геометрическим координатам. Совокупность весовых коэффициентов иногда называют стыковочной функцией, или функцией сопряжения (blending function). Стыковочные функции B задаются простым матричным соотношением

$$B = T_x M.$$

Способы вычисления матрицы M зависят от специфики параметрических кривых.

3.2.2. Кривые Эрмита

Свое название кривые получили от имени французского математика Шарля Эрмита, который подробно исследовал их свойства. Это частный случай кубиче-

ских полиномиальных кривых, которые задаются на значениями конечных точек и величинами касательного вектора. Обозначим P , P_A — значения кривой на конечных точках сегмента, а R , R_S — значения касательного вектора в начале и конце сегмента. Удобнее использовать такую нумерацию вместо последовательной, так как далее номерами 2 и 3 обозначаются внутренние точки сегментов кривых.

Для определения базисной матрицы M_h кривой Эрмита, которая связана с геометрическим вектором G , полиномиальных коэффициентов, запишем четыре уравнения (по одному на каждое геометрическое условие) и, решив полученную систему, найдем значения неизвестных величин.

Запишем компоненту x геометрической матрицы Эрмита в следующем виде:

$$G_{h_x} = \begin{bmatrix} P_{4x} \\ P_{4x} \\ R_{1x} \\ R_{4x} \end{bmatrix}$$

и из уравнения (3.2) найдем $x(t)$:

$$x(t) = a_x t^3 + b_x t^2 + c_x t + d_x = T \times C_x = T \times M_h \times G_{h_x} = [t^3, t^2, t, 1] M_h G_{h_x} \quad (3.4)$$

Ограничения в начальной и конечной точках интервала могут быть записаны в виде

$$\begin{aligned} x(0) &= P_{1x} = [0, 0, 0, 1] M_h G_{h_x}; \\ x(1) &= P_{4x} = [1, 1, 1, 0] M_h G_{h_x}. \end{aligned}$$

Найдем производную от уравнения (3.4):

$$\frac{dx}{dt} = [3t^2, 2t, 1, 0] M_h G_{h_x}.$$

С помощью этого уравнения запишем значения, которые принимает касательный вектор на границах интервала:

$$\begin{aligned} \frac{dx}{dt}(0) &= R_{1x} = [0, 0, 1, 0] M_h G_{h_x}; \\ \frac{dx}{dt}(1) &= R_{4x} = [3, 2, 1, 0] M_h G_{h_x}. \end{aligned}$$

Четыре уравнения $\left(x(0), x(1), \frac{dx(0)}{dt}, \frac{dx(1)}{dt} \right)$, использующие граничные

условия, можно записать в более компактном матричном виде:

$$\begin{bmatrix} P_x \\ P_{4x} \\ R_{1x} \\ R_{4x} \end{bmatrix} = G_{hx} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 3 & 2 & 1 & 0 \end{bmatrix} M_h \times G_{hx}. \quad (3.5)$$

Из этого уравнения легко найти значение матрицы M_h :

$$M_h = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix}^{-1} = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}. \quad (3.6)$$

Полученное значение можно использовать для определения величины $x(t)$. Для этого достаточно воспользоваться соотношением

$$x(t) = Tx M_h x G_{hx},$$

которое позволяет вычислить искомую величину по известным значениям геометрического вектора G_{hx} . Следуя приведенной схеме, можно найти выражения для координат $y(t)$ и $z(t)$:

$$y(t) = Tx M_h x G_{hy};$$

$$z(t) = Tx M_h x G_{hz}.$$

Используя три последних уравнения, можно записать компактное матричное выражение для уравнения кривой:

$$Q(t) = [x(t), y(t), z(t)] = Tx M_h x G_h; \quad (3.7)$$

$$G_h = \begin{bmatrix} P_1 \\ P_4 \\ R_1 \\ R_4 \end{bmatrix}.$$

Напомним, что P_1, P_4 — значения кривой в концевых точках; R_1, R_4 — значения касательного вектора на границах интервала; G_h — геометрический вектор. Если в уравнении (3.7) раскрыть подвыражение $Tx M_h$, то можно определить стыковочную функцию Эрмита B_h в явном виде:

$$\begin{aligned} Q(t) &= Tx M_h x G_h = B_h x G_h = \\ &= (2t^3 - 3t^2 + 3t)P_1 + (-2t^3 + 3t^2)P_4 + (t^3 - 2t^2 + t)R_1 + (t^3 - t^2)R_4. \end{aligned} \quad (3.8)$$

На рис. 3.8 показаны графики четырех стыковочных функций, используемых в кривых Эрмита в качестве весов элементов геометрического вектора. На ри-

сунке они помечены именами начальных условий, вклады которых регулируют. Так, функция $f(t) = 2t^3 - 3t^2 + 1$, которая задает удельный вес P_1 , фигурирует под этим именем.

В начальной точке $t = 0$) только одна кривая P_1 отлична от нуля. В начальной точке на форму кривой $Q(t)$ оказывает влияние только одна кривая P_1 . С ростом параметра t влияние функций R_1, R_4, P_4 становится все более заметным.

Введем обозначения:

$$P_1(t) = 2t^3 - 3t^2 + 1;$$

$$P_4(t) = -2t^3 + 3t^2;$$

$$R_1(t) = t^3 - 2t^2 + t;$$

$$R_4(t) = t^3 - t^2.$$

На рис. 3.9 представлены стыковочные функции, взвешенные значениями геометрического вектора (рис. 3.9, а), их сумма (рис. 3.9, б) и результирующая кривая $Q(t)$ (рис. 3.9, в).

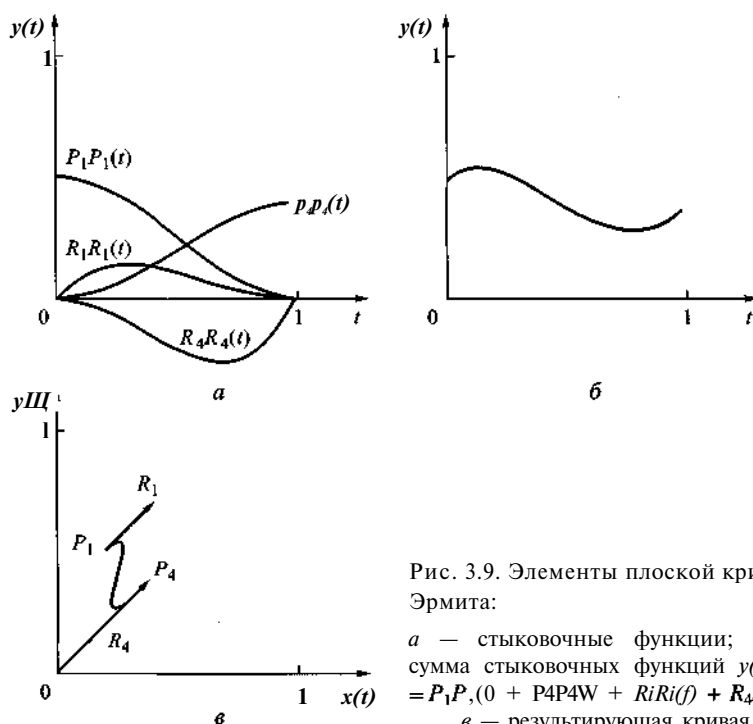


Рис. 3.9. Элементы плоской кривой Эрмита:

а — стыковочные функции; б — сумма стыковочных функций $y(t) = P_1P_1(t) + P_4P_4(t) + R_1R_1(t) + R_4R_4(t)$; в — результирующая кривая

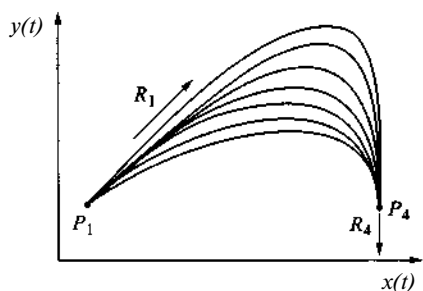


Рис. 3.10. Семейство кривых Эрмита

Семейство кривых Эрмита изображено на рис. 3.10. Различная форма кривых продуцируется вариациями длины касательного вектора R . Направление этого вектора и остальные величины, влияющие на форму кривых, остаются без изменения.

На рисунке видно, что чем длиннее касательный вектор, тем большее влияние он оказывает на форму кривой.

На рис. 3.11 показано еще одно семейство кривых Эрмита, полученное в результате изменения направления касательного вектора одной длины. Если провести дополнительные эксперименты с кривыми Эрмита, то можно заметить еще одну закономерность. Чем меньше длина касательного вектора, тем более спрямленную форму имеет кривая.

Многие интерактивные графические системы позволяют управлять формой кривых Эрмита интерактивно, посредством настройки положения конечных точек и величин касательных векторов (рис. 3.12). Эти объекты выводятся на экране дисплея и выполняют функции регуляторов формы. Оператор может при помощи мышки перемещать конечные точки и буксировать окончания касательных, программа отслеживает сделанные изменения, рассчитывает и рисует новую форму кривой. Гладкость точки сочленения фрагментов (точка Л на рис. 3.12) обычно задается специальной командой, по умолчанию окончания касательных и точка сочленения являются коллинеарными.

Если две кубические кривые Эрмита соединяются в концевой точке класса G^1 (рис. 3.13, точка P), то их геометрические векторы должны иметь вид

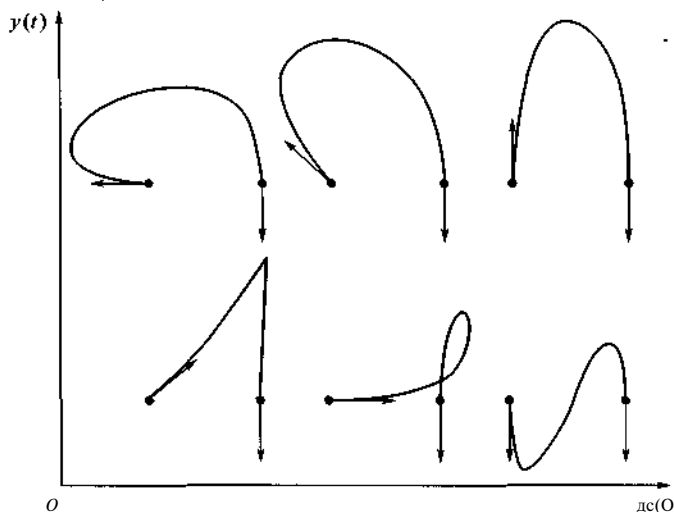


Рис. 3.11. Семейство кривых Эрмита



Рис. 3.12. Интерактивное управление формой кривой Эрмита

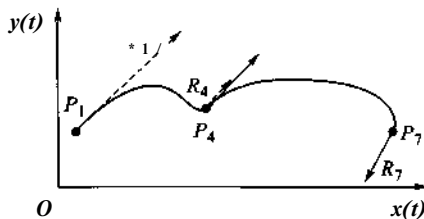


Рис. 3.13. Пример соединения двух фрагментов кривых Эрмита класса G^1

$$\begin{bmatrix} P_1 \\ PA \\ \mathbf{A} \end{bmatrix} \quad \text{и} \quad \begin{bmatrix} PA \\ PI \\ kR_4 \\ R_7 \end{bmatrix} \quad \text{при } k > 0,$$

т. е. они обязаны иметь общую точку (P_4) и касательные векторы, по крайней мере, с равными направлениями. Более строгие ограничения накладывает принадлежность к классу $C \setminus B$ в этом случае $k=1$, поэтому в точке сочленения должны совпадать направления и значения касательных векторов смежных сегментов.

С точки зрения визуализации кривые Эрмита очень просты в обработке. Для этого достаточно воспользоваться основным уравнением (3.8). Интервал изменения параметра t делится на несколько частей, после чего вычисляется последовательность значений полинома непосредственно по формуле определения. Для выполнения искомых расчетов существует очень компактный программный код. Эффективность вычислений можно повысить за счет использования схемы Горнера разложения полиномов:

$$f(t) = at^3 + bt^2 + ct + d = ((at + b)t + c)t + d.$$

Поскольку кубическая кривая представляет собой линейную комбинацию элементов геометрического вектора (3.3), то можно использовать рациональную стратегию преобразования кривых. Вместо обработки кривой как последовательности сегментов целесообразно выполнить искомое преобразование над элементами геометрического вектора, а затем построить трансформированную кривую. Этот подход оказывается работоспособным для операций поворота, масштабирования и переноса.

3.2.3. Кривые Безье

Рассмотренные в предыдущем разделе методы представления кривых дают хорошие результаты. Они очень удобны при описании формы, основа которой получена с помощью экспериментов или математических расчетов. Это может быть фюзеляж или крыло самолета, некоторые фрагменты двигателя внутреннего сгорания, многие стандартные механические детали и т. д. Существует, одна-

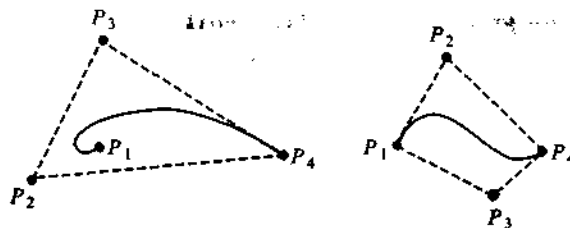


Рис. 3.14. Примеры кривых Безье и положение их контрольных точек

ко, и другой класс задач, когда решение зависит как от функциональных, так и от эстетических требований. Многочисленные примеры таких поверхностей можно найти в любых отраслях дизайна и художественного проектирования, например корпус автомобиля, формы посуды, мебели и т. п. Кроме количественных критериев здесь требуется учет предпочтений дизайнера, и часто необходимо интерактивное вмешательство проектировщика.

Кривые Эрмита не очень удобны для интерактивной работы с кривыми. Направление и величина касательных не дают необходимого интуитивного представления о кривой, так как неочевидна связь между набором чисел и формой соответствующей кривой. Французский исследователь Пьер Безье (P. Bezier) предложил другой метод создания кривых и поверхностей любой формы.

Кривые Безье — специальный вид кубических полиномиальных кривых, у которых для определения положения касательных векторов используются специальные контрольные точки, не принадлежащие самому объекту. На рис. 3.14 пунктиром обозначены выпуклые оболочки контрольных точек. В общем случае выпуклая оболочка может и не касаться всех контрольных точек.

Начальный R_1 и конечный R_4 касательные векторы кривой Безье зависят от векторов P_1, P_2 и P_3, P_4 , где все P_i — радиусы-векторы, соединяющие контрольные точки с началом координат. Эти величины задаются следующими соотношениями:

$$\begin{aligned} R_1 &= \frac{dQ(0)}{dt} = 3(P_2 - P_1); \\ R_4 &= \frac{dQ(1)}{dt} = 3(P_4 - P_3). \end{aligned} \quad (3.9)$$

Геометрический вектор кривой Безье имеет вид

$$G_b = \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{bmatrix}.$$

Обозначим через M_{bb} матрицу, которая задает соотношение между геометрическим вектором Эрмита и геометрическим вектором Безье. Она определя-

ется соотношением $G_h = M_{hb} \times G_b$, которое в развернутом виде имеет следующий вид:

$$G_h = \begin{bmatrix} P_1 \\ P_4 \\ R_1 \\ R_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -3 & 3 & 0 & 0 \\ 0 & 0 & -3 & 3 \end{bmatrix} \begin{bmatrix} P_1 \\ P_4 \\ P_2 \\ P_4 \end{bmatrix} = M_{hb} \times G_b. \quad (3.10)$$

Для нахождения базисной матрицы M_b воспользуемся уравнением для кривых Эрмита

$$Q(t) = [x(t), y(t), z(t)] = Tx M_h x G_h,$$

в котором заменим G_h на $M_{hb} G_b$ и представим M_b в виде $M_b M_{hb}$, тогда

$$Q(t) = Tx M_h x G_h = Tx M_h x (M_{hb} x G_b) = Tx (M_h x M_{hb}) x G_b = Tx M_b x G_b.$$

Выполнив матричное умножение, получим

$$M_b = M_h \times M_{hb} = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}. \quad (3.11)$$

Окончательно имеем

$$Q(t) = (1-t)^3 P_1 + 3t(1-t)^2 P_2 + 3t^2(1-t) P_3 + t^3 P_4. \quad (3.12)$$

Четыре полинома, которыми взвешены координаты геометрического вектора в уравнении (3.12), называются полиномами Бернштейна (базисом Бернштейна) (рис. 3.15). По внешнему виду этих полиномов легко оценить влияние контрольных точек на результирующую форму кривой Безье. При $t = 0$ только B_1 отличен от нуля, поэтому в этой точке кривая приближается к P_1 . Аналогично при $t = 1$ только полином B_4 вносит свой заметный вклад в поведение кривой, поэтому на форму кривой влияет только положение точки P_4 .

На рис. 3.16 показаны два сегмента кривой Безье, соединенные общей точкой. Для ее принадлежности классу G^1 требуется выполнение условия

$$P_3 - P_4 = \kappa(P_4 - P_2), \quad \kappa > 0.$$

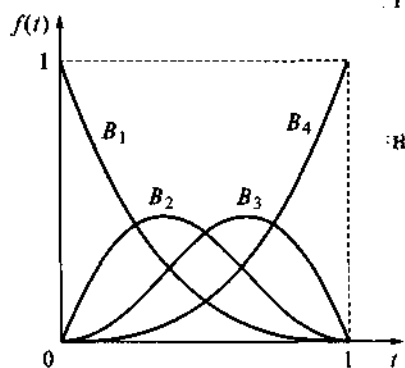


Рис. 3.15. Представление полиномов Бернштейна, которые служат весовыми коэффициентами в математическом описании кривых Безье:

$$B_1 = (1-t)^3; \quad B_2 = 3t(1-t)^2; \\ B_3 = 3t^2(1-t); \quad B_4 = t^3.$$

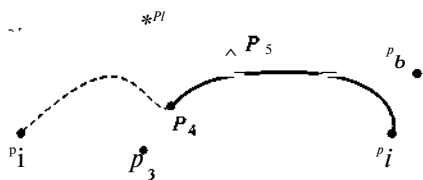


Рис. 3.16'. Два сегмента кривой Безье, соединенные общей точкой P_4

Это значит, что три точки P_b , P_d , P_5 должны отличаться друг от друга, но лежать на одной прямой. Если коэффициент $\kappa = 1$, то дополнительно к непрерывности класса G^1 гарантируется принадлежность кривой классу C^1 .

Обозначим полиномы двух смежных сегментов x_l (левого) и x_r (правого). Используя введенные обозначения, можно сформулировать условия, которые обеспечивают выполнимость условий C^0 и C^1 в точке сочленения сегментов:

$$x_l(1) = x_r(0); \quad \frac{dx_l(1)}{dt} = \frac{dx_r(0)}{dt}.$$

Применив уравнение (3.12), получим

$$\begin{aligned} x_l(1) &= x_r(0) = P_{4x}; \\ \frac{dx_l(1)}{dt} &= 3(P_{4x} - P_{3x}); \\ \frac{dx_r(0)}{dt} &= 3(P_{5x} - P_{4x}). \end{aligned}$$

Как и предполагалось, эти условия выполняются и для двух оставшихся координат y и z . Для экономии места соответствующие выражения опускаются. Таким образом, для выполнимости условий C^0 и C^1 требуется, чтобы $P_4 - P_3 = P_5 - P_4$.

Если проанализировать полиномы Бернштейна ($y| - B_j$), то очевидно, что их сумма всегда равна единице, а каждый из них принимает только неотрицательные значения на интервале $0 < t < 1$. Таким образом, $Q(t)$ представляет собой взвешенное среднее четырех контрольных точек. Это значит, что каждый сегмент кривой Безье полностью вписан в выпуклую оболочку, форма и положение которой зависит от контрольных точек (см. рис. 3.14). Для двумерных кривых эту оболочку можно представить как многоугольник, который образует резиновая нить, натянутая на контрольные точки. Для кривых в трехмерном пространстве оболочкой служит полиэдр, образованный подобно многоугольнику.

Свойство выпуклой оболочки сохраняется для всех кубических кривых (не только кривых Безье), определенных как взвешенная сумма контрольных точек, если их веса неотрицательны и в сумме дают единицу. Более того, это утверждение оказывается справедливым и для n -мерного пространства.

Еще один очевидный вывод следует из равенства суммы полиномов единице. Значение четвертого полинома всегда могут быть найдены по известным величинам трех других простым вычитанием из единицы их суммы.

Выпуклые оболочки сегментов можно использовать для эффективного решения задачи отсечения изображения. Вместо того, чтобы определять принадлежность области видимости каждого сегмента кривой отдельно, достаточно рассмотреть их выпуклые оболочки. Если выпуклая оболочка не принадлежит области видимости, то ее сегмент автоматически исключается из рассмотрения. И наоборот, принадлежность оболочки фрагмента кривой влечет за собой включение соответствующего сегмента. Если оболочка пересекает границу области, то для решения поставленной задачи требуется рассмотреть положения самого сегмента кривой.

В заключение раздела приведем краткую сводку основных свойств кривых Безье:

- форма кривой Безье зависит от выпуклой оболочки, образованной контрольными точками кривой;
- первая и последняя точки кривой совпадают с соответствующими точками определяющего многоугольника;
- векторы касательных на концах кривой Безье по направлению совпадают с первой и последней сторонами многоугольника;
- кривая обладает свойством уменьшения вариации. Это означает, что кривая пересекает любую прямую линию не чаще, чем определяющий многоугольник;
- кривая инвариантна относительно аффинных преобразований.

С математической точки зрения кривая, заданная вершинами многоугольника, зависит от интерполяции или аппроксимации, устанавливающей связь кривой и многоугольника. Здесь основой является выбор базисных функций. Базис Бернштейна, который порождает кривые Безье, обладает двумя свойствами, ограничивающими гибкость кривых. Ограничение состоит в том, что количество вершин многоугольника жестко задает порядок многочлена. Например, кубическая кривая должна быть задана четырьмя вершинами и тремя отрезками. Многоугольник из шести точек всегда порождает кривую пятого порядка. Единственный способ понизить степень кривой — сократить число вершин, а повысить степень кривой — увеличить их число.

Второе ограничение следует из глобальной природы базиса Бернштейна. Это означает, что величина аппроксимирующих функций $B_i(t)$ из уравнения (3.12) ненулевая для всех значений параметра t в интервале от 0 до 1. Любая точка на кривой Безье зависит от всех определяющих вершин, поэтому изменение какой-либо одной вершины оказывает влияние на всю кривую. Локальные воздействия на кривую невозможны.

Поскольку наклон концов кривой Безье задан соответствующими сторонами многоугольника, можно передвинуть среднюю вершину пятиточечного многоугольника, не меняя направления на концах. Вследствие глобальности базиса Бернштейна меняется форма всей кривой. Невозможность внесения локальных изменений не является решающим критерием оценки кривых Безье. Это ограничение преодолевается алгоритмическими и программными средствами. Все пакеты, оперирующие с такими кривыми, разрешают пользователю создать новую

вершину в любой внутренней точке сегмента кривой. Это влечет за собой разбиение сегмента на две части и предоставляет дополнительные возможности локального управления формой кривой.

3.2.4. Однородные нерациональные В-сплайны

Термин «сплайн» заимствован из традиционной технологии изготовления поверхностей сложного профиля. Так ранее называли полосы материала, которые под действием грузов выгибались таким образом, чтобы получить искомую конфигурацию сложной фигурной поверхности. Математическим эквивалентом физических сплайнов служат так называемые натуральные кубические сплайны, представляющие собой непрерывные кубические полиномы, форма которых задается n контрольными точками. Кривые этого вида обладают более высокой степенью гладкости, чем кривые Эрмита и Безье. При выполнении некоторых дополнительных условий они могут принадлежать к классам C^0 , C^1 и C^2 . Однако их применение в системах машинной графики сдерживается двумя недостатками. Вычисление формы натуральных полиномов требует обращения матриц высокого порядка, что затрудняет интерактивную работу с ними даже на мощных вычислительных системах. Кроме того, смещение контрольной точки влечет за собой не локальные изменения формы, а влияет на общую форму кривой.

•• В-сплайны — совокупность полиномиальных сегментов, положение которых задается небольшим числом контрольных точек. Смещение одной контрольной точки влияет только на некоторый фрагмент кривой и не вносит глобальных изменений в ее форму. В-сплайны обладают такой же гладкостью в точках сочленения, как натуральные сплайны, но значительно превосходят последние по эффективности вычислений.

При обсуждении В-сплайнов немного изменим систему обозначений, принятую в предыдущих разделах. Будем рассматривать эти объекты как интегральные кривые, а не как отдельные сегменты. Сегменты В-сплайнов не обязаны проходить через контрольные точки. Условия гладкости могут передаваться на сегмент со стороны соседних фрагментов.

Кубический В-сплайн задается последовательностью полиномиальных сегментов $2z$, $Q^{\wedge--}, Qm$, положение которых зависит от контрольных точек P_o, P_u, P_m , $m > 3$. Сегменты, образующие кривую, могут быть определены как функции аргумента t на интервале $0 < t < 1$, однако удобнее переопределить их таким образом, чтобы они принимали значения на общем последовательном интервале (для этого достаточно выполнить подстановку $t = t + \kappa$).

Итак, каждый сегмент Q_i задан на интервале $t_i < t < t_i + \frac{1}{3}$, где $3 \leq i < m$. Первый сегмент ($2z$ ($w = 3$)) задан на интервале $\frac{1}{3} \leq t < u$ четырьмя контрольными точками P_0, \dots, P_6 . Для каждого $i > 4$ между соседними сегментами Q_{i-1} и Q_i существует точка сочленения или узел, положение которого задается значением па-

параметра t . Это значение будем называть узловым. Начальный узел расположен в точке Q_3 , конечный — в t_{m+1} , общее число узлов на кривой равно $m+1$.

На рис. 3.17 показан пример двухмерного сплайна с выделенными узлами и контрольными точками.

Название «однородный сплайн» означает, что все узлы расположены на равном расстоянии по значениям параметра t . Без потери общности можно предположить $t_0 = 0, t_m - t, \dots = 1$. В подразд. 3.2.5 рассматриваются сплайны, у которых расстояния между узлами могут быть не равны. Такие объекты называются неоднородными В-сплайнами. Термин «нерациональные» используется для того, чтобы

отличить нерациональные объекты от так называемых рациональных сплайнов. Последние представляют собой параметрические кривые, у которых координаты $x(t)$, $y(t)$ и $z(t)$ представляют собой отношение двух кубических полиномов. Рациональные сплайны рассматриваются в следующем разделе. Префикс «В» означает, что кривые этого вида представляют собой взвешенную сумму полиномиальных базисных функций. Он вводит явное различие В-сплайнов от натуральных, которые этим свойством не обладают.

Каждый из $m - 2$ криволинейных сегментов В-сплайна задается четырьмя точками из $m + 1$ контрольных. В частности, сегмент Q_2 определяется контрольными точками P_{i-2}, P_{i-1} и P_i . Таким образом, геометрический вектор В-сплайна $G_{B_{si}}$ для сегмента Q_i имеет следующий вид:

$$G_{B_{si}} = \begin{bmatrix} P_{i-3} \\ P_{i-2} \\ P_{i-1} \\ P_i \end{bmatrix}, \quad 3 \leq i \leq m.$$

Первый сегмент Q_3 кривой задается контрольными точками P_0, P_1, P_2 при значениях параметра t в диапазоне от $t_3 = 0$ до $t_4 = 1$, положение сегмента Q_4 определяется точками P_1, P_2, P_3 и значениями параметра в диапазоне от $t_4 = 1$ до $t_5 = 2$. Наконец, последний сегмент Q_m задается посредством контрольных точек $P_{m-3}, P_{m-2}, P_{m-1}, P_m$ в диапазоне от $t_m = m - 3$ до $t_{m+1} = m - 2$. В общем случае сегмент Q_i берет свое начало в районе контрольной точки P_{i-2} , а заканчивается в окрестности контрольной точки P_i . Можно показать, что стыковочные функции сегментов В-сплайна всюду неотрицательны, а их сумма равна единице. Это значит, что геометрия сегментов определяется выпуклой оболочкой, которая задается четырьмя контрольными точками.

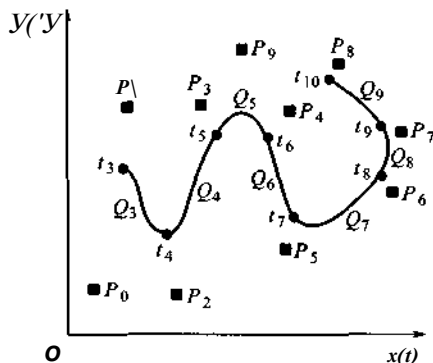


Рис. 3.17. Пример В-сплайна, образованного сегментами $Q^i - Q_{g'}$
• — узлы; • — контрольные точки

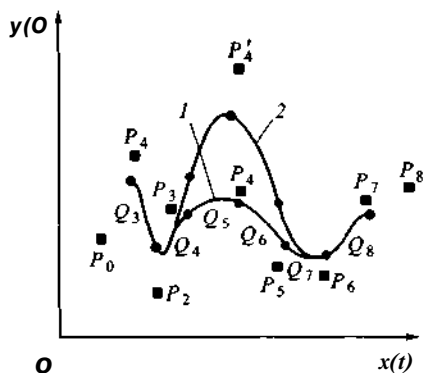


Рис. 3.18. Пример В-сплайна с разными положениями одной контрольной точки (P_4):

• — узлы; • — контрольные точки; 1 — исходная кривая; 2 — кривая после

сдвига

Итак, каждый криволинейный фрагмент задается четырьмя контрольными точками. В свою очередь, каждая контрольная точка, исключая первую и последнюю в последовательности P_0, P_1, \dots, P_m , оказывает влияние на четыре сегмента кривой. Если передвинуть контрольную точку в определенном направлении, то в ту же сторону будут смещены зависящие от нее сегменты кривой. Остальные фрагменты не изменят своего положения и формы.

На рис. 3.18 показан пример В-сплайна в двух положениях. Верхняя кривая получена смещением одной контрольной точки P_4 , что повлияло только на четыре подчиненных сегмента кривой, остальные ее части не претерпели никаких изменений. Это свойство ограниченности изменений характерно для всех типов В-сплайнов, а также и другим ви-

дам кривых, обсуждаемых далее.

Обозначим через Γ вектор-строку:

$$[(t-t_f), (t-t_f), (t-b), 1].$$

Тогда формулу В-сплайна для сегмента i можно записать в следующем виде:

$$Q_i(t) = T_{\chi} M_{B_s} \times G_{B_{ij}}, \quad *, . < * \ll t_{i+1} \quad (3.13)$$

Уравнение всей кривой можно получить, применив уравнение (3.13) для $3 < i \leq m$.

Пропуская промежуточные вычисления, приведем выражение базисной матрицы M_{B_s}

В-сплайна:

$$M_{B_s} = \frac{1}{6} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix}.$$

Стыковочные функции B_{B_s} записываются (по аналогии с кривыми Безье и Эрмита) в виде $T_{\chi} M_{B_s}$. Важно, что стыковочные функции всех криволинейных сегментов В-сплайна

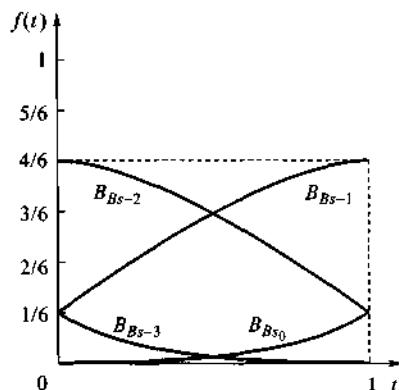


Рис. 3.19. Стыковочные функции В-сплайна

равны друг другу. Для каждого i -го сегмента значение $t - ?$; лежит в пределах от 0 (при $t = t_j$) до 1 (при $t = t_{j+1}$). Если заменить $t - ?$, на t , а интервал $[t_j, t_{j+1}]$ на $[0, 1]$, то получим

$$\begin{aligned} B_{es} &= TM_{Bs} = B_{Bs-2}, B_{Bs-1}, B_{Bs}, B_{Bs+1}, B_{Bs+2} \\ &= \frac{1}{6} [-t^3 + 3t^2 - 3t + 1, \quad 3t^3 - 6t^2 + 4, \quad -3t^3 + 3t^2 + 3t + 1, \quad t^3] = \quad (3.14) \\ &= \frac{1}{6} [(1-t)^3, \quad 3t^3 - 6t^2 + 4, \quad -3t^3 + 3t^2 + 3t + 1, \quad t^3]; \quad 0 < t < 1. \end{aligned}$$

На рис. 3.19 показаны все четыре стыковочные функции В-сплайна. Отметим особенности этих кривых: на концах единичного интервала три из четырех функций отличны от нуля; все функции неотрицательны, а их сумма равна единице. При этом каждый сегмент кривой принадлежит некоторой выпуклой оболочке.

Развернем уравнение (3.13), заменив на втором шаге преобразований $t-t_i$, на t .

$$\begin{aligned} Q_i(t-t_i) &= T_i \times M_{B_i} \times G_{B_{si}} = T \times M_{B_i} \times G_{B_{si}} = \\ &= B_{B_i} \times G_{B_{si}} = B_{B_{i-2}} \times P_{i-2} + B_{B_{i-1}} \times P_{i-1} + B_{B_i} \times P_i = \\ &= (1-Q_{i-2}^3, \quad 3Q_{i-2}^3 - 6Q_{i-2}^2 + 4Q_{i-2}, \quad -3Q_{i-2}^3 + 3Q_{i-2}^2 + 3Q_{i-2} + 1, \quad Q_{i-2}) \quad (3.15) \\ &\quad \text{до} \quad \text{до} \end{aligned}$$

Покажем, что сегменты Q_i и Q_{i+1} в точке своего сочленения принадлежат классам C^0 , C^1 , C^2 . Рассмотрим координату x , которую с левой и правой сторон узла t_i можно записать в виде $x_i(t - t_i)$ и $x_{i+1}(t - t_i)$. Все выкладки, которые далее приведены для координаты x , можно распространить на координаты y и z .

Для доказательства утверждения достаточно подтвердить правильность соотношений

$$\begin{aligned} \frac{dx_j(t_M)}{dt} &= \frac{dx_M(t_M)}{dt}, \\ \frac{d^2x_i(t_M)}{dt^2} &= \frac{d^2x_M(t_M)}{dt^2}. \end{aligned}$$

Заменим аргумент $t - t_i$ на t и перепишем эти уравнения в следующем виде:

$$\begin{aligned} x_i(t-t_i) &= x_M(t-t_i) = 0; \\ \frac{dx_i}{dt}(t-t_i) &= \frac{dx_M}{dt}(t-t_i) = 0; \\ \frac{d^2x_i}{dt^2}(t-t_i) &= \frac{d^2x_M}{dt^2}(t-t_i) = 0. \end{aligned}$$

Вычислив приведенные выражения, получим

$$\begin{aligned} x_i \Big|_{1/2, i=1}^{x_i+1} \Big|_{1-r, -i+1=0} &= \frac{P_{i-2x} + 4P_{i-1x} + P_{ix}}{6}; \\ \frac{dx_i}{dt} \Big|_{1/2, i=1}^{dx_M} \Big|_{1-r, -i+1=0} &= \frac{-P_{i-2x} + P_{ix}}{2}; \\ \frac{d^2 x_i}{dt^2} \Big|_{2M} &= \frac{d^2 M}{dt^2} \Big|_{M, i=0} = P_{i-2x} - 2P_{i-1x} + P_{ix}. \end{aligned}$$

В-сплайны обладают высокой степенью гладкости. За это полезное во многих приложениях свойство приходится платить некоторой потерей управляемости данного сорта объектов. Формой кривой можно управлять изменяя положения контрольных точек. Отметим несколько важных частных случаев. Если положить $P_{j-2} = P_i$, то кривая пройдет ближе к данной контрольной точке. Сегмент Q_j кривой, в пределах которого локализовано данное воздействие, зависит теперь только от трех контрольных точек, а точка $P_{i-2} = P_i$ в уравнении (3.15) имеет двукратный вес и соответственно большее влияние на поведение данного фрагмента. Если некоторая контрольная точка используется 3 раза, например $P_{i-2} = P_{i-1} = P_i$ то уравнение (3.15) принимает вид

$$Q_i (0=B_{BS-3} \times x / b + (V_2 + V_1 + B_{BS0}) \times P_i,$$

а сегмент Q_i превращается в прямую линию.

На рис. 3.20 изображены три конфигурации сегментов сплайна и их контрольных точек. Рис. 3.20, а демонстрирует ситуацию, когда существуют ограничения на число свободных контрольных точек. Выпуклые оболочки двух смежных сегментов перекрываются; точка сочленения сегментов Q_3 и Q_4

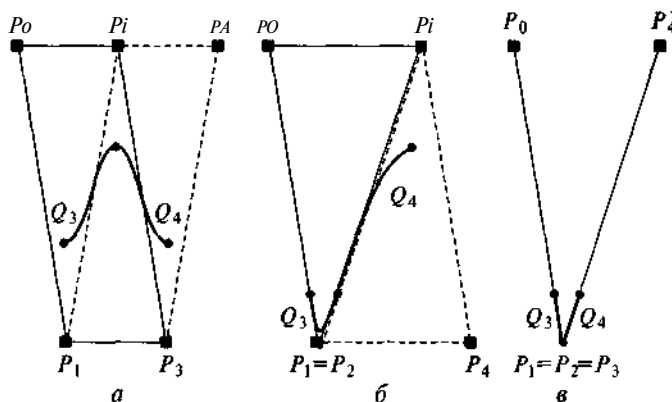


Рис. 3.20. Влияние контрольных точек на положение сегментов В-сплайна:

а — контрольные точки различные; б — две контрольные точки P_1 и P_2 совпадают; в — три контрольные точки P_1 , P_2 и P_3 совпадают; — выпуклая оболочка сегмента Q_3 , — выпуклая оболочка сегмента Q_4 .

принадлежит области пересечения оболочек. На рис. 3.20, б положение контрольных точек P_1 и P_2 совпадает. В результате выпуклые оболочки соседних фрагментов кривой получили общую границу по линии P_2P_3 а точка сочленения сегментов принадлежит этой прямой. На рис. 3.20, в показан пример с тремя совпадающими контрольными точками ($P_1 = P_2 = P_3$)- Это самый строгий тип ограничений приводит к вырождению выпуклых оболочек и накладывает предельно жесткие ограничения на размещения точки сочленения сегментов. В этом случае точка сочленения сегментов совпадает с тройной контрольной точкой.

3.2.5. Неоднородные нерациональные В-сплайны

Как отмечалось в подразд. 3.2.4, у однородных нерациональных сплайнов расстояния между соседними точками сочленения равны между собой. Иногда это жесткое ограничение преодолевается введением неоднородных нерациональных В-сплайнов. Эти сплайны могут иметь неравномерную последовательность узлов. У сегментов кривых этого вида стыковочная функция в общем случае неодинаковая.

Неоднородные нерациональные В-сплайны имеют несколько преимуществ по сравнению с равномерными сплайнами: обладают высокой чувствительностью к изменениям контрольных точек, дают большую оперативную свободу пользователю, демонстрируют хорошую управляемость в начальной и конечной точках и имеют значительно более высокую устойчивость по отношению к вырожденным ситуациям. В любой момент работы с неоднородным В-сплайном в состав кривой можно ввести дополнительный узел и изменить ее форму при помощи дополнительных контрольных точек. В однородных В-сплайнах этого сделать нельзя.

Чтобы рассмотреть неоднородные В-сплайны в общем случае, требуется немного изменить принятые ранее соглашения об обозначениях. Как и ранее, сплайн представляет собой непрерывную последовательность сегментов, которые продуцируются кубическими полиномами. Положение кривой задается контрольными точками P_0-P_m . Узлы кривой образуют неубывающую последовательность t_0-t_m+4 , число узлов превосходит число контрольных точек ровно на четыре. Минимальное число контрольных точек равно четырем, поэтому самая короткая последовательность узлов равна восьми.

По определению неоднородных нерациональных В-сплайнов последовательность узлов должна быть неубывающей. В частности, расстояния между соседними узлами могут быть равны. В подобных ситуациях можно говорить о кратности узлов и параметров. Пусть, например, последовательность значений параметра для узлов некоторой кривой имеет вид (0, 0, 0, 1, 1, 2, 3, 4, 4, 5, 5, 5, 5). Здесь значение 0 имеет кратность четыре, 1 — кратность два, значения 2 и 3 характеризуются единичной кратностью и т. д.

Сегмент кривой Q_i задается контрольными точками $P_{i-3}, P_{i-2}, P_{i-1}, P_i$ и стыковочными функциями $\#_{i-3,4}(t), \#_{i-2,4}(t), \#_{i-1,4}(t), \#_{i,4}(t)$ в виде взвешенной суммы

$$Q_i(t) = P_{i-3} \times B_{i-3,4}(t) + P_{i-2} \times B_{i-2,4}(t) + P_{i-1} \times B_{i-1,4}(t) + P_i \times B_{i,4}(t);$$

$$3 \leq i \leq m, \quad t_i \leq t \leq t_{i+1}.$$

Кривая не определена за пределами интервала $[t_i, t_{i+1}]$. Если $t_i = t_{i+1}$, то сегмент Q_i вырождается в точку, что дает дополнительные возможности по обработке неоднородных сплайнов.

В отличие от других типов сплайнов для неоднородных нерациональных В-сплайнов не существует единственного набора стыковочных функций. Они зависят от интервалов между узлами и определяются рекурсивно. Обозначим через $B_{ij}(t)$ стыковочную функцию j -го порядка контрольной точки P_i . Поскольку рассматриваются кубические сплайны, то рекурсивное определение будет ограничено функциями вида $B_{i,j}(t)$. Рекурсивные соотношения, задающие стыковочные функции, имеют следующий вид:

$$B_{i,1}(t) = \begin{cases} 1, & t_i \leq t < t_{i+1}, \\ 0, & \text{в противном случае;} \end{cases}$$

$$B_{i,2}(t) = \frac{t - t_i}{t_{i+2} - t_i} B_{i,1}(t) + \frac{t_{i+3} - t}{t_{i+3} - t_{i+1}} B_{i+1,1}(t);$$

$$B_{i,3}(t) = \frac{t - t_i}{t_{i+3} - t_i} B_{i,2}(t) + \frac{t_{i+4} - t}{t_{i+4} - t_{i+1}} B_{i+1,2}(t);$$

$$B_{i,4}(t) = \frac{t - t_i}{t_{i+4} - t_i} B_{i,3}(t) + \frac{t_{i+5} - t}{t_{i+5} - t_{i+1}} B_{i+1,3}(t).$$

На рис. 3.21 показана схема вычисления стыковочных функций (3.16). Рисунок демонстрирует, почему для вычисления четырех стыковочных функций требуется вектор длиной восемь. На интервале $0 < t < 1$ функция $B_{i,1}(t)$ равна единице, все остальные функции принимают нулевое значение. Очевидно, что $B_{i,1}(t)$ и $B_{i+1,1}(t)$ представляют собой линейные функции и служат для задания линейной интерполяции между двумя контрольными точками. Функции $B_{i,2}(t)$, $B_{i+1,2}(t)$ являются квадратичными и служат для определения квадратичной интерполяции.

Вычисление стыковочных функций требует больших вычислительных ресурсов. Поэтому в некоторых приложениях ограничиваются вектором, который содержит только нулевые и единичные интервалы, что ограничивает трудоемкость матричных операций по формулам (3.16).

Можно показать, что все стыковочные функции принимают неотрицательные значения, а их сумма равна единице. Это значит, что все сегменты неоднородного нерационального В-сплайна лежат внутри некоторой выпуклой оболочки, опирающейся на четыре контрольные точки.

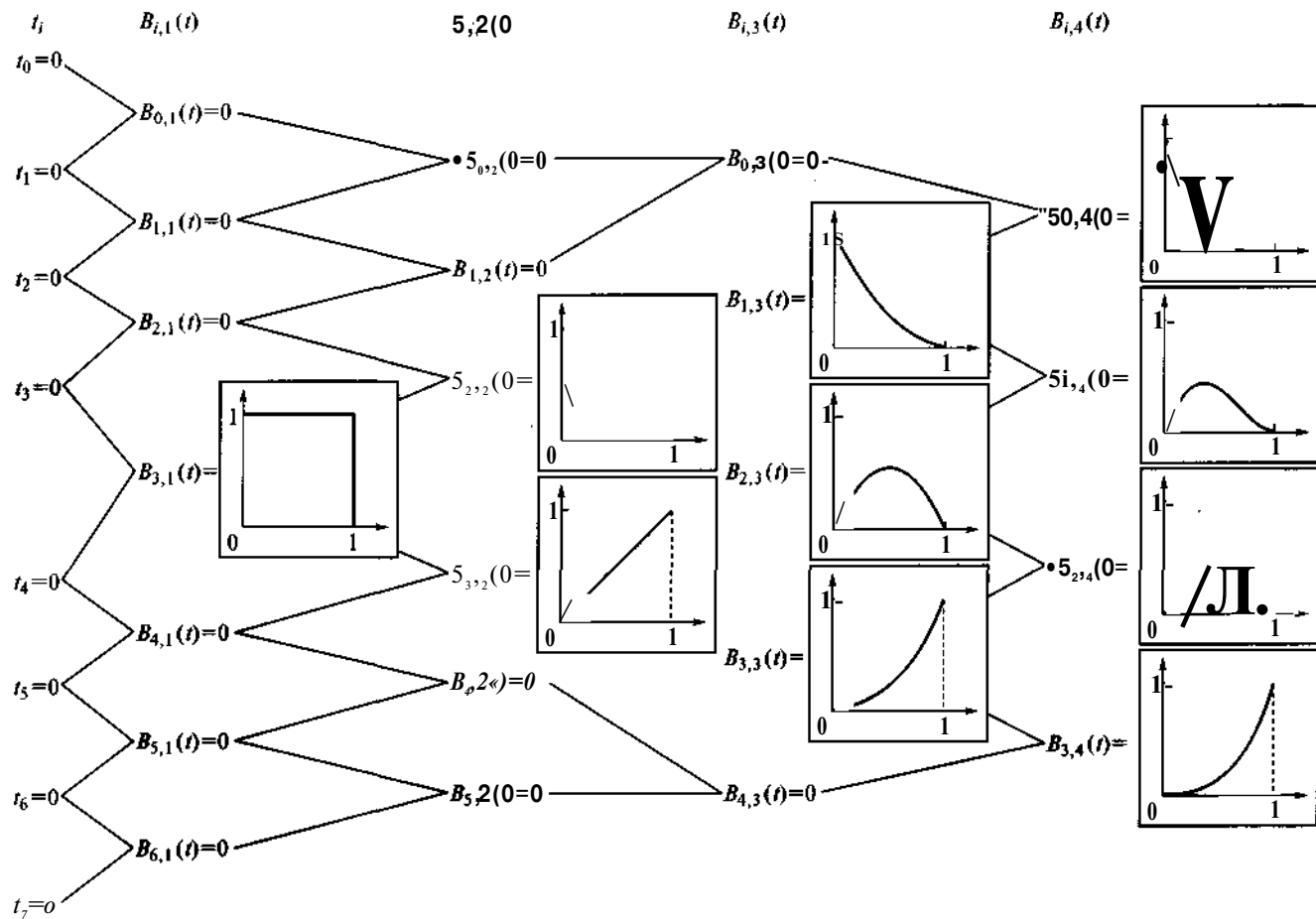


Рис. 3.21. Схема вычисления стыковочных функций: распределение узлов кривой задается вектором $(0, 0, 0, 0, 1, 1, 1, 1)$

Увеличение кратности узлов влечет за собой два главных следствия. Во-первых, каждое значение аргумента t , принадлежит выпуклой оболочке точек P_{i-3} , P_{i-2} и P_{i-1} . Если величины t_i и t_{i+1} равны, то они, с одной стороны, должны принадлежать выпуклой оболочке P_{i-3} , P_{i-2} и P_{i-1} и, с другой стороны, оболочке P_{i-2} , P_{i-1} и P_i . Это значит, что они будут лежать на прямой, соединяющей точки P_{i-2} и P_i . Если $t_i - t_{i+1} = ti+2$, то такой узел обязан совпадать с контрольной точкой P_{i+2} . Условие $t_i = t_{i+1} = t_{i+2} = t_{i+3}$ требует размещения узла сразу на двух контрольных точках P_{i+1} и P_{i+2} . Это значит, что кривая в данном узле претерпевает разрыв. Кроме того, кратные узлы снижают максимальный уровень гладкости кривой.

На рис. 3.22 показаны два примера кривых, поясняющих зависимость формы от кратности узлов. Все узлы верхней кривой, изображенной на рис. 3.22, *a*, имеют кратность, равную единице. Последовательность узлов можно описать вектором $(0, 1, 2, 3, 4, 5)$. В местах сочленения сегментов кривая принадлежит классам C^1 и C^2 . Сегменты этой кривой задаются четырьмя контрольными точками и зависят от четырех стыковочных функций. Каждый фрагмент принадле-

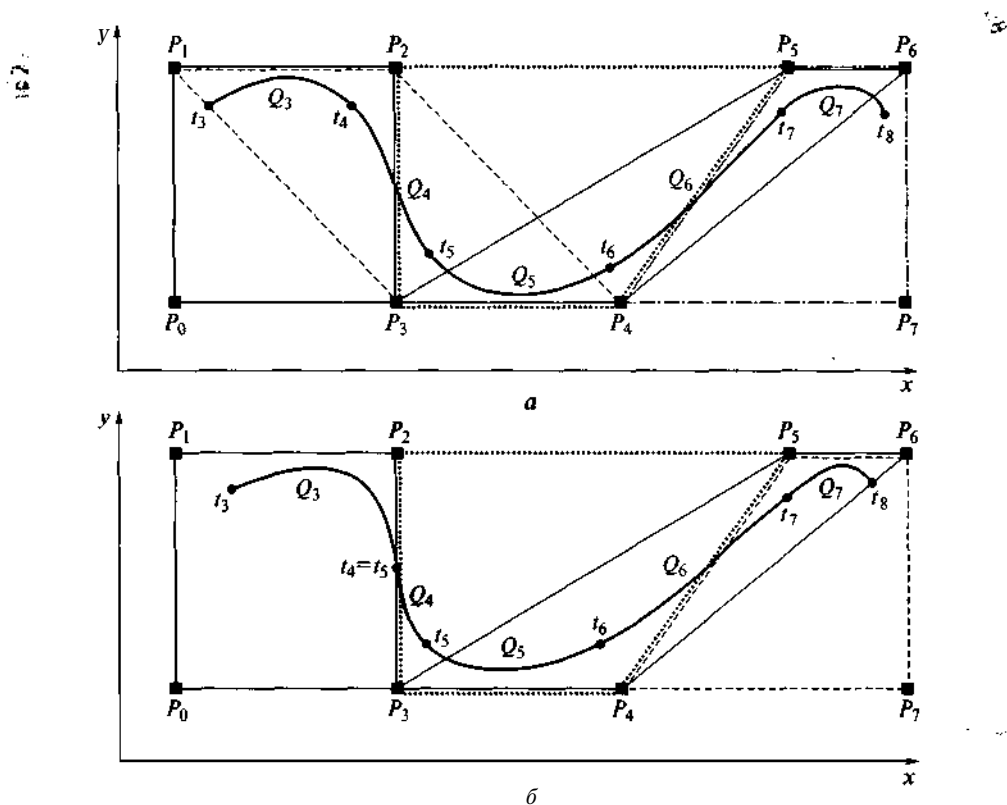


Рис. 3.22. Влияние кратных узлов на поведение кривых:

a — без кратных узлов; *б* — с одним двойным узлом; —, ----, ———— границы выпуклых оболочек

жит выпуклой оболочке. Смежные сегменты имеют по три общие контрольные точки. Например, сегмент Q_3 зависит от точек P_0, P_1, P_2, P_3 , а следующий на нем сегмент Q_4 задается точками P_1, P_2, P_3 и P_4 .

Кривая, представленная на рис. 3.22, б, имеет один двойной узел $f_4 = ts$, поэтому сегмент Q_4 равен нулю. Последовательность узлов можно записать в виде вектора $(0, 1, 1, 2, 3, 4)$. В результате сегменты Q_3 и Q_5 становятся смежными. Общей частью их выпуклых оболочек является прямая P_2P_3 . Точка сочленения сегментов принадлежит этой прямой. В точке сочленения кривая обладает гладкостью по классам C^1 и G^2 .

Еще два примера кривых приведены на рис. 3.23. Последовательность узлов, изображенная на рис. 3.23, а, задается вектором $(0, 1, 1, 1, 2, 3)$, что означает наличие узла кратности три. В результате сегменты Q_4 и Q_5 вырождаются в точку. Выпуклые оболочки сегментов Q_3 и Q_6 пересекаются в одной точке P_3 . В соответствии с этим условием кривая должна проходить через данную контрольную точку.

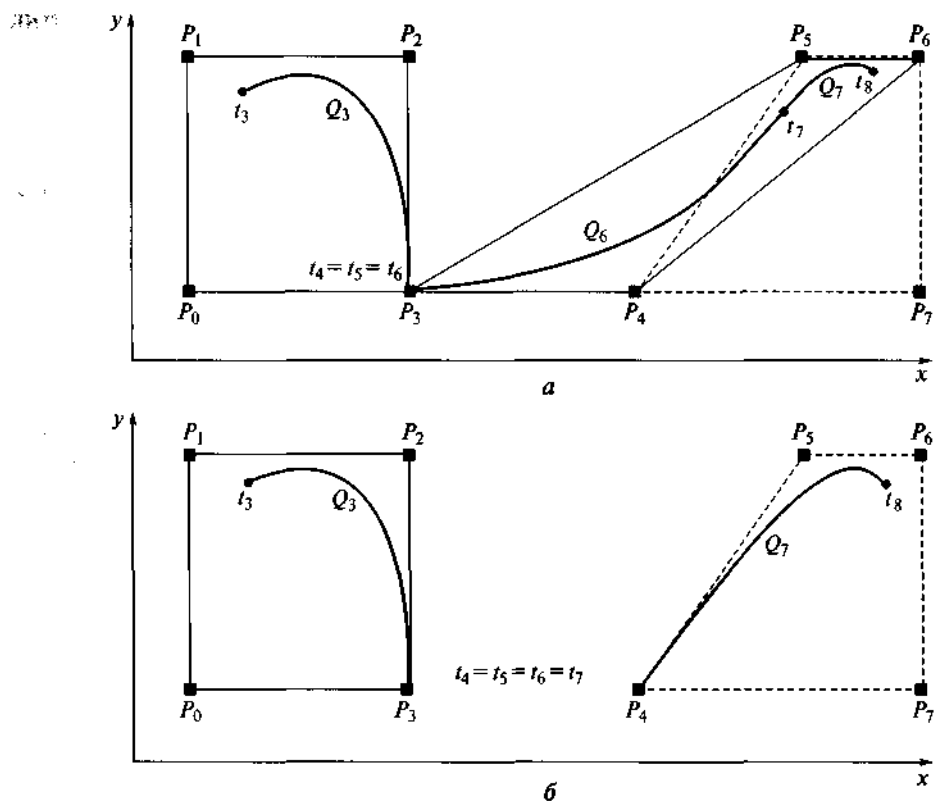


Рис. 3.23. Влияние узлов высокой кратности на поведение кривых:
а — с одним узлом кратности три; б — кривая с узлом кратности четыре;
—, — границы выпуклых оболочек

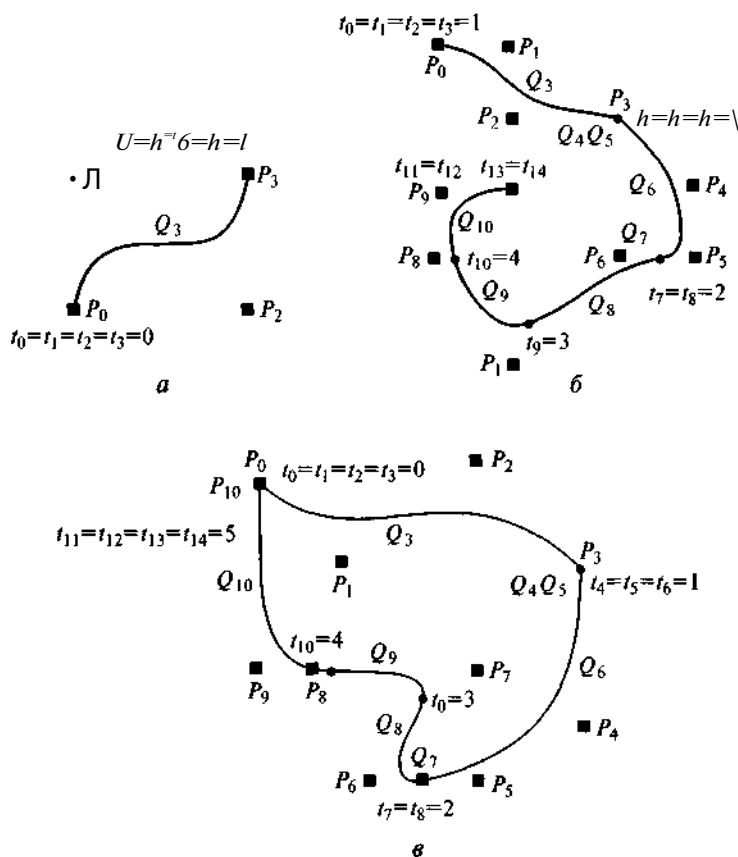


Рис. 3.24. Примеры вырожденных сплайнов:

a — форма сплайна задается конечными точками; *б* — открытый сплайн с узловым вектором $(0, 0, 0, 0, 1, 1, 1, 2, 2, 3, 4, 5, 5, 5, 5)$; *в* — замкнутый сплайн, задаваемый узловым вектором $(0, 0, 0, 0, 1, 1, 2, 2, 3, 4, 5, 5, 5, 5)$

Кривая, приведенная на рис. 3.23, б, имеет узел кратности четыре, а соответствующий вектор узлов имеет вид $(0, 1, 1, 1, 1, 2)$. Сплайн состоит из двух сегментов Q_3 и Q_4 , которые не имеют общих контрольных точек, что приводит к нарушению непрерывности.

На рис. 3.24 показаны три кривые, полученные описанным математическим аппаратом. Сплайн, представленный на рис. 3.24, а, описывается вектором узлов $(0, 0, 0, 0, 1, 1, 1, 1)$. Фактически положение этой кривой определяется конечными точками, промежуточные точки не оказывают влияния на ее форму и положение. Это вырожденная форма неоднородного нерационального В-сплайна, которая совпадает с кривой Безье. Два других В-сплайна (рис. 3.24, б, в) начинаются и оканчиваются в узлах кратности три. Они описываются одинаковыми узловыми векторами $(0, 0, 0, 0, 1, 1, 1, 2, 2, 3, 4, 5, 5, 5, 5)$, но различными кон-

трольными точками. Каждая кривая состоит из сегментов Q^i . Сегменты Q_i , Q_{i+1} и Q_n опираются на кратные узлы и имеют нулевую длину.

3.2.6. Рациональная форма кривых и сплайнов

В общем случае любой сегмент рациональной кубической кривой можно записать в следующем виде:

$$\frac{Y(t)}{W(t)} = \frac{Z(t)}{W(t)}$$

где $X(t)$, $Y(t)$, $Z(t)$ и $W(t)$ — кубические полиномиальные кривые, контрольные точки которых заданы в однородных координатах.

Кривую можно представить как четверку $Q(t)=[X(t), Y(t), Z(t), W(t)]$, существующую в некотором однородном пространстве. Чтобы задать кривую в трехмерном пространстве, достаточно каждую координату этого вектора поделить на $W(t)$. Очевидно, что любая нерациональная кривая может быть представлена в рациональном виде простым добавлением $W(t) = 1$ в качестве четвертого элемента. В частности, в таком виде можно представить кривые Безье и Эрмита и многие другие типы кривых. Неоднородные В-сплайны, приведенные к рациональной форме, в литературе по машинной графике называются NURBS-кривые (сокращение от nonuniform rational B-spline).

Рациональные кривые обладают несколькими преимуществами, самым важным из которых является то, что они инвариантны относительно некоторых преобразований контрольных точек. К числу допустимых преобразований относятся: поворот, масштабирование, параллельный перенос и перспективное отображение. Нерациональные кривые менее устойчивы в этом отношении, так как они сохраняют инвариантность только относительно поворота, масштабирования и перемещения.

Таким образом, для реализации перспективного преобразования рациональной кривой, достаточно применить его к контрольным точкам, новое положение которых позволяет вычислить преобразованную форму кривой. Для нерациональных кривых существуют два пути выполнения этой операции. Первый состоит в приведении кривой к рациональной форме. Второй отличается более высокой трудоемкостью: требуется по контрольным точкам восстановить форму кривой, а затем к ее точкам применить перспективное преобразование.

Еще одним важным преимуществом рациональных сплайнов является их способность описывать точные формы конических сечений, что имеет большое значение для систем автоматизированного проектирования, которые интенсивно используют эти формы в различных проектных задачах. Нерациональные кривые могут только аппроксимировать такие формы. Причем для точного приближения приходится использовать большое число контрольных точек, что требует значительных вычислительных ресурсов.

Вектор узлов NURBS-кривой представляет собой набор числовых параметров, которые задают позиции и силу влияния контрольных точек на форму кривой. Эти значения используются для внутренних вычислений и лишь в немногих программных приложениях выполняют функции интерфейсных элементов. Например, система трехмерного моделирования Alias Maya позволяет оператору интерактивно влиять на узлы, но этот способ формообразования оказался менее удобным, чем прямая работа с контрольными точками.

3.2.7. Разбиение кривых

Пусть создана кривая в виде последовательности сегментов, которая приближает некоторую физическую или мыслимую форму. Очень часто возникает ситуация, когда манипуляции с доступными контрольными точками не позволяют добиться требуемой точности аппроксимации или искомой формы этой кривой, поэтому необходимо увеличить управляемость кривой.

Существуют два основных подхода для решения этой задачи:

- увеличение степени полиномов, например с трех до четырех или выше. Это решение является оправданным лишь в некоторых частных случаях, когда требуется повысить гладкость кривой. Часто в результате этого выбора появляются осцилляции на кривой, вследствие чего повышается трудоемкость вычислительных операций;

- увеличение числа контрольных точек путем разбиения сегментов кривых. Пусть, например, имеется сегмент кривой Безье с четырьмя контрольными точками. Если разделить его на две части, то число контрольных точек возрастет до семи, что улучшит управляемость кривой. Для неоднородных В-сплайнов аналогичный процесс называется подразбиением (refinement) и заключается в добавлении произвольного числа новых контрольных точек. Очень часто эту операцию приходится выполнять в процессе визуализации кривых и поверхностей.

Рассмотрим постановку задачи. Дана кривая Безье $Q(t)$, положение которой определяется точками P_0, P_2, P_3 и P_4 . Требуется найти левую кривую, задаваемую точками L_0, L_1, L_2 и L_4 , которая совпадает с исходной кривой на полуинтервале $0 < t < 1/2$. По тем же исходным данным необходимо определить правую кривую, которая совпадает с заданной кривой на полуинтервале $1/2 < t < 1$ с точками P_1, R_2, R_3 и R_4 .

Обозначим через t аргумент дополнительной точки. Все необходимые обозначения и схема генерации новой точки показана на рис. 3.25. Для ее нахождения требуется построить вспомогательную линию L_2H , которая делит отрезки P_0P_2 и P_2P_3 в отношении $\frac{t}{1-t}$. Далее необходимо найти положение линии $///P_3$, которая делит отрезки P_2P_3 и P_3P_4 в той же пропорции. После этого следует про-

вести линию Lj, R_2 , разбивающую отрезки L_2H и HRT , в заданном соотношении. В результате на кривой $Q(t)$ будет получена точка $L_4 = R_1$, которая, в свою очередь, разобьет отрезок

L_1L_2 в отношении $\frac{t}{1-t}$.

Все вычисления существенно упрощаются, если принять самый простой вид пропорции, когда обрабатываемые отрезки делятся пополам. Приведем формулы для расчета положения точек:

$$\begin{aligned} L_2 &= \frac{P_1 + P_2}{2}; \quad H = \frac{P_2 + P_3}{2}; \quad L_3 = \frac{L_2 + H}{2}; \\ R_3 &= \frac{P_3 + P_4}{2}; \quad R_2 = \frac{H + R_3}{2}; \quad L_4 = R_1 = \frac{L_3 + R_2}{2}. \end{aligned}$$

Обозначим D_B^L и D_B^R — левую и правую части кривой Безье в матричном представлении. С помощью этих матриц можно записать выражение, задающее левую G_B^L и правую G_B^R части геометрического вектора разделенной кривой Безье. Они описываются следующими соотношениями:

$$\begin{aligned} G_B^L &= D_B^L G_B = \frac{1}{8} \begin{bmatrix} 8 & 0 & 0 & 0 \\ 4 & 4 & 0 & 0 \\ 2 & 4 & 2 & 0 \\ 1 & 3 & 3 & 1 \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ J \end{bmatrix}; \\ G_B^R &= D_B^R G_B = \frac{1}{8} \begin{bmatrix} 1 & 3 & 3 & 1 \\ 0 & 2 & 4 & 2 \\ 0 & 0 & 4 & 4 \\ 0 & 0 & 0 & 8 \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ J \end{bmatrix}. \end{aligned}$$

Каждая новая контрольная точка L_i (R_i) кривой Безье представляет собой взвешенную сумму P_i , причем каждая координата принимает только неотрицательные значения и их сумма равна единице. Таким образом, части кривой Безье лежат внутри выпуклой оболочки, опирающейся на контрольные точки кривой. Это значит, что новые контрольные точки не могут находиться далеко от кривой. В действительности они лежат ближе к ней, чем ее оригинальные контрольные точки. Это свойство сокращения осцилляции справедливо для любого сплайна, обладающего выпуклой оболочкой. Из очевидной симметрии геометрических построений следует симметрия матриц D_B^L и D_B^R .

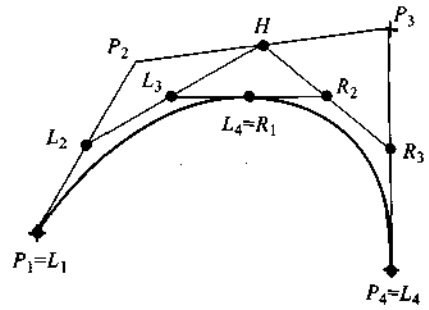


Рис. 3.25. Схема разбиения кривой Безье при $t = 1/2$

Разделение кривой Безье при $t = 1/2$ существенно упрощает саму процедуру и все сопутствующие вычисления. Иногда требуется решить эту задачу в общем виде, т. е. разделить кривую в произвольной точке, положение которой задается пользователем. Процедура разбиения в основных своих операциях совпадает с описанной за исключением пропорции деления сторон, которая в общем случае

$$\text{равна } \frac{t}{1-t}.$$

Соответствующие матрицы $D_{B_s}^L$ и $D_{B_s}^R$, задающие разделение В-сплайна, в общем случае имеют вид

$$G_{B_{si}}^L = D_{B_s}^L \times G_{B_{si}} = \frac{1}{8} \begin{bmatrix} 4 & 4 & 0 & 0 \\ 1 & 6 & 1 & 0 \\ 0 & 4 & 4 & 0 \\ 0 & 1 & 6 & 1 \end{bmatrix} \begin{bmatrix} \sim P_{i-1} \\ P_{i-2} \\ P_{i-1} \\ A \end{bmatrix};$$

$$G_{B_{si}}^R = D_{B_s}^R \times G_{B_{si}} = \frac{1}{8} \begin{bmatrix} 1 & 6 & 1 & 0 \\ 0 & 4 & 4 & 0 \\ 0 & 1 & 6 & 1 \\ 0 & 0 & 4 & 4 \end{bmatrix} \begin{bmatrix} \sim P_{i-3} \\ P_{i-2} \\ P_{i-1} \\ A \end{bmatrix}.$$

Если внимательно посмотреть на эти матрицы, то очевидно, что четыре контрольные точки $G_{B_{si}}$ и новые контрольные точки находятся в конфликте. Изменения старых или новых точек приводят к нарушению связности кривой. Проблема решается использованием неоднородных В-сплайнов.

Разделение неоднородного В-сплайна — непростая задача, которая не описывается матричными зависимостями, заданными в явном виде. Эта процедура может быть задана рекурсивно. Ее основной целью является введение нового узла в последовательность узлов сплайна.

Пусть неоднородный В-сплайн задан последовательностью контрольных точек P_0, P_1, \dots, P_n . Будем считать, что известно значение параметра $t = t'$, задающего положение нового узла. Требуется найти новую последовательность контрольных точек Q_0, Q_1, \dots, Q_n , задающую положение уточненной кривой. Без ограничения общности можно считать, что $t_i < t < t_{i+1}$. Новые контрольные точки кривой можно найти по следующим рекуррентным зависимостям:

$$\begin{aligned} Q_0 &= P_0; \\ Q_i &= (1 - a_i)P_{i-1} + a_i P_i, \quad 1 \leq i \leq n; \\ Q_{n+1} &= P_n, \end{aligned} \tag{3.17}$$

где a_i задается соотношениями

$$\begin{aligned} a_{i-1} &= 1, \quad K / < J - 3; \\ a_j &= \frac{t_j - t_m}{t_{i+3} - t_i}, \quad j - 2 < i \wedge j; \\ a_{i+1} &= 0, \quad i + 1 < i < n. \end{aligned} \quad (3.18)$$

Рассмотрим работу описанного способа на примере кривой, заданной последовательностью узлов $(0, 0, 0, 0, 1, 1, 1, 1)$. Координаты x вектора контрольных точек записываются в виде $(5, 8, 9, 6)$, положение новой точки задается параметром $t = 0,5$ ($n = 3, j = 3$). Непосредственные вычисления по формулам (3.18) позволяют найти значения коэффициентов a_j ; их удобно записать в виде вектора $(0,5; 0,5; 0,5)$. Применяя уравнения (3.17), найдем координаты x новых контрольных точек Q_j . Они равны $(5,0; 6,5; 8,5; 7,5; 6,0)$.

Добавление нового узла влечет за собой замену двух старых контрольных точек сплайна на три новые. После операции подразбиения сплайна сегменты, которые примыкают к разделяемому сегменту, определяются в терминах новых контрольных точек. Более громоздкие последствия влечет за собой процедура разбиения равномерных сплайнов. Во-первых, четыре старые контрольные точки заменяются пятью новыми. Во-вторых, сегменты, примыкающие к обрабатываемому сегменту, как и ранее, задаются старыми контрольными точками, а новые сегменты зависят от собственного набора точек.

3.2.8. Преобразование представлений

Решение многих прикладных задач требует преобразования формы представления кривых. Иными словами, задана кривая, представленная геометрическим вектором G и базисной матрицей M . Требуется найти геометрический вектор G_2 и базисную матрицу M_2 , не меняющие кривую, а значит, удовлетворяющие соотношению $G \setminus M_x = G_2 M_2$. Решим это простое матричное уравнение относительно G_2 :

$$\begin{aligned} M_2^{-1} \times M_2 \times G_2 &= M_2^{-1} \times M_1 \times G_1; \\ G_2 &= M_2^{-1} \times M_1 \times G_1 = M_{12} \times G_1, \end{aligned}$$

Матрица $M_{12} = M_2^{-1} \times M_1$ выполняет преобразование известного геометрического вектора G в неизвестный вектор G_2 . Преобразование В-сплайнов в кривые Безье выполняется при помощи матрицы

$$M_{12} = M_2^{-1} \times M_1 = \frac{1}{6} \begin{bmatrix} 1 & 4 & 1 & 0 \\ 0 & 4 & 2 & 0 \\ 0 & 2 & 4 & 0 \\ 0 & 1 & 4 & 1 \end{bmatrix},$$

а обратное преобразование задается матрицей

$$M_{B_s}^{-1} = M_{B_s}^{-1} M_{B_s} = \begin{bmatrix} 6 & -7 & 2 & 0 \\ 0 & 2 & -1 & 0 \\ 0 & -1 & 2 & 0 \\ 0 & 2 & -7 & 6 \end{bmatrix}.$$

Неоднородные В-сплайны не описываются матричными соотношениями в явной форме, но существует косвенный путь преобразования такого рода объектов. Напомним, что В-сплайн с последовательностью узлов $(0, 0, 0, 0, 1, 1, 1, 1)$ представляет собой кривую Безье. Чтобы перевести неоднородный сплайн в иную форму, достаточно предварительно превратить его в кривую Безье, добавив необходимое количество дополнительных узлов. Трансформация кривых Безье в другие формы выполняется посредством известных матричных преобразований.

3.2.9. Рисование кривых

Существуют два основных способа рисования параметрических кривых:

- пошаговая визуализация, когда значения координат $x(t)$, $y(t)$ и $z(t)$ вычисляются при последовательно возрастающих значениях параметра t ;
- процедура рекурсивного подразделения, которая завершается при достаточном приближении контрольных точек к кривой.

Одним из простых и распространенных решений задачи визуализации является использование схемы Горнера разложения многочлена. Для этой схемы в каждой точке в трехмерном пространстве приходится выполнять девять операций умножения и десять операций сложения, что для кривых сложной формы влечет за собой значительные вычислительные издержки.

Рассмотрим более эффективный способ определения кубических полиномов, основанный на вычислениях правых разностей. Правая разность функции $f(t)$ задается соотношением

$$\Delta f(t) = f(t + \delta) - f(t), \quad \delta > 0.$$

Перепишем это уравнение в виде $f(t + \delta) = f(t) + \Delta f(t)$, а затем в более экономной итеративной форме

$$f_{n+1} = f_n + \Delta f_n, \quad (3.19)$$

где функция / вычисляется в точках, равноотстоящих друг от друга на одинаковый интервал δ . Поэтому $t_n = n\delta$ и $f_n = f(t_n)$.

Для полинома третьей степени $f(t) = at^3 + bt^2 + ct + d = TxC$ правая разность запишется в следующем виде:

$$\begin{aligned} Af(t) &= a(t+5)^3 + b(t+b)^2 + c(t+b) + d - \{at^3 + bt^2 + ct + d\} = \\ &= 3at^2\delta + t(3a\delta^2 + 2bd) + ad^3 + b\delta^2 + cb. \end{aligned} \quad (3.20)$$

Из уравнения (3.20) следует, что правая разность функции ДО представляет собой полином второй степени. Это не очень оптимистический вывод, поскольку данная формула является частью уравнения (3.19), которое требует еще и операции сложения. Попробуем упростить вычисления. Из уравнения (3.19) можно получить

$$\Delta^2 f(t) = \Delta(\Delta f(t)) = \Delta f(t+\delta) - \Delta f(t). \quad (3.21)$$

Используя выражения (3.20) и (3.21), после приведения подобных имеем

$$\Delta^2 f(t) = 6a\delta^2 t + 6a\delta^3 + 2b\delta^2,$$

т. е. уравнение первой степени относительно параметра t . Запишем уравнение (3.21) в индексной форме:

$$A^2 f_n = A f_{n+1} - A f_n.$$

Разрешим его относительно $\Delta/_n$ и заменим n на $n-1$:

$$\Delta f_n = \Delta f_{n-1} + \Delta^2 \Pi, \quad (3.22)$$

Теперь, чтобы по уравнению (3.19) вычислить слагаемое $\Delta/_n$, подсчитаем $\Delta^2/_n$, и добавим результат к $\Delta/\^$. Поскольку $\Delta^2/_n$ является линейной функцией относительно t , предложенная схема намного менее трудоемка по сравнению с непосредственным вычислением $\Delta/_n$ по уравнению (3.20).

Попробуем еще упростить процедуру вычислений, для этого найдем

$$\Delta^3/(O) = A(A^2/(O)) = \Delta^2/(? + 6) - \Delta^2/(O) = 6a\delta^3. \quad (3.23)$$

Эта правая разность представляет собой постоянную величину. Перепишем уравнение (3.23) в индексной форме и подставим в это выражение найденное ранее значение для $\Delta^3 f_n$:

$$\Delta^2 f_{n+1} = \Delta^2 f_n + \Delta^3 f_n = \Delta^2 f_n + 6a\delta^3.$$

Еще раз перепишем это уравнение, заменив в нем n на $n-2$,

$$\Delta^2 f_{n-1} = \Delta^2 f_{n-2} + 6a\delta^3.$$

Полученный результат можно использовать для быстрого вычисления $\Delta/_n$ по формуле (3.22).

Чтобы организовать вычисления на интервале от $n = 0$ до $n8 = 1$, найдем начальные условия по формулам (3.20)–(3.23) при $t = 0$. Все необходимые расчеты выполняются прямым использованием указанных формул. Результат запишем в матричной форме:

$$D = \begin{bmatrix} /0 \\ A/0 \\ A^2/0 \\ A^3/0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 6^3 & 6^2 & 6 & 0 \\ 65^3 & 26^3 & 0 & 0 \\ 65^3 & 0 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}. \quad (3.24)$$

Перепишем это выражение в более компактной форме, обозначив квадратную матрицу через $E(5)$, а правый вектор-столбец — через C :

$$D = E(5) \times C.$$

Для решения поставленной задачи требуется найти три функции $x(t)$, $y(t)$, $z(t)$; для этого запишем соответствующие начальные условия в следующем виде:

$$\begin{aligned} D_x &= E(5) \times C_x; \\ D_y &= E(5) \times C_y; \\ D_z &= E(5) \times C_z. \end{aligned} \quad (3.25)$$

Ниже приводится исходный текст программы, которая выполняет визуализацию параметрической кубической кривой, используя приведенные соотношения:

```
void DrawCurveFwdDif (
int n,
/* Число шагов, необходимых для рисования кривой */
double x, double Dx, double D2x, double D3x,
/* Начальные значения для x(t), найденные при t = 0 по формуле
(3.25) */
double y, double Dy, double D2y, double D3y,
/* Начальные значения для y(t), найденные при t = 0 по формуле
(3.25) */
double z, double Dz, double D2z, double D3z
/* Начальные значения для z(t), найденные при t = 0 по формуле
(3.25) */
/* Для расчета Dx, Dy, Dz используется шаг 5 = 1/π */
)
{
int i;
MoveAbs3 (x, y, z);
```



```
/* Перейти к началу кривой */
for(r=0; i < n; i++) {
x+=DX; Dx+=Л2x; Л2x+=Д3X;
y+=Ay; ЛУ+=Л2У; Л2У+=Д3У;
z+=Z\z; Az+=A2z; Д22+=Д3г;
LineAbs(x,y,z)
/* Рисование короткого сегмента линии */
}
} /* Конец процедуры DrawCurveFwdDif */
```

Описанный способ отличается высокой вычислительной эффективностью. На расчет одной точки в трехмерном пространстве он затрачивает всего лишь девять операций сложения и ни одной операции умножения. Даже если учесть трудоемкость инициализации, которая добавляет несколько операций сложения и умножения, предложенный метод оказывается намного более эффективным по сравнению с вычислением по схеме Горнера. Недостатком метода следует считать возможное накопление ошибки в процессе вычислений.

Вторым способом визуализации кубических кривых, заданных в параметрической форме, является рекурсивное подразбиение. Это адаптивная процедура, которая прекращает обрабатывать сегмент по достижении им достаточной линейаризации. Технические детали алгоритмов и их реализация отличаются для различных типов кривых. Отличаются и критерии линейаризации, поэтому можно привести только общую схему алгоритма визуализации:

```
void DrawCurveRecSub (curve, e)
{
if (Straight (curve, e))
/* Проверка принадлежности контрольных точек заданной окрестности E линии */
DrawLine (curve);
else {
SubdivideCurve (curve, leftCurve, rightCurve);
DrawCurveRecSub (leftCurve, e);
DrawCurveRecSub (rightCurve, e);
}
}
/* Конец процедуры DrawCurveRecSub */
```

В этой программе условный оператор возвращает истинное значение, если проверяемая кривая обладает достаточной линейностью.

Процедура рекурсивного подразбиения оказывается особенно эффективной для кривых Безье. Приблизительные оценки трудоемкости показывают, что для одной операции деления кривой требуются шесть микроопераций сдвига и столько же сложения. Проверка линейности кривых Безье не вызывает затруднений, поскольку опирается на простые манипуляции с выпуклой оболочкой и контрольными точками. Схема такой проверки показана на рис. 3.26. Если

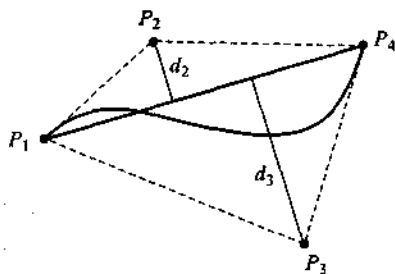


Рис. 3.26. Проверка линейности
конвой Безье

большее из расстояний d_2 и $c/3$ становится меньше, чем заданная окрестность δ , то кривая с достаточной точностью может быть заменена прямым отрезком, опирающимся на конечные точки P_1 и P_4 . Для кривых Безье конечными точками служат первая и последняя контрольные точки P_1 и P_4 .

Практика показывает, что преобразование кривых различного типа в кривые Безье дает возможность применить эффективную процедуру визуализации, основанную на рекурсивном подразбиении. Эта схема позволяет избе-

жать лишних вычислений, неизбежных при использовании алгоритма, основанного на вычислении правых разностей. Чтобы получить приемлемую точность фрагментов с высокой кривизной, шаг 5, с которым вычисляются правые разности, должен иметь небольшое значение. Для сегментов с формой, приближенной к прямой, величина шага может быть увеличена. С другой стороны, алгоритм рекурсивного подразбиения расходует дополнительные вычислительные ресурсы на проверку линейности фрагментов. Неплохой альтернативой является применение рекурсивного алгоритма с фиксированной глубиной, что позволяет избежать проверки линейности.

3.2.10. Сравнение кубических кривых

Свойства параметрических кривых могут значительно отличаться друг от друга. Восприимчивость к интерактивным изменениям, гладкость в точках сочленения, скорость визуализации, трудоемкость вычислений — это далеко неполный перечень критериев, которые влияют на успешность применения типа кривых в прикладных областях и задачах. В большинстве случаев вопрос о выборе оптимальной формы представления кривых не стоит. Ранее было показано, что многие математические модели кривых преобразуются одна в другую без значительных потерь и искажений. Например, неоднородные В-сплайны могут быть использованы для внутреннего представления кривых, тогда как пользователю удобнее работать с узлами и касательными, представленными в форме Безье или Эрмита. Многие популярные графические редакторы поддерживают пользовательский интерфейс кривых Эрмита, которые внутри редактора обрабатываются как кривые Безье, описанные на языке PostScript. Развитые CAD-системы характеризуются еще большей гибкостью. Они оставляют на усмотрение пользователя выбор одной из форм представления кривых, которыми могут быть кривые Безье, Эрмита, однородные и неоднородные В-сплайны. В большинстве случаев для обработки внутри системы выбираются неоднородные В-сплайны, поскольку они обладают наибольшей общностью и самой высокой гибкостью. > . - . »*.

Таблица 3.1

Свойства кривой	Тип кривой				
	Эрмита	Безье	Однородные В-сплайны	Неоднородные В-сплайны	Р-сплайны
Выпуклая оболочка, заданная контрольными точками	Неприменимо	Да	Да	Да	Да
Интерполяция некоторых контрольных точек	Да	Да	Нет	Нет	Нет
Интерполяция всех контрольных точек	Да	Нет	Нет	Нет	Нет
Простота подразделения	Приемлемая C^0, G^0	Хорошая C^0, G^0	Средняя	Высокая	Средняя
Типичная гладкость в узлах класса	$C \setminus G^x$	$C \setminus G'$	$C \setminus G^2$	$C \setminus G^2$	C^0, G^2
Достижимая гладкость в узлах класса	$C \setminus G^x$	$C \setminus G'$	$C \setminus G^2$	C^2, G^2	$C \setminus G^2$
Число управляющих параметров	4	4	4	5	6

В табл. 3.1 приведена сводка важнейших технических параметров типов кривых. Для сравнения даны свойства сравнительно редкого типа кривых — Р-сплайны. В таблице не упомянута такая важная характеристика, как простота обращения. Подобное сравнение было бы необъективным, поскольку это свойство зависит от особенностей реализации в пакете или редакторе.

Достижимая гладкость в точках сочленения описывается классом, который может иметь кривая после выполнения некоторых дополнительных условий, например коллинеарность контрольных точек или их совпадение. Принадлежность классу C — более жесткое условие, чем принадлежность классу G'' . Выполнимость первого условия всегда влечет за собой истинность второго.

Во многих геометрических задачах автоматизированного проектирования требуется выполнимость условия G^2 , что ограничивает выбор формы представления кривых тремя типами сплайнов. Самые жесткие ограничения на свободу формообразования накладывают однородные В-сплайны. Напротив, неоднородные сплайны предлагают пользователю наибольший контроль над формой кривой.

В настоящее время стала общепринятой обработка кривых, когда пользователь настраивает ее форму, меняя положение узлов и касательных. Подобный набор интерактивных приемов поддерживается многими графическими редакторами и принят сообществом пользователей в качестве стандарта де-факто. К сожалению, контрольные точки В-сплайнов в общем случае не лежат на самой кривой. Некоторые графические программы скрывают от пользователя контрольные точки, предоставляя в его распоряжение узлы сплайна. Если пользователь смещает помеченный узел на некоторое расстояние с координатами A_x, A_y , то программа

выполняет пересчет весов контрольных точек и реализует подлинное смещение узла на величину $Ax' \& x$, $Ay' \& Ay$, затем восстанавливает положение курсора. Этот процесс повторяют циклически до получения искомой формы сплайна.

3.3. Параметрические бикубические поверхности

Параметрические бикубические поверхности во многих отношениях представляют собой обобщение параметрических кубических кривых. Напомним, что в самом общем виде параметрические кубические кривые описывались следующим выражением:

$$Q(t) = T \times M \times G,$$

где G — геометрический вектор, представляющий собой постоянную величину.

Чтобы сохранить общность изложения, изменим принятый способ обозначения. Во-первых, заменим параметр t на s и перепишем основное уравнение

$$Q(s) = S \times M \times G.$$

Пусть значения геометрического вектора меняются по некоторой функции, зависящей от аргумента t . Учитывая это обобщение, запишем основное уравнение в следующем виде:

$$Q(s, t) = S \times M \times G(t) = S \times M \times \begin{bmatrix} G_1(t) \\ G_2(t) \\ G_3(t) \\ G_4(t) \end{bmatrix}. \quad (3.26)$$

При фиксированных значениях параметра $t = t_0$ выражение $Q(s, t_0)$ представляет собой кривую, поскольку в этом случае геометрический вектор $G(t_0)$ является константой. Пусть параметр t принимает значения t_0, t_1, t_2, \dots которые незначительно отличаются по величине и лежат в интервале $[0, 1]$. В результате будет порождено семейство близких по форме кривых $Q(s, t)$. Множество этих кривых задает поверхность в трехмерном пространстве. Если координаты геометрического вектора $G_i(t)$ описываются кубическими полиномами, то говорят, что созданная поверхность относится к классу параметрических бикубических поверхностей. В дальнейшем будем исходить из этого предположения и рассматривать только поверхности подобного вида.

Представим $G_i(t)$ в следующем виде:

$$G_i(t) = T \times M \times G_i,$$

где

$$G_i = [g_{i1}, g_{i2}, g_{i3}, g_{i4}]^T.$$

Здесь использовано полужирное начертание для отличия от геометрического вектора кривых, обозначаемого G . В этом выражении g_i представляет собой первый элемент геометрического вектора кривой G_i ; g_2 — второй и т. д.

Перепишем это уравнение, используя тождество $(Ax Bx C)^T = (C^T x B^T x A^T)$:

$$G_i(t) = G_i^T \times M^T \times \Gamma = [g_{i1}, g_{i2}, g_{i3}, g_{i4}] \times M^T \times f.$$

Л.

Распространим полученное выражение на все четыре координаты геометрического вектора и подставим результат в уравнение (3.26):

$$Q(s, t) = SxMx \begin{bmatrix} §11 & §12 & §13 & §14 \\ §21 & §22 & §23 & §24 \\ §31 & §32 & §33 & §34 \\ §41 & §42 & §43 & §44 \end{bmatrix} xM^T x T \quad (3.27)$$

Эту громоздкую зависимость можно представить в более компактной форме:

$$Q(s, t) = SxMxGxM^T x T^T; \quad 0 < s; \quad * < 1. \quad (3.28)$$

Перепишем уравнение отдельно для каждой координаты:

$$\begin{aligned} x(s, t) &= SxMxG_x xM^T x \Gamma^T; \\ y(s, t) &= SxMxG_y xM^T x T^T; \\ z(s, t) &= SxMxG_z xM^T x T^T. \end{aligned} \quad (3.29)$$

Совокупность уравнений (3.29) представляет собой общую модель параметрических бикубических поверхностей. Далее рассмотрены самые интересные частные случаи частных случаев поверхностей данного типа.

3.3.1. Поверхности Эрмита

Поверхности Эрмита полностью задаются геометрической матрицей G_H размера 4×4 . Способ генерации этой матрицы полностью совпадает с подходом, использованным при выводе уравнения (3.28). Немного уточним этот вывод, применив принятую схему к $x(s, t)$. Воспользуемся уравнением (3.4), которое задает определение кривых Эрмита. Заменив в нем параметр t на s , получим

$$x(s) = SxM_H xG_{Hx}.$$

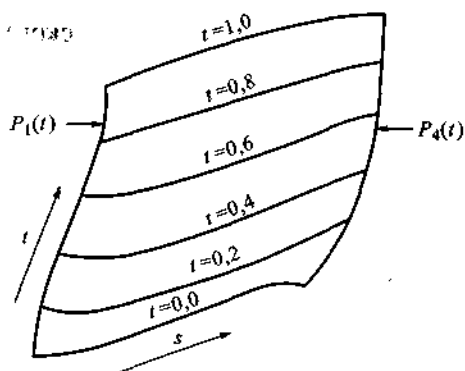


Рис. 3.27. Способ представления бикубической поверхности

По аналогии с рассуждением, приведенным в подразд. 3.2.1, запишем это выражение с переменным геометрическим вектором:

$$x(s, t) = SxM_{Hx}G_{Hx}(t) = SxM_{Hx} \begin{bmatrix} P_1(t) \\ PAO \\ R_1(t) \\ R_4(t) \end{bmatrix}_x. \quad (3.30)$$

Здесь функции $P_{1x}(t)$, $P_{4x}(t)$ задают координаты x конечной и начальной точек кривой по параметру s .

Похожим образом описываются функции $R_u(t)$, $R_{4x}(t)$, они определяют касательные векторы в начальной и конечной точках кривой. Каждое допустимое значение параметра t определяет положение двух концевых точек и касательных векторов в этих точках.

На рис. 3.27 показан способ представления бикубической поверхности. Кривая $P_1(t)$ получена при $s = 0$, а кривая $P_4(t)$ — при $s = 1$. Прочие кубические кривые зависят от аргумента s и получены для последовательности значений $t = 0,0; 0,2; 0,4; 0,6; 0,8; 1,0$. Поверхность, изображенная на рис. 3.27, по сути представляет собой кубическую интерполяцию куска поверхности, ограниченного четырьмя линиями $P_1(t) = Q(0, t)$, $P_4(t) = Q(1, t)$, $Q(s, 0)$, $Q(s, 1)$.

Особо выделяют частный случай бикубических поверхностей, когда все элементы границы $\langle Q(0, t), Q(1, t), Q(s, 0), Q(s, 1) \rangle$ являются прямыми линиями. Поверхности такого сорта называются линейчатыми. Если прямые линии принадлежат одной плоскости, то задаваемый ими фрагмент представляет собой четырехсторонний многоугольник.

Представим компоненты $P_{1x}(t)$, $P_{4x}(t)$, $R_{1x}(t)$, $R_{4x}(t)$ в принятой для кривых Эрмита форме:

$$\begin{aligned} P_{1x}(t) &= TxM_{Hx} \begin{bmatrix} §\Pi \\ §12 \\ §13 \\ §14 \end{bmatrix}_x; & P_{4x}(t) &= TxM_{Hx} \begin{bmatrix} §21 \\ §22 \\ §23 \\ §24 \end{bmatrix}_x; \\ R_{1x}(t) &= TxM_{Hx} \begin{bmatrix} "§31" \\ §32 \\ §33 \\ §34 \end{bmatrix}_x; & R_{4x}(t) &= TxM_{Hx} \begin{bmatrix} ~§4, ' \\ §42 \\ §43 \\ §44 \end{bmatrix}_x. \end{aligned} \quad (3.31)$$

Зависимости (3.31) можно записать в более компактной форме:

$$[ад, ад, вд, R_4(t)]_x = T x M_H x G_{Hx}^T, \quad (3.32)$$

где

$$G_{Hx} = \begin{bmatrix} g_{11} & g_{12} & g_{13} & g_{14} \\ g_{21} & g_{22} & g_{23} & g_{24} \\ g_{31} & g_{32} & g_{33} & g_{34} \\ g_{41} & g_{42} & g_{43} & g_{44} \end{bmatrix}_x$$

Выполнив транспонирование обеих частей уравнения (3.32), получим

$$\begin{bmatrix} ад \\ ад \\ од \\ R_4(t) \end{bmatrix}_x = \begin{bmatrix} g_{11} & g_{12} & g_{13} & g_{14} \\ g_{21} & g_{22} & g_{23} & g_{24} \\ g_{31} & g_{32} & g_{33} & g_{34} \\ g_{41} & g_{42} & g_{43} & g_{44} \end{bmatrix}_x x M_H^T x T^T = G_{Hx} x M_H^T x T^T. \quad (3.33)$$

Подставив выражение (3.33) в уравнение (3.30), имеем

$$x(s, t) = S x M_H x G_{Hx} x M_H^T x T^T.$$

По аналогии можно записать аналитические выражения для двух других координат:

$$y(s, t) = S x M_H x G_{Hy} x M_H^T x T^T;$$

$$z(s, t) = S x M_H x G_{Hz} x M_H^T x T^T.$$

Для поверхностей Эрмита три матрицы G_{Hx} , G_{Hy} , G_{Hz} размера 4×4 играют такую же роль, что и матрица G_H для кривых. Смысл элементов этих матриц станет яснее, если обратиться к ранее приведенным уравнениям (3.30) и (3.31). Элемент g_{1x} представляет собой иное обозначение для $x(0, 0)$, поскольку это стартовая точка для кривой $P_{\backslash x}(t)$. Равным образом g_{2x} есть ни что иное, как $x(0, 1)$, поскольку этот элемент представляет собой конечную точку кривой $P_{\backslash x}(t)$. Кроме того, g_{13x} есть другое обозначение для $\frac{\partial x}{\partial t}(0, 0)$, поскольку является начальным касательным вектором для $P_{\backslash x}(t)$. Наконец, g_{33x} есть $\frac{\partial^2 x}{\partial s \partial t}(0, 0)$, поскольку является стартовым касательным вектором для $R_{\backslash x}(t)$.

Используя приведенные интерпретации, перепишем матрицу G_{Hx} в следующем виде:

$$G_{Hx} = \begin{bmatrix} x(0,0) & x(0,1) & \frac{\partial x(0,0)}{\partial t} & \frac{\partial x(0,1)}{\partial t} \\ 410) & x(1,1) & \frac{\partial x(1,0)}{\partial t} & \frac{\partial x(1,1)}{\partial t} \\ \frac{\partial x(0,0)}{\partial s} & \frac{\partial x(0,1)}{\partial s} & \frac{\partial^2 x(0,0)}{\partial s \partial t} & \frac{\partial^2 x(0,1)}{\partial s \partial t} \\ \frac{\partial x(1,0)}{\partial s} & \frac{\partial x(1,1)}{\partial s} & \frac{\partial^2 x(1,0)}{\partial s \partial t} & \frac{\partial^2 x(1,1)}{\partial s \partial t} \end{bmatrix} \quad (3.34)$$

По виду и смыслу матрица (3.34) делится на характерные квадратные подматрицы размером 2×2 . Рассмотрим их содержание. В верхней левой части матрицы расположены координаты четырех угловых вершин фрагмента поверхности. Верхний правый и левый нижний квадранты матрицы описывают координаты x касательных векторов вдоль каждого параметрического направления фрагмента поверхности. Нижняя правая часть матрицы объединяет вторые частные производные по аргументам s и t . Эти величины описывают скручивание в углах фрагмента поверхности. Чем выше значение вторых производных, тем сильнее проявляют себя деформации данного типа.

На рис. 3.28 показан пример фрагмента поверхности Эрмита; угловые точки этого фрагмента помечены элементами матрицы G_{Hx} . Каждый вектор на этом рисунке представляет собой кортеж длиной три элемента, которые заимствованы из уравнения (3.34).

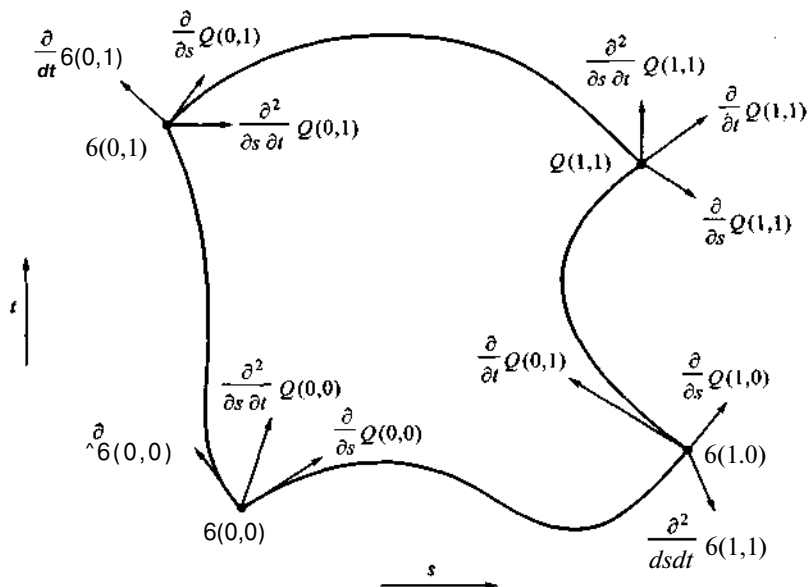


Рис. 3.28. Пример фрагмента поверхности Эрмита.



Рис. 3.29. Пример поверхности, состоящей из двух пограничных фрагментов

Форма эрмитовых бикубических фрагментов представляет собой важный частный случай объектов другого вида, так называемых поверхностей Кунса. Фрагменты поверхностей Кунса могут ограничиваться кривыми любого типа и произвольной крутизны. Подобно тому, как кривые Эрмита в узлах обеспечивают гладкость класса C^1 и G^1 , поверхности Эрмита гарантируют принадлежность этим классам в окрестности сочленения различных кусков.

Рассмотрим эти свойства более подробно. Чтобы обеспечить гладкость класса C^0 , граничные линии стыкуемых кусков должны быть идентичны. Это значит, что контрольные точки обеих поверхностей полностью совпадают на граничной кривой. Для выполнения условия C^1 требуется полное совпадение на границе не только контрольных точек, но и касательных векторов, а также вторых смешанных производных.

Условие G^1 является менее жестким. В этом случае должны совпадать только направления касательных векторов; их величины могут быть различными. Пусть, как показано на рис. 3.29, два фрагмента имеют общую границу. Причем у первого фрагмента на этой границе аргумент $s = 1$, а у второго — $s = 0$. Тогда для выполнения гладкости по классу G^1 геометрические матрицы фрагментов должны иметь следующий вид:

$$\begin{bmatrix} - & - & - & - \\ g_{21} & g_{22} & g_{23} & g_{24} \\ - & - & - & - \\ g_{41} & g_{42} & g_{43} & g_{44} \end{bmatrix} \begin{bmatrix} g_{21} & g_{22} & g_{23} & g_{24} \\ - & - & - & - \\ kg_{41} & kg_{42} & kg_{43} & kg_{44} \\ - & - & - & - \end{bmatrix}$$

Здесь коэффициент $k > 0$, а прочерками обозначены позиции, которые могут принимать любые значения.

3.3.2. Поверхности Безье

Математическое описание бикубических поверхностей Безье можно получить по единой с поверхностями Эрмита схеме. Если выполнить выкладки, аналогичные предыдущему разделу, то получим

$$\begin{aligned} x(s,t) &= SxM_BxG_{Bx}xM_B^TxT; \\ y(s,t) &= SxM_BxG_{By}xM_B^TxT; \\ z(s,t) &= SxM_BxG_{Bz}xM_B^TxT \end{aligned} \quad (3.35)$$

Геометрическая матрица поверхности Безье состоит из 16 контрольных точек. На рис. 3.30 показан фрагмент поверхности этого типа и положение ее контрольных точек.

Поверхности Безье, как и их двухмерные аналоги, предоставляют оператору удобные интерактивные средства управления. Для создания необходимой геометрической конфигурации разрешается менять положение контрольных точек и касательных векторов. Применение поверхностей Безье в качестве внутренней расчетной геометрической модели имеет свои преимущества. Среди них прежде всего следует упомянуть о существовании выпуклых оболочек, которые подробно обсуждались ранее, и о существовании эффективной процедуры визуализации.

Для принадлежности поверхностей классам C^0 и G^0 требуется, чтобы граничные линии смежных кусков кривых Безье имели четыре совпадающие контрольные точки. Принадлежность классу G^1 обеспечивается более сложным условием. Для этого требуется совпадение четырех контрольных точек обоих смежных фрагментов. Таковыми для примера, приведенного на рис. 3.31, являются точки P_{14} , P_{2b} , P_u , P_{AA} . Кроме того, наборы точек, лежащие по обе стороны от граничных узлов, должны быть коллинеарными, т. е. лежать на одной прямой. Для рис. 3.31 это (P_{13}, P_u, P_{15}) , (P_{22}, P_u, P_{15}) , (P_{23}, P_u, P_{15}) и (P_{43}, P_u, P_{15}) .

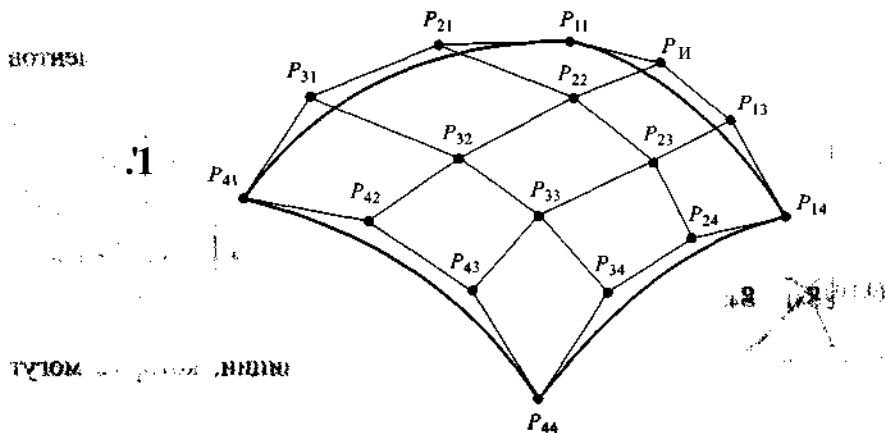


Рис. 3.30. Фрагмент поверхности Безье и ее контрольные точки

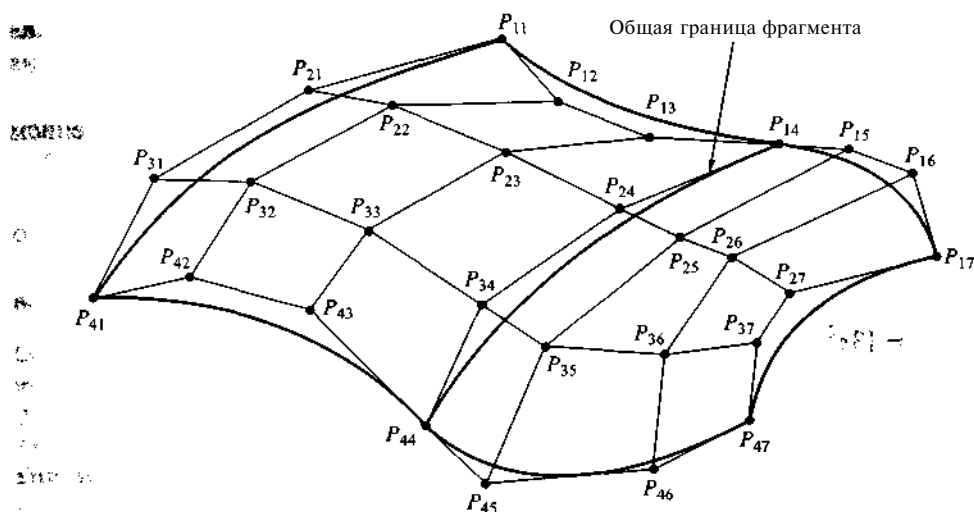


Рис. 3.31. Два фрагмента поверхности Безье, граничащие по P₁₄, P₂₄, P₃₄, P₄₄

3.3.3. Бисплайновые поверхности

Бисплайновыми называются поверхности, которые описываются системой уравнений следующего вида:

$$\begin{aligned} x(s, t) &= SxM_{B_s} \times G_{B_{st}} \times M_{B_s}^T \times T^T; \\ y(s, t) &= S \times M_{B_s} \times G_{B_s} \times M_{B_s}^T \times T^T; \\ z(s, t) &= S \times M_{B_s} \times G_{B_s} \times M_{B_s}^T \times T^T. \end{aligned} \quad (3.36)$$

Для поверхностей этого вида на границах двух фрагментов автоматически выполняется принадлежность классу C². При этом не требуется накладывать никаких дополнительных условий на контрольные точки за исключением их совпадения. Как было показано в подразд. 3.2.4, дублирование контрольных точек влечет за собой нарушение непрерывности.

Бикубические неоднородные бисплайновые поверхности главными свойствами напоминают своих двумерных аналогов — неоднородные рациональные В-сплайны. В частности, техника подразбиения и визуализации почти без изменений переносится на трехмерный случай.

3.3.4. Нормали поверхностей

Во многих прикладных задачах необходимо знать величину и положение нормали фрагмента бикубической поверхности. Эти сведения требуются для

визуализации поверхности, расчета затенения, проверки условий непересечения при проектировании механических узлов, вычисления траекторий перемещения и во многих других проектных ситуациях.

Для вывода аналитического выражения нормали воспользуемся уравнением (3.28), из которого легко найти два касательных вектора к поверхности $Q(s, t)$.

Элементарные вычисления дают выражения для касательного вектора по s :

$$\begin{aligned} \frac{\partial}{\partial t} Q(s, t) &= -(SxMxGxM^T x T^T) = \frac{\partial}{\partial s} (S)xMxGxM^T x T^T = \\ &= [3s^2, 2s, 1, 0] \times M \times G \times M^T \times T^T. \end{aligned}$$

Аналогично найдем касательный вектор по t :

$$\begin{aligned} \frac{\partial}{\partial t} Q(s, t) &= -(SxMxG \quad xM^T x T^T) = SxMxG \quad xM^T x \frac{\partial}{\partial t} (T^T) = \\ &= SxMxGxM^T x [3t^2, It, 1, 0]^T. \end{aligned}$$

Оба полученных касательных вектора расположены параллельно поверхности $Q(s, t)$ в любой точке (s, t) , поэтому их векторное произведение является перпендикуляром к $Q(s, t)$. Напомним, что касательные векторы представляют собой тройки, координаты которых есть значения для x , y и z для бикубической поверхности. С учетом этого запишем выражение для нормали поверхности:

$$\frac{\partial}{\partial s} Q(s, t) \times \frac{\partial}{\partial t} Q(s, t) = [y_s z_t - y_t z_s, \quad z_s x_t - z_t x_s, \quad x_s y_t - x_t y_s],$$

где x_s — это сокращенная запись для координаты x касательного вектора по s ; y_s — сокращенная запись координаты y и т. д.

Если в некоторой точке поверхности оба касательных вектора равны нулю, то их векторное произведение тоже равно нулю. Это означает, что в данной точке представление о нормали к поверхности не имеет смысла. Напомним, что касательные векторы могут быть равны нулю в смежных точках, которые принадлежат классу C^1 , но не обладать принадлежностью классу G^1 .

Аналитическое выражение для нормали — полином высокой степени (две переменные и пятая степень), поэтому его вычисление представляет собой задачу большой трудоемкости. Во многих графических редакторах нормали подсчитываются приблизительно — по некоторым аппроксимирующим зависимостям невысокой степени.

3.3.5. Визуализация бикубических поверхностей

Для бикубических поверхностей, подобно кубическим параметрическим кривым, существует два основных способа визуализации: итеративное вычисле-

ние полиномов, описывающих поверхность, и процедура последовательного подразделения.

Рассмотрим способ визуализации, основанный на вычислении. Итерационное вычисление дает хорошие результаты для бикубических поверхностей, типичная форма которых показана на рис. 3.32. Этот фрагмент поверхности представляет собой два семейства кривых: одно из них состоит из кривых, зависящих от аргумента t , другое — от аргумента s .

Ниже дано описание процедуры визуализации фрагмента бикубической поверхности как сетки кривых:

```
typedef double Coeffs[4][4][3];
void DrawSurface(
    Coeffs coefficients,
    /* Коэффициенты для  $Q(s,t)$  */
    int n_s,
    /* Число кривых по константе  $s$ . Обычно оно равно 5-10 */
    int n_t,
    /* Число кривых по константе  $t$ . Обычно оно равно 5-10 */
    int n)
    /* Число шагов для воспроизведения каждой кривой. Обычно оно
    равно 20-100 */
    {
        double S=1.0/π;
        /* Размер шага для рисования кривых */
        double S_s=1.0/(n_s - 1);
```

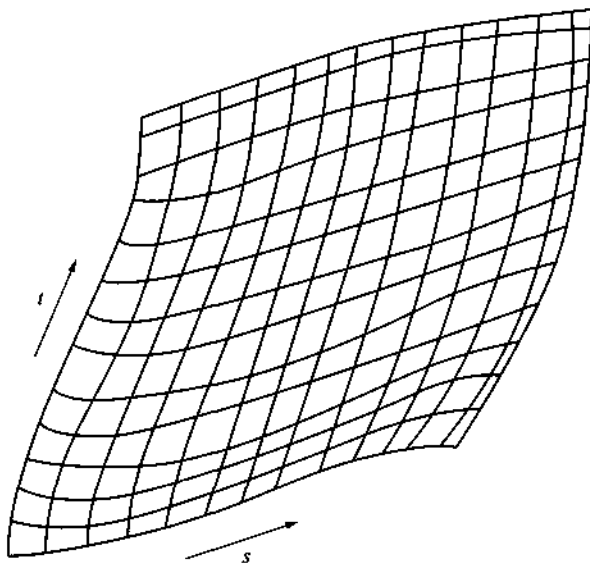


Рис. 3.32. Пример бикубической поверхности, допускающий простую организацию процесса визуализации

```

/*Размер шага для перехода к следующей кривой по константе t*/
double 5t=1.0/(nt - 1);
/*Размер шага для перехода к следующей кривой по константе s*/
int i,j; double v, t;
/* Рисование ns.кривых по константе s* /
for (i=0, s=0.0; i<ns; i++, s+=5s) {
/* Рисование кривой по константе s, t меняется от 0 до 1 */
/* X, Y, и Z представляют собой функции, по которым выполняются
расчеты */
MoveAbs3 (X(s,0.0), Y(s,0.0), Z(s,0.0));
for (j=1, t=5; j<n; j++, t+=5) {
/* n - 1 шагов используется для каждой кривой */
LineAbs3 (X(s, t), Y(s, t), Z(s, t));
}
}
/* Рисование nt кривых по константе t */
for (i=0, t=0.0; i<nt; i++, t+=5t) {
/* Рисование кривой по константе, s изменяется от 0 до 1 */
MoveAbs3 (X(0.0, t), Y(0.0, t), Z(0.0, t));
for (j=1, s=5; j<n; j++, s+=5) {
/* n-1 шагов используется для каждой кривой */
LineAbs3 (X(s, t), Y(s, t), Z(s, t));
}
}
/* Конец процедуры DrawSurface */

```

Переборный алгоритм расчета поверхностей оказывается намного более трудоемким, чем подобный алгоритм для кривых. Действительно, уравнения поверхности должны быть вычислены $2/8^2$ раз: при $5 = 0,1$ трудоемкость равна 200, при $8 = 0,01$ она возрастает до 20 000, что заставляет искать альтернативные способы вычисления. Одним из возможных является метод визуализации, основанный на вычислении правых разностей, который подробно рассмотрен в подразд. 3.2.9.

Обратимся к процедуре визуализации, основанной на схеме рекурсивного подразделения. Оказывается, что для бикубических поверхностей она представ-

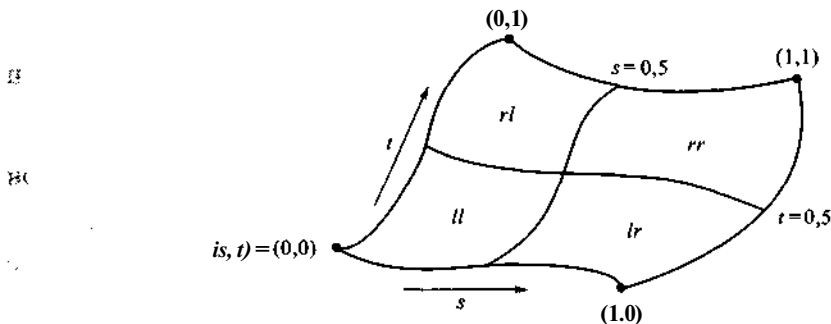


Рис. 3.33. Кусок поверхности разбит на четыре фрагмента ll , lr , rl , rr

ляет собой обобщение аналогичного алгоритма визуализации кривых. Напомним, что процедура *DrawCurveRecSub* была рассмотрена в подразд. 3.2.9, посвященном визуализации кривых. Основное отличие процедур визуализации кривых поверхностей состоит в том, что кусок бикубической поверхности разбивается на четырехугольники, которые повторными рекурсивными вызовами процедуры визуализации уменьшаются в размерах и своей формой все более приближаются к плоскости. Один шаг этой процедуры показан на рис. 3.33. Здесь фрагмент бикубической поверхности разделен на четыре фрагмента и каждый из полученных фрагментов обладает более высокой «плоскостностью», чем исходная поверхность.

Критерием остановки служит достаточное приближение к плоскости фрагментов разбиения. Тест на планарность является обобщением аналогичной процедуры проверки кубических кривых. Строится вспомогательная плоскость, проходящая через три из четырех контрольных точки. Затем находятся расстояния от этой плоскости до 13 остальных контрольных точек. Если максимальная длина оказывается меньше, чем некоторая наперед заданная константа ϵ , то проверка считается успешной.

В процессе рекурсивного подразбиения может возникать проблема, связанная с появлением разрывов между аппроксимирующими прямоугольниками. На рис. 3.34 два смежных куска находятся на разных уровнях обработки. Правый нижний фрагмент был разделен одинаково с верхней частью поверхности, поэтому между фигурами возникла «аппроксимирующая трещина». Этого эффекта можно избежать, задавая фиксированную глубину рекурсии или устанавливая достаточно малое значение окрестности ϵ . Оба решения влекут дополнительные вычислительные расходы на избыточные итерации процесса подразбиения. Более эффективное решение связано с изменением схемы деления поверхности (рис. 3.35).

Прямой алгоритм делит поверхность на четырехугольники $ABCE$, $EFGD$ и $GDCH$. Для уменьшения зазора схема деления изменяется на $ABCE$, $EFGD'$ и $GD'CH$, т. е. вместо точки D используется точка D' .

Во многих задачах требуется отобразить только часть бикубической поверхности, примерами таких поверхностей являются поверхности с разрывами или запре-



Рис. 3.34. Возникновение разрыва аппроксимации

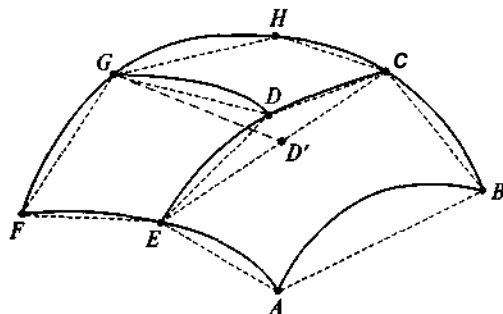


Рис. 3.35. Схема сокращения возможных разрывов в процессе рекурсивного подразбиения

шенными зонами. Проблема визуализации поверхностей с разрывами решается при помощи построения так называемой отсекающей кривой. Это вспомогательный сплайн, заданный в координатах (s, t) , а не в исходных координатах (x, y, z) .

3.3.6. Поверхности второго порядка

Поверхности второго порядка описываются уравнением неявного вида

$$f(x, y, z) = ax^2 + by^2 + cz^2 + 2dxy + 2eyz + 2fzx + 2gx + 2hy + 2jz + k = 0. \quad (3.37)$$

Это уравнение порождает целое семейство поверхностей второго порядка (квадрик). Например, положим $a = b = c = -k = 1$, а остальные коэффициенты возьмем равными нулю. Этот частный случай уравнения (3.37) определяет сферу с центром в начале координат. Если все коэффициенты от a до j равны нулю, то получится плоскость.

Существуют области геометрического моделирования, где квадратичные поверхности оказывают большую пользу. В теоретической физике квадратичные поверхности применяются для построения моделей молекул, в системах автоматизированного проектирования они служат средством твердотельного моделирования механических узлов. Напомним, что рациональные кубические кривые могут описывать конические сечения, аналогично, рациональные бикубические поверхности способны представлять поверхности второго порядка. Существуют и другие причины для изучения квадратичных поверхностей:

- существование простого алгоритма вычисления нормалей к поверхностям;
- возможность проверки принадлежности точки поверхности;
- вычисление координаты z по известным значениям координат x и y ;
- простота расчета области пересечения двух поверхностей второго порядка и др.

Запишем уравнение (3.37) в компактной матричной форме

$$P^T x Q x P = 0,$$

где

$$Q = \begin{bmatrix} a & d & f & g \\ d & b & e & h \\ f & e & c & j \\ g & h & j & k \end{bmatrix}; \quad P = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}.$$

Поверхность, представленная матрицей Q , легко деформируется и перемещается. Если задана трансформационная матрица размера 4×4 , то новое состояние поверхности Q' может быть представлено в следующем виде:

$$Q' = (M^{-1})^T Q M^{-1}.$$

Также легко найти выражение для нормали к поверхности. Пусть поверхность второго порядка задана неявным выражением $f(x, y, z) = 0$, тогда выра-

жение для нормали можно записать как $\left[\frac{df}{dx}, \frac{df}{dy}, \frac{df}{dz} \right]^T$. Очевидно, что нормали к поверхностям второго порядка значительно проще вычислить, чем для бикубических поверхностей.

3.4. Твердотельное моделирование

3.4.1. Основные положения

В предыдущем разделе рассматривались математические модели кривых и поверхностей, расположенных в двумерном и трехмерном пространствах. Существует множество прикладных задач, которые с достаточной полнотой описываются кривыми и поверхностями, но возможности рассмотренного аппарата не покрывают всех потребностей геометрического моделирования. Например, в процессе автоматизированного формообразования детали требуется строго различать внутренние и внешние области ее замкнутой геометрической модели. В большинстве задач проектирования процессов обработки и сборки объект должен быть представлен в виде замкнутой объемной конфигурации. Программирование поведения роботов и робототехнических систем в среде с препятствиями также требует подобного описания сцены. Моделирование работы физических механизмов не обходится без учета пространственного фактора. На твердотельных моделях базируются программы конечно-элементного анализа, технологические системы с числовым программным управлением, модули, предназначенные для расчета масс-инерционных характеристик детали и множество других задач, решаемых в системах CAD/CAM/CAE.

Описание объема трехмерных тел не является простым расширением методов моделирования кривых и поверхностей. Эта проблема требует особых под-

ходов и специально разработанных средств. Краткому введению в проблематику твердотельного моделирования и посвящен этот раздел.

Следует строго различать форму геометрической модели и ее выразительные возможности. Наличие видимого объема у геометрического объекта не означает его способность моделировать все или даже некоторые объемные характеристики. Рассмотрим это противоречие на примере куба, показанного на рис. 3.36.

На рис. 3.36, *а* фигура задана вершинами и линиями. Если условиться, что каждый боковой набор, состоящий из четырех линий, определяет грань, то рисунок представляет куб. Однако то же самое описание может изображать и фигуру с отверстиями, у которой некоторые или все боковые грани отсутствуют. Пусть все боковые наборы соединенных линий считаются гранями, но даже этого соглашения недостаточно для описания объемного тела.

На рис. 3.36, *б* показана фигура, описание которой поглощает описание левой фигуры. Все точки и линии левой фигуры входят в правую фигуру, которая, кроме того, содержит еще одну свободную грань *BCIJ*, поэтому не является объемом. Очевидно, что для корректного описания объемной геометрии на используемые математические средства следует наложить дополнительные ограничения. Ниже приведен примерный список таких требований:

- математическая модель твердотельной геометрии должна отличаться высокой универсальностью, достаточной для описания различных физических объектов и проектных ситуаций. Очень важны такие качества модели, как непротиворечивость и отсутствие двусмысленностей. Этим критериям не отвечает описание, приведенное на рис. 3.36;

- модель должна обладать свойством однозначности. Это значит, что каждая трехмерная форма получает единственное описание в терминах данной модели. Если требование однозначности обеспечено, то задача проверки эквивалентности формы двух объектов решается намного проще, чем для многозначных моделей;

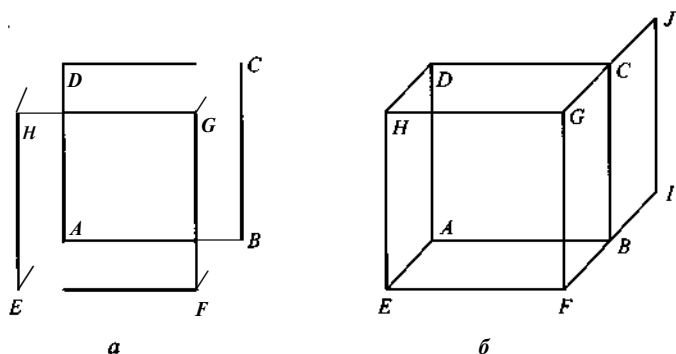


Рис. 3.36. Каркасная модель куба:

а — корректная; *б* — некорректная, вершины $A(0, 0, 0)$; $B(1, 0, 0)$; $C(1, 1, 0)$; $D(0, 1, 0)$; $D(0, 0, 1)$; $F(1, 0, 1)$; $G(1, 1, 1)$; $Ц(0, 1, 1)$; линии: $AB, BC, CD, DA, EF, FG, GH, HE, AE, BF, CG, DH$

- модель должна быть точной. Многие векторные редакторы используют ломанные линии для приближенного представления кривых. В системах автоматизированного проектирования сложные трехмерные объекты часто описываются сочетанием простых геометрических форм. Все это примеры неточных геометрических описаний, в которых используется аппроксимация двухмерных или трехмерных оригиналов. Точная геометрическая модель представляет оригинал без существенных пропусков и изъятий, заметным образом влияющих на качество моделирования;

- описание должно корректно реагировать на типовые операции по преобразованию формы и положения объекта. К числу таких операций относятся прежде всего перемещение, поворот и масштабирование;

- описание должно быть компактным. Это универсальное требование становится еще актуальнее с развитием сетевых и распределенных вычислений. Следует упомянуть и о существовании эффективных способов вычисления различных физических и геометрических характеристик объекта, например массы, момента инерции и пр.

Специалистам в области вычислительной геометрии не известна модель, которая отвечала бы всем перечисленным требованиям. Все способы описания объемной геометрии, применяемые в настоящее время, предоставляют разработчикам и пользователям некоторый компромисс при выполнении отдельных критериев приведенного перечня.

Далее рассмотрены важнейшие способы, наиболее широко применяемые в современных графических системах и пакетах.

Точки, линии, плоские грани представляют собой математические абстракции, поскольку у этих образований, по крайней мере, один размер равен нулю. Физические тела занимают некоторый конечный объем трехмерного пространства и имеют размеры, отличные от нуля. Для описания геометрии реальных тел и предметов используются математические объекты, называемые телами, или твердыми телами.

Существуют объекты, которые допускают приблизительное описание геометрии при помощи самой простой аппроксимации границы — фрагментами плоскостей. Представление такого вида называется плоскогранным (Faceted Representation). Геометрические модели этого типа нашли широкое применение в архитектурном моделировании, программировании игр и компьютерной анимации.

Более продуктивный подход к описанию трехмерных тел дает так называемая конструктивная твердотельная геометрия (Constructive Solid Geometry — CSG), в которой средствами моделирования являются геометрические примитивы и операции над ними. В качестве примитивов обычно выступают простые тела, имеющие точное геометрическое описание в координатной форме: сферы, прямоугольники, цилиндры, конусы, призмы и пр. Над примитивами и производными телами можно выполнять некоторый набор разрешенных операций. В большинстве промышленных систем геометрического моделирования разре-

шенными преобразованиями геометрических операндов являются булевские операции объединения, пересечения и вычитания. Возможности конструктивной твердотельной геометрии весьма значительны. Достаточно сказать, что ее средствами можно описать большую часть деталей из классификатора машиностроительных деталей.

Самый общий подход к моделированию состоит в представлении трехмерных тел в виде совокупности ограничивающих поверхностей, ребер и вершин. Каждая граничная оболочка представляет собой множество соединенных друг с другом поверхностей произвольной формы и включает в себя полное описание своих границ и связей с соседними фрагментами. Этот способ описания объектов называется граничным представлением, в англоязычной литературе и в интерактивных руководствах многих пакетов машинной графики он называется Boundary Representation, или B-*rep*. Представление тел с помощью границ позволяет моделировать трехмерные объекты самой сложной геометрии. Оно допускает множество формообразующих операций над телами, сохраняя при этом единый способ описания их внутреннего устройства.

3.4.2. Регулярные булевские операции

Пусть разработан эффективный способ описания геометрии объемных тел. Отвлечемся от технических деталей такого описания и будем опираться на гипотезу о его существовании. Из исходных простых геометрических форм можно построить более сложные производные объекты. Существует множество способов формообразования, основанных на комбинации геометрических операндов, но самой естественной техникой такого вида являются булевские операции. Операции сложения, вычитания и пересечения (рис. 3.37) являются обязательными в любой развитой системе геометрического моделирования. Иногда в их состав включают некоторые дополнительные способы синтеза, которые не обладают общностью трех перечисленных.

Если обычные булевские операции применить к объемным телам, то в результате можно получить объект с иной размерностью: поверхность, линию или даже

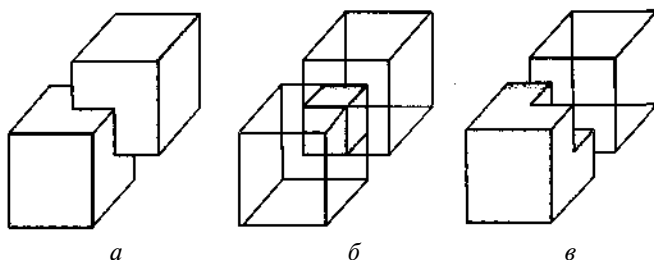


Рис. 3.37. Булевские операции на примере двух кубов:

$$a - A \cup B; \quad b - A \cap B; \quad v - A - B$$

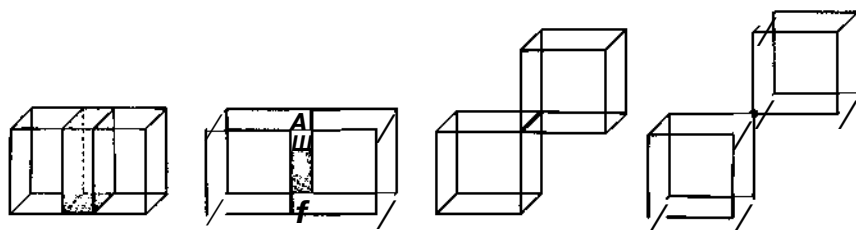


Рис. 3.38. Варианты операции булевского пересечения

точку. Справедливость этого тезиса подтверждает пример, приведенный на рис. 3.38, где показаны различные варианты пересечения двух кубических объемных тел.

Вместо обычных булевских операций будем использовать так называемые регулярные операции, которые обозначим привычными символами, но с добавлением верхнего индекса — звездочки (U^* , Π^* , $-^*$). Регулярные операции, будучи примененными к объемным операндам, всегда дают геометрическое тело, обладающее объемом, например на рис. 3.38 регулярное пересечение дает непустой результат только в первом варианте.

Чтобы глубже исследовать разницу между обычными и регулярными булевскими операциями, рассмотрим объект, определенный как множество точек, которые разделены на внутренние и граничные. Граничными называются точки, имеющие нулевое расстояние до объекта и его дополнения. Множество граничных точек не всегда принадлежит самому объекту. Замкнутые объекты включают в себя границу, а открытые — нет. Объединение множества внутренних точек объекта и его границы обычно называется операцией замыкания. Регуляризацией множества называется замыкание множества внутренних точек. Множество, которое эквивалентно само себе после выполнения операции регуляризации, называется регулярным.

На рис. 3.39 приведен пример операции регуляризации. Исходное состояние объекта представлено на рис. 3.39, *а*. Он определен как множество внутренних точек, показанных серым цветом и множеством граничных точек, изображенных черным цветом. Часть границы, которая не входит в состав объекта, изображена пунктиром. Объект включает в себя несколько висячих фрагментов: две линии и

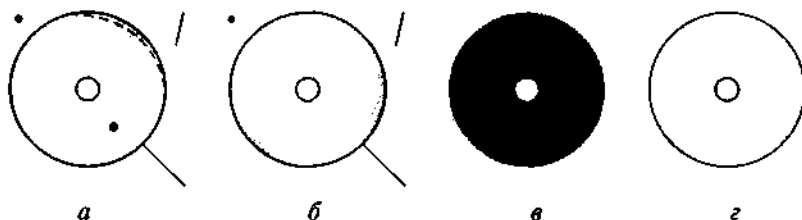


Рис. 3.39. Регуляризация объекта:

а — исходное состояние множества; *б* — множество после операции замыкания;
в — внутренняя область множества; *г* — регуляризованное множество

точку. Черная точка внутри объекта представляет собой границу, которая не принадлежит самой фигуре.

На рис. 3.39, б показано состояние объекта после выполнения операции замыкания. Теперь все граничные точки принадлежат фигуре, внутренний фрагмент границы также входит в ее состав.

На рис. 3.39, в изображены все внутренние точки объекта, иными словами, он представлен как открытое множество точек. Чтобы получить внутренние точки, следует отбросить все висячие и несвязные фрагменты.

На рис. 3.39, г изображен объект после выполнения операции регуляризации.

По определению, регулярное множество не может иметь граничных точек и фрагментов, не обладающих смежностью с некоторыми внутренними точками, поэтому для таких объектов исключается существование висячих граничных фрагментов любого вида (точек, линий или поверхностей).

Регулярные операции можно сформулировать в терминах обычных булевских операций:

$$A \cup^* B = \text{closure}(\text{interior}(A \cup B));$$

$$A \cap^* B = \text{closure}(\text{interior}(A \cap B));$$

$$A -^* B = \text{closure}(\text{interior}(A - B)).$$

Регулярные операции над регулярными операндами всегда дают регулярные множества.

Обычная булевская операция находит пересечение внутренних точек обоих операндов, и, кроме того, включает в состав новой фигуры пересечение границ с внутренними и граничными областями обеих исходных фигур. Регулярное пересечение состоит из пересечения внутренних областей операндов и пересечения внутренностей с границами другой фигуры за исключением пересечения границ.

Рассмотрим более подробно разницу между двумя типами операций на примере объектов, изображенных на рис. 3.40.

На рис. 3.40, а показаны два исходных объекта, которые являются операндами для обычного (рис. 3.40, б) и регулярного (рис. 3.40, в) булевского пересече-

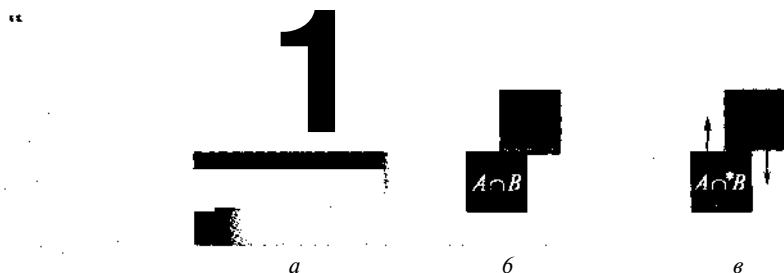


Рис. 3.40. Булевское пересечение:

а — исходное состояние операндов; б — булевское пересечение;
в — регулярное булевское пересечение

чения. Пересечение внутренних областей, а также внутренних и граничных областей всегда входит в исходный результат. На рисунке этот фрагмент выделен темно-серым цветом. Правила включения пересечения границы с границей формулируются несколько сложнее. Если оба операнда лежат по одну сторону от общей границы, то такое пересечение входит в результат. Для нашего примера — это вся граница темного прямоугольника. Если операнды размещены по разные стороны от границы, то пересечение не входит в результат регуляризованной операции. На рис. 3.40, б это кусок линии, обозначенный C .

Таким образом, пересечение границ входит в результат регулярной булевой операции тогда и только тогда, когда внутренние области обоих операндов лежат по одну сторону от общего фрагмента границы. Это интуитивное определение дает четкий критерий в большинстве простых ситуаций, когда операнды принадлежат одной плоскости. Для операндов общего положения требуется выработать более четкий операциональный критерий. Для этого следует ввести понятие нормали к поверхности. Если нормали пересекающихся объектов направлены в одну сторону, то они лежат по одну сторону от общей границы. На рис. 3.40, в нормали черного квадрата и объекта A совпадают, поэтому их общая граница входит в результат операции.

Пусть операнды расположены по разные стороны от общей границы, а их нормали — встречно. В этом случае внутренние области объектов, смежные с общей границей, не входят в пересечение. В результате общая граница не обладает смежностью с внутренними точками результирующей области, поэтому и сама не входит в регулярное пересечение.

Известно, что нормали к поверхностям имеют большое значение для затенения граней поверхности. Определение точной границы булевского пересечения геометрических тел — еще одна важная функция этих объектов.

Результаты регулярных операций можно описать в терминах обычных булевских операций над границами и внутренними областями исходных форм.

Таблица 3.2

Нерегулярные операции	Регулярные операции		
	$A \cup B$	$A \cap B$	$A - B$
$A \cap B$	+	+	-
$A \cup B$	+	-	+
$B \cap A$	+	-	-
$A \cap B$	-	+	-
$B \cap A$	-	+	+
$A \cup B$	+	-	+
$B \cup A$	+	-	-
$\#(A, B)$	+	+	-
$\sim(A \cap B)$	-	-	+

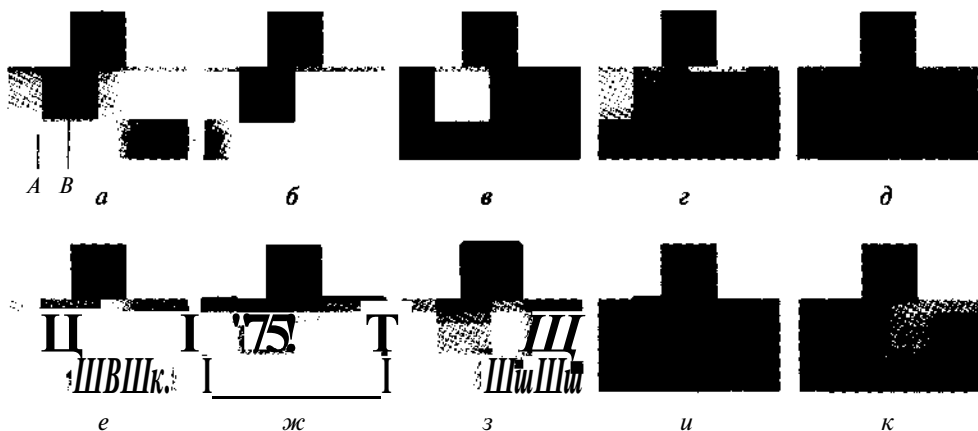


Рис. 3.41. Обычные булевские операции над элементами объектов:

a — операнды A и B ; $б$ — $A \cup B$; $в$ — $A \cap B$; $г$ — $A \setminus B$; $д$ — $B \setminus A$; $е$ — $A \oplus B$; $ж$ — $A \setminus B$; $з$ — $B \setminus A$; $и$ — $\#(A \cap B)$; $к$ — $\sim(A \cap B)$

В табл. 3.2 приведены правила сведения регулярных операций к обычным. На рис. 3.41 на примере простейших форм показаны схемы этих операций. Буквами A и B обозначены операнды, нижние индексы i и j означают соответственно внутренние и граничные области, $\#(A_i \cap B_j)$ — часть общей границы объектов A и B , относительно которой A_i и B_j лежат по одну сторону, наконец, $\sim(A_i \cap B_j)$ означает часть общей границы, где A и B лежат по разные стороны. Каждая регулярная булевская операция сводится к набору обычных операций, которые в табл. 3.2 отмечены знаком «+».

Для всех регулярных операций каждый фрагмент результирующей границы принадлежит либо одному, или обоим операндам. Для операций $A \cap B$ и $A \cup B$ нормали к граням результирующего объекта наследуют свое положение от одного или обоих операндов. Для операции $A \setminus B$ действует иное правило. Нормали граней результирующего тела, которое получено вычитанием B из A , имеют противоположное направление нормалям соответствующих граней B . Это следует из тождества $A \setminus B = A \cap \bar{B}$, где \bar{B} — дополнение B . Тело \bar{B} может быть получено дополнением внутренней части тела B с последующим обращением нормалей границы.

Во многих графических редакторах и системах автоматизированного проектирования пользовательский интерфейс основывается на использовании регулярных булевских операций. Эта техника предоставляет оператору удобный и надежный способ формирования сложных объектов из нескольких простых.

3.4.3. Параметрическое моделирование геометрии

Во многих отраслях техники сложный объект или система рассматривается как сочетание некоторого набора стандартных элементов. Очевидные преимущества

этого подхода к проектированию сделали привлекательной идею о такой организации процесса геометрического моделирования, когда облик изделия создается из стандартных форм — элементов некоторого геометрического конструктора. Эти элементы принято называть геометрическими примитивами. Примитивы, у которых зафиксированы все геометрические и конструктивные параметры, имеют ограниченную область применения, поэтому в геометрическом проектировании используются параметризованные примитивы. Например, примитивом будет многоугольник, для которого пользователь может задавать не только геометрические размеры, но и число сторон. Примитивом может быть элемент некоторого семейства объектов, члены которого различаются несколькими конструктивными, технологическими и геометрическими параметрами. Такой подход, названный групповой технологией, используется в системах автоматизированного проектирования.

Параметризация примитивов часто применяется для определения таких сравнительно сложных объектов, как болты, гайки, зубчатые колеса и т. п. Конечно, эти формы допускают определение в терминах регулярных булевских операций, однако данный подход требует кропотливой работы над элементарными геометрическими телами, которые являются операндами сложных тел.

Примером сложного параметризованного примитива является зубчатое колесо, которое описывается набором из четырех геометрических и конструктивных параметров (рис. 3.42).

Привлекательная идея параметризации примитивов имеет несколько существенных ограничений. В этой парадигме не существует никаких средств для синтеза сложных объектов из набора простых. Для создания нового примитива требуется разрабатывать его код заново. Каждый примитив должен иметь программное окружение, состоящее из набора сервисных процедур и утилит. Например, в нее могут входить программы для расчета массы, объема, момента инерции и других конструктивных характеристик объекта. Как правило, утилиты такого типа являются уникальными: они создаются под спецификацию конкретного примитива и в общем случае не способны рассчитывать другие элементы базиса.

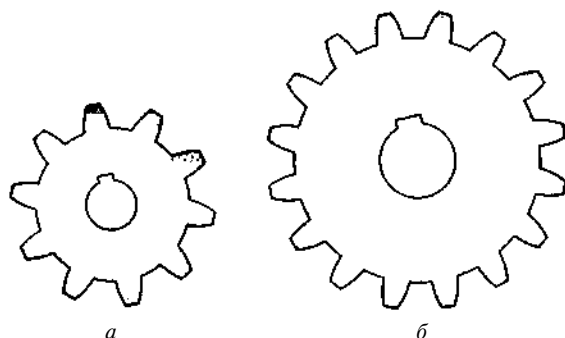


Рис. 3. 42. Параметризованное зубчатое колесо:

a — диаметр — 40, толщина — 10, отверстие — 10, число зубьев — 10;
б — диаметр — 60, толщина — 20, отверстие — 15, число зубьев — 16

3.4.4. Заметание

Заметанием (sweeping) называется способ формообразования при помощи движения объекта по некоторой траектории или согласно некоторому заданному закону. Это очень продуктивный метод, позволяющий получать трехмерные тела, которые с трудом воспроизводятся с помощью иных способов геометрического синтеза. Простейший пример этого метода генерации форм дает движение замкнутой двухмерной кривой вдоль прямолинейной траектории. Во многих пакетах машинной графики данный метод формообразования называют экструдированием (extrude). Экструдирование позволяет получить множество трехмерных форм, описывающих машиностроительные детали, элементы архитектурных конструкций, предметы интерьера и т. п.

Возможности формообразования значительно расширятся, если использовать траектории в виде кривых произвольной формы. В этом случае движение сечения ограничивается дополнительным условием, согласно которому в каждой точке пути нормаль сечения должна совпадать с касательной к траектории.

Вращение сечения вокруг некоторой фиксированной оси — еще один результативный способ генерации трехмерных форм, потенциально способный породить множество всех объектов, которые в инженерной практике именуются телами вращения. В англоязычной литературе этот способ называется rotational sweeping, что обычно переводят как заметание вращением.

Таким образом, если выбрать образующую, направляющую и закон движения, то множество точек пространства, которое заметет первый объект при смещении по второму, полностью определяет новое, в общем случае трехмерное тело. На рис. 3.43

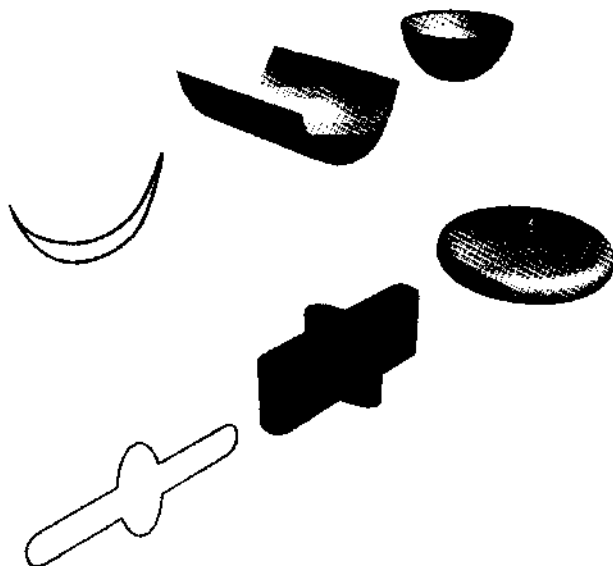


Рис. 3.43. Примеры кривых и трехмерных форм, порожденных их движением в пространстве

показаны примеры кривых и трехмерных тел, которые они порождают своим движением вдоль различных траекторий.

Чтобы получить трехмерную форму методом заметания, необязательно смещать двухмерное сечение, в некоторых случаях намного удобнее в качестве образующей использовать некоторый трехмерный объект. Этот метод формообразования используется при программировании металлообрабатывающих систем с числовым программным управлением и в системах управления роботами и робототехническими комплексами.

Следует отметить, что рассмотренный метод формообразования может иметь различные наименования в системах геометрического моделирования и литературе по КГ. Например, в популярной системе трехмерной графики 3Ds Max заметание называется Loft, в описаниях пакета на русском языке его именуют лофтингом. Заметание вращением (rotational sweeping) называется Lathe. Существуют и другие терминологические варианты, получившие распространение в кругах пользователей определенных пакетов трехмерной графики или систем автоматизированного проектирования.

Заметание — естественный способ генерации новых трехмерных форм. Он допускает простое управление и дает предсказуемые результаты. Однако его изобразительные возможности не безграничны, существуют трехмерные объекты, которые нельзя создать движением образующих. Заметание требует значительных вычислительных ресурсов для расчета трехмерных объектов, полученных движением по самопересекающимся траекториям. Кроме того, класс объектов, полученных этим способом, не является замкнутым относительно регуляризованных булевских операций.

На рис. 3.44 представлены две призмы, полученные смещением треугольников вдоль прямой траектории. Булевское объединение этих форм не может быть создано заметанием каких-либо двухмерных или трехмерных образующих.

Формообразование можно существенно расширить, если допустить использование нескольких образующих, связанных с одной траекторией. На рис. 3.45

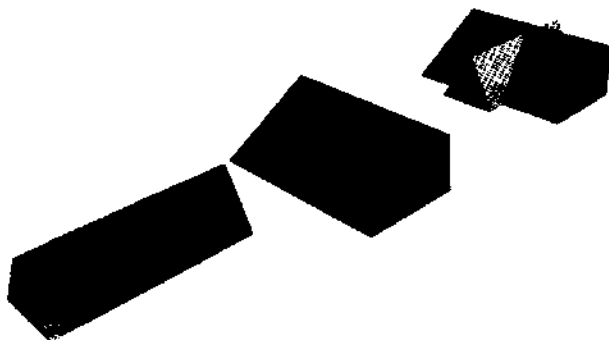
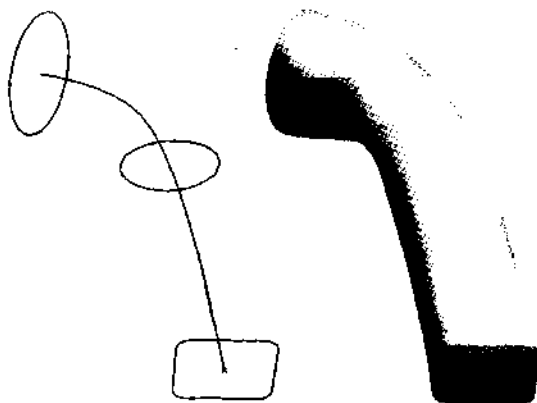


Рис. 3.44. Объединение двух трехмерных объектов, созданных экстрадированием треугольников, не может быть получено заметанием



%*

Рис. 3.45. Заметание с тремя образующими

показан пример трехмерного тела, полученного движением трех образующих вдоль одной кривой.

В некоторых пакетах машинной графики этот способ формообразования считается самостоятельным и называется скиннингом (skinning).

3.4.5. Граничное представление

Граничным представлением (Boundary Representation, B-rep) называется описание пространственных тел в терминах границ и их элементов. Это один из самых простых, выразительных и хорошо изученных способов определения трехмерной геометрии.

В описании границы тела участвуют объекты с интуитивно понятным геометрическим содержанием: вершины, ребра, грани и оболочки. Рассмотрим подробнее топологические свойства граничных составляющих.

Оболочки тела должны обладать свойством однородности — это значит, что все они должны быть описаны по единым правилам. Так, согласно этому принципу, не разрешается смешивать в одном описании явные и неявные модели граничных оболочек. Оболочки могут состоять из нескольких граней. Каждая грань представляет собой фрагмент некоторой поверхности с данными о связях с соседними гранями и ориентации относительно внутреннего объема тела.

Ограничивающие оболочки делят пространство на две части: внутреннюю и внешнюю. Невозможно перейти из одной части пространства в другую без пересечения границы. Существуют объекты, для описания которых требуется несколько граничных оболочек. Примерами таких тел являются отливки с кавернами, машиностроительные детали, обладающие отверстиями, резервуары и пр. Геометрия всех названных примеров описывается, по крайней мере двумя оболочками: внешней и внутренней. Внутренние оболочки определяют пустоты и

располагаются внутри внешней. Оболочки любого типа не должны иметь пересечений друг с другом и самопересечений.

Во многих операциях автоматизированного конструирования и технологической подготовки производства важно различать внутренние и внешние точки деталей и узлов. Поэтому граничные оболочки считаются ориентированными, одна сторона их обращена внутрь тела, другая — наружу. Чтобы сделать ориентацию определенной каждой точке границы приписывается нормаль, которая направлена от оболочки в наружном направлении. Это соглашение представляется совершенно ясным для внешних оболочек, некоторые противоречия с геометрической интуицией могут возникать с нормальными внутренними фрагментами границы, которые имеют нормали, направленные внутрь ограничиваемой части пространства. Можно считать, что внутренние оболочки тела ведут себя как вывернутые наизнанку внешние.

Таким образом, для точного математического описания трехмерного тела требуется задать внешнюю оболочку и в зависимости от связности объекта одну или несколько внутренних оболочек.

Большая часть современных систем геометрического моделирования поддерживает представление, у которого фрагментами границ являются так называемые двумерные многообразия — геометрические объекты, каждая точка которых имеет окрестность, топологически подобную плоскому двумерному диску. Точное название отношения топологического подобия — гомоморфизм. Это абстрактная математическая категория, которая нуждается в четком и развернутом определении. С небольшой погрешностью подобными можно считать две геометрические фигуры, между точками которых можно установить непрерывное взаимно-однозначное соответствие.

На рис. 3.46 изображены образцы тел. Две призмы, показанные на рис. 3.46, б, соединены по одному ребру. Если взять любую точку на этом ребре и проанализировать ее окрестность, то можно заметить ее коренное топологическое отличие от двумерного диска. Примеры объектов, не относящихся к классу многообразий: соединение трехмерного тела с отдельным ребром; два многообразия, имеющие только одну общую вершину; трехмерное тело с внутренними переборками.



Рис. 3.46. Форма многообразного тела (а) и не многообразного тела (б)

Полиэдры и формула Эйлера

В КГ полиэдром называется твердое тело, границу которого образует множество многоугольников, расположенных таким образом, что любая пара из них либо не пересекается, либо пересечение этих фигур является ребром каждого многоугольника. Простым полиэдром называется тело, топологически эквивалентное обычной сфере. Если не использовать точные определения из гомотопической топологии, то можно считать, что такая эквивалентность означает потенциальную возможность одну фигуру перевести в другую посредством непрерывной деформации. Очевидно, что несмотря на некоторое родовое и видовое сходство, сфера и тор не являются топологически подобными фигурами, поскольку внутреннее отверстие тора принципиально отличает его от сферы.

Граничное представление любых простых полиэдров должно удовлетворять формуле Эйлера, которая связывает число вершин, ребер и граней:

$$V - E + F = 2, \quad (3.38)$$

где V — число вершин; E — число ребер; F — число граней.

На рис. 3.47 показаны примеры простых полиэдров и приведены значения множества вершин, ребер и граней. Очевидно, что для всех этих примеров верна формула Эйлера. Следует отметить, что формула Эйлера справедлива и для объектов, границу которых образуют неплоские грани и криволинейные ребра. Для простых полиэдров формула Эйлера является необходимым, но не достаточным условием. Если для некоторого трехмерного тела выполняется это соотношение, то отсюда не следует, что оно представляет собой простой полиэдр. Можно привести примеры объектов, которые не являются даже многообразиями, но удовлетворяют соотношению Эйлера.

Двухмерные многообразия, имеющие грани с отверстиями, описываются обобщенной формулой Эйлера:

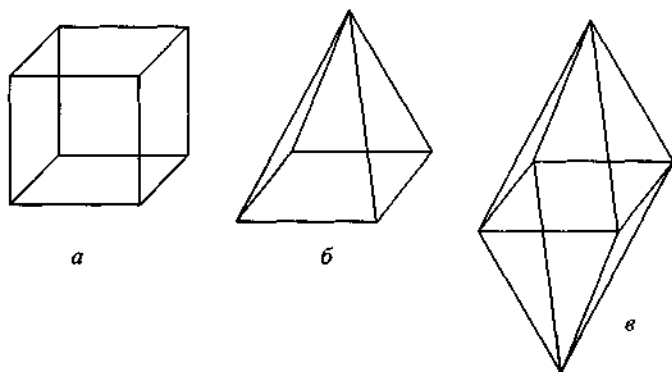


Рис. 3.47. Примеры простых полиэдров:

a — $V=8$, $E=12$, $F=6$; $б$ — $V=5$, $E=8$, $F=5$; $в$ — $V=6$, $E=12$, $F=8$

iii HS' - .tf&UIW

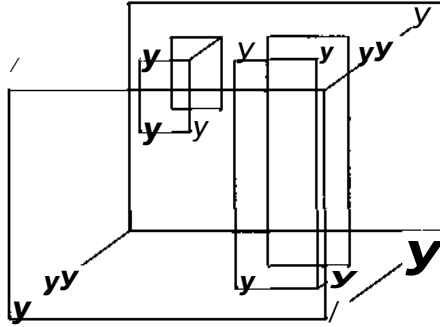


Рис. 3.48. Пример многообразия с отверстиями, удовлетворяющего обобщенной формуле Эйлера:

$$V=24; E=36; F=15; Я=3; C=1; G=1$$

(3.39)

$$V-E+F-H=2(C-G),$$

где Я — общее число отверстий на гранях; G — число отверстий, проходящих через объект (род многообразия); C — число связанных компонентов, образующих многообразие.

На рис. 3.48 приведен пример фигуры с отверстиями, которая полностью удовлетворяет обобщенной формуле Эйлера.

Очевидно, что любые операции над телами, которые не меняют числа вершин, ребер и граней, сохраняют соотношение Эйлера. К таким операциям относятся все аффинные преобразования геометрии трехмерных тел. В теоретических исследованиях по КГ обсуждаются так называемые эйлеровы операции, которые на основе нескольких трехмерных операндов синтезируют новые геометрические формы. Формула Эйлера является инвариантом в таких преобразованиях.

Структура данных граничного представления

Граничные оболочки трехмерных тел представляют собой множества граней. Грани — фрагменты поверхностей, математическое описание которых входит в общую структуру данных. Важно различать направление сторон граней. Одна сторона грани направлена наружу граничной оболочки, другая — внутрь. Описание грани в структуре данных должно содержать признак, по значению которого можно определить направление. Будем считать, что признак ориентации принимает положительное значение, если нормаль грани направлена наружу, отрицательное значение — в противном случае.

Пересечение граней является ребром. Ребро как объект структуры данных строится на основе линии пересечения поверхностей, которые являются носителями граней. Ребро может иметь ориентацию, совпадающую с направлением линии или противоположную ей. Для дифференциации таких случаев в структуру данных вводится специальный флаг, который принимает поло-

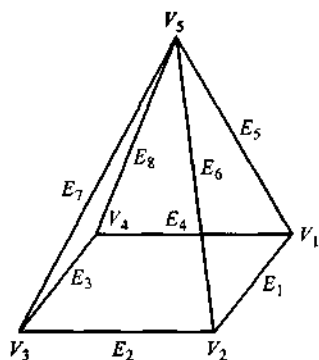


Рис. 3.49. Простой полиэдр, заданный описанием границ. Списки в правой части рисунка означают грани фигуры:

V — вершины; E — ребра;
 F — грани; $F_1 = (V_5, V_2, V_3)$;
 $F_2 = (V_2, V_3, V_1)$; $F_3 = (V_3, V_4,$
 $V_5)$; $F_4 = (V_1, V_4, V_3)$;
 $F_5 = (V_1, V_2, V_3, V_4)$

жительное значение при совпадении направлений и отрицательное — в случае их различия.

Граничные ребра каждой грани образуют новый топологический объект, называемый циклом. В структуре данных В-гер цикл — замкнутое образование, имеющее ориентацию. Если направление ребра совпадает с циклом, то ему приписывается положительный флаг, в противном случае — отрицательный. Таким образом, циклом называется отсортированное по порядку следования множество ребер с флагами.

Если грань имеет отверстия, то ее граница будет описываться несколькими циклами: одним внешним и несколькими (в общем случае) внутренними. Понятно, что внутренние циклы должны целиком лежать внутри внешнего. Будем считать, что внешний цикл ориентирован против часовой стрелки, если смотреть на грань с внешней стороны, и, наоборот, внутренние циклы ориентированы по часовой стрелки при сохранении той же позиции наблюдателя. Согласно этой договоренности, грань

всегда остается с левой стороны, если двигаться по циклам с сохранением их ориентации.

Ребро, которое является границей двух граней, входит в два цикла: в одном цикле направление ребра совпадает с направлением цикла, а в другом — противоположно направлению цикла.

Каждое ребро имеет две граничные вершины. С точки зрения структуры данных, объект «вершина» должен хранить информацию о координатах точки пространства и сведения о всех инцидентных ребрах.

Простейшей формой хранения данных В-гер является табличная структура данных. Рассмотрим возможный вариант организации такой структуры на примере тела, показанного на рис. 3.49.

В табл. 3.3 представлены данные об элементах граничной оболочки полиэдра, показанного на рис. 3.49. В каждой строке подтаблицы записаны ребра, которые образуют граничный цикл соответствующей грани. Похожим образом организована и подтаблица вершин. В ней представлены граничные вершины ребер и числовые значения координат всех вершин. Для точного определения этих значений требуется выбрать некоторую систему координат, связанную с данным телом. Если из этой структуры исключить подтаблицу граней, то получится таблица, которая полностью задает каркасную модель трехмерного тела.

Несмотря на простоту, структуры данных подобного вида практически не применяются в современных системах автоматизированного проектирования и геометрического моделирования. Основными причинами этого являются:

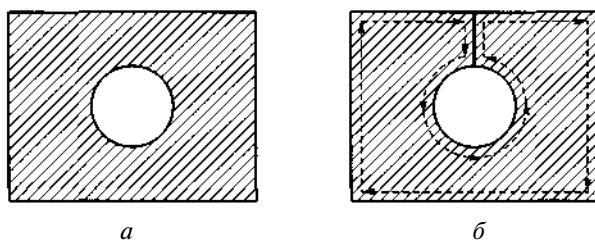


Рис. 3.50. Описание нескольких границ одним списком:

a — многосвязная грань; *б* — грань с мостом

• данная структура ориентирована на хранение плоских многогранников. Она не рассчитана на описание объектов с криволинейными гранями и ребрами;

• табличная структура в описанном виде не подходит для представления многосвязных граней. Трехмерные тела со сквозными отверстиями имеют грани, границы которых состоят из нескольких циклов. Существует простой технический прием, который позволяет распространить возможности этой структуры данных и на случай многосвязных границ. Для этого достаточно добавить одно ребро, соединяющее внутреннюю и внешнюю границы, как показано на рис. 3.50. Соединительное ребро принято называть мостом, или перемычкой;

• табличная форма записи затрудняет генерацию данных о связности объектов. Достаточно рассмотреть пример, когда требуется найти две грани с общим ребром. Для этого в общем случае придется просмотреть всю таблицу граней и найти строки, в которых присутствует одно ребро. Это подход в случае больших таблиц требует очень высоких вычислительных затрат.

Таблица 3.3

Подтаблица граней	Подтаблица ребер	Подтаблица вершин			
		Ребро	Вершина	Вершина	Координаты
Λ_1	$E_6 E_5, E_6$	E_1	$V_1 V_2$	V_1	M, Y, Z
F_2	E_2, E_4, E_5	E_2	Y_6, Y_3	Y_2	X_6, Y_6, Z_1
F_3	E_3, E_4, E_5	E_3	Y_3, Y_4	Y_3	X_3, Y_3, Z_3
F_4	E_4, E_5, E_6	E_4	V_4, V_1	V_4	X_4, Y_4, Z_4
F_5	E_5, E_2, E_3, E_4	E_5	V_5, V_6	Y_6	X_6, Y_6, Z_6
—	—	E_6	Y_2, Y_6	—	—
—	—	E_7	Y_3, Y_6	—	—
—	—	E_8	Y_4, Y_6	—	—

Существуют две простые структуры данных, лишенные указанных недостатков табличного описания — структура полуребер (half-edge data structure) и структура крыльевых ребер (winged-edge data structure).

Структура полуребер

Специалистам в области информатики и практикующим программистам хорошо известны все основные структуры данных и их сильные и слабые стороны. Так, очевидные недостатки массивов, которые служат простейшей формой описания таблиц, преодолеваются применением многосвязных списочных структур. Для хранения информации о граничном представлении трехмерных тел можно использовать такой способ организации данных, у которого основной единицей хранения будет двухсвязный список ребер грани.

Более того, для каждой грани создается указатель на первое ребро списка, а в описании ребра создаются указатели на предыдущее и последующие ребра. Пример такой организации данных в виде двухсвязного списка ребер грани F_1 показан на рис. 3.51.

Двухсвязный список позволяет использовать простой способ восстановления всей границы по любому предъявленному ребру. Для этого достаточно пройти весь граничный цикл, двигаясь по указателям. Если принять эту форму хранения для всех граней в виде двухсвязного списка, то появятся проблемы, связанные с целостностью представления всей оболочки. Пусть грань F_2 представлена списком со входом в E_2 (рис. 3.52). Очевидно, что упорядоченность ребер этого списка противоречит системе указателей списка, задающего грань F_1 . Общее для граней F_1 и F_2 ребро E_6 (см. рис. 3.49) получает различную ориентацию в списках обеих граней. Для разрешения этого противоречия можно разделить каждое ребро пополам и использовать половинки в описаниях тех граней, к которым они относятся. Частям одного ребра приписывается противоположная ориентация, а описание граней представляет собой двухсвязный список полуребер. Элементы списочного описания грани связываются ссылками таким образом, что направление обхода каждого из них согласуется с направлением обхода грани (против часовой стрелки, если смотреть снаружи тела).

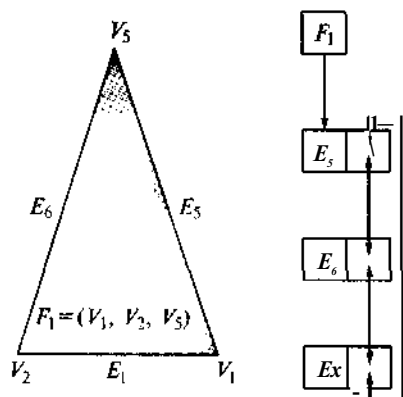


Рис. 3.51. Двухсвязный список ребер грани F_1

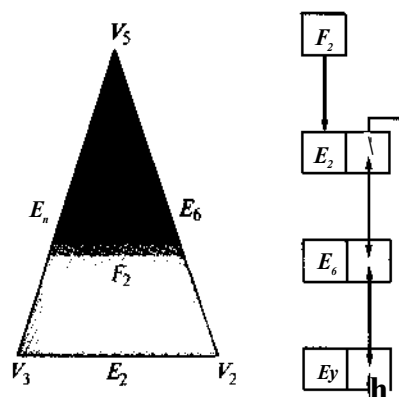


Рис. 3.52. Двухсвязный список ребер грани F_2

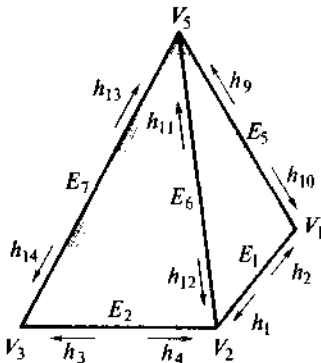
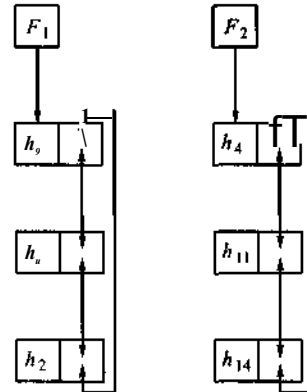


Рис. 3.53. Полуребра простого полиэдра


Рис. 3.54. Двухсвязные списки полуребер граней F_1 и F_2

Пример разбиения ребер на полуребра показан на рис. 3.53. Здесь представлены только две грани простого полиэдра, полное изображение которого дано рис. 3.49. На рис. 3.54 изображены две списочные структуры, которые задают границы граней F_x и F_y .

Структура полуребер позволяет компактно описывать грани с отверстиями без дополнительных ребер-мостиков. В общем случае грань может иметь сложную границу, состоящую из нескольких внутренних фрагментов, задающих отверстия, и одного внешнего граничного контура. Последовательность ребер, определяющая любой граничный фрагмент, представляет собой цикл, или петлю. Для описания грани с отверстиями можно организовать список списков, в котором входами служат циклы, а каждый цикл задается двухсвязным списком полуребер.

На рис. 3.55 приведен пример трехсвязной грани с разметкой полуребер, а на рис. 3.56 изображена списочная структура этого объекта. Из рис. 3.53 и 3.54 следует, что при этом способе организации данных грань ссылается на список полуребер косвенно, через двухсвязный список циклов. Входом этого списка служит обычно внешний цикл грани.

Последовательность перечисления полуребер в списках зависит от принятой ориентации циклов. Считается, что ребра внешнего цикла ориентированы против часовой стрелки, а ребра всех внутренних — по часовой, если смотреть на грань с внешней стороны тела (см. рис. 3.55). Это соглашение об ориентации сохраняет внутреннюю часть грани по одну сторону от границы. Внутренние точки грани всегда лежат по левую сторону при обходе циклов в выбранном направлении.

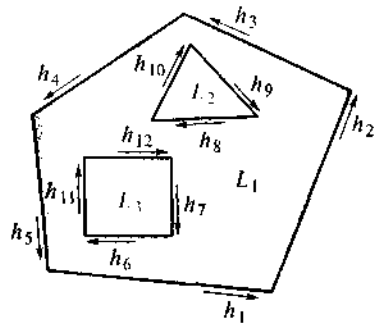


Рис. 3.55. Пример многосвязной грани

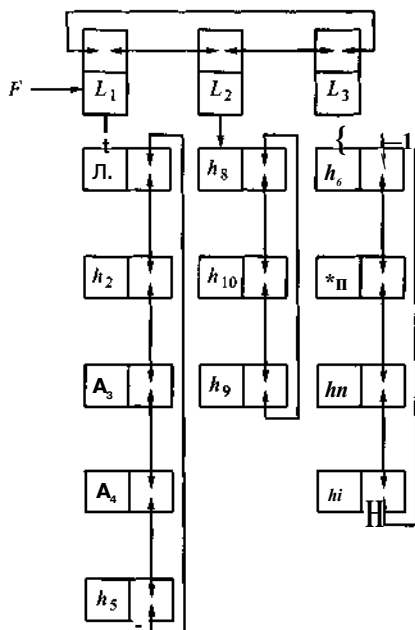


Рис. 3.56. Списочная структура грани с отверстием

Чтобы придать списочной структуре данных свойства полноценной модели, следует пополнить ее информацией о связях полуребер с инцидентными вершинами и родительскими ребрами. Эта задача решается созданием дополнительных ссылок.

Чтобы создать связи между ребрами и полуребрами, вводятся указатели ребер на порожденные полуребра и указатели полуребер на родительские ребра. Аналогичным образом создаются связи между вершинами и полуребрами.

Описанная структура данных имеет значительные преимущества по сравнению с табличным представлением. Отметим только самое главное. Эта форма описания позволяет сохранить информацию о связях вершин, ребер и граней и по этим данным синтезировать любые необходимые сведения о смежности.

Рассмотрим способ определения множества ребер, исходящих из вершины V_i , на примере простого многогранника, показанного на рис. 3.57.

Если вершина V_i является начальной вершиной (это справедливо для приведенного примера), то выбирается полуребро $prev / \Gamma_b$ предшествующее A_j . Его родительское ребро представляет собой один из искомым объектов, соединенных с V_i . После этой операции вместо ребра $h \setminus$ рассматривается сопряженное ему полуребро (обозначим его $new h \setminus$), и первый шаг алгоритма повторяется заново. Если V_i является конечной вершиной для $/ \Gamma_b$ то выбирается полуребро, следующее за $/ \Gamma_b$ а его родитель включается в число смежных ребер. Далее вместо $h \setminus$ рассматривается полуребро, сопряженное со следующим за $/ \Gamma_b$ и первая операция повторяется с полуребром $new h \setminus$. Процедура повторяется до тех пор, пока не будет достигнуто полуребро, сопряженное со стартовым $h \setminus$.

Структура крыльевых ребер

Рассмотренная в подразд. 3.3 структура полуребер представляет собой список граней, каждая из которых задана двухсвязным списком ребер. По сравнению с табличной формой хранения данных это намного более эффективное представление оболочек твердых тел. В твердотельной вычислительной геометрии существуют задачи, вычисление которых в

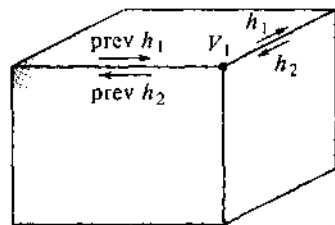


Рис. 3.57. Генерация данных о смежности ребер и вершин

структуре полуребер данных требует значительных вычислительных ресурсов. Известно несколько более эффективных подходов к описанию оболочек. Самым распространенным из них является подход Бомгарта (Baumgart), основанный на применении так называемых крыльевых ребер (winged edges). Сердцевиной этой структуры данных служит описание ребер, а не граней, как это принято в структуре полуребер. Каждое ребро представляется системой ссылок на связанные с данным ребром объекты многоугольника. В эту систему входят ссылки на две граничные вершины, на две грани, пересечение которых образует данное ребро, и четыре ребра, исходящих из граничных вершин.

Фрагмент многоугольника на рис. 3.58 иллюстрирует структуру данных крыльевых ребер. На рисунке в виде стрелок показаны все восемь ссылок ребра E_1 .

Чтобы обеспечить более эффективное решение наиболее востребованных топологических задач, структура данных пополняется дополнительными ссылками:

- каждая вершина получает обратную ссылку на одно из ребер, исходящих из нее;
- каждая грань снабжается ссылкой на одно из граничных ребер.

При любом направлении обхода соседних граней (например, по часовой стрелке) вершины, принадлежащие общей грани, будут упорядочены в последовательности пройденных вершин в противоположном порядке. Пусть граничными вершинами некоторого ребра, общего для двух граней, являются вершины n и p . Грань называют p -гранью, если при обходе ее границы по часовой стрелке вершина p встречается в перечислении вершин позже (стоит правее) вершины n . Грань называется i -гранью в противном случае.

Пусть граничная вершина V_i ребра E_1 (см. рис. 3.58) есть n , а вершина V_2 — p . Тогда, согласно введенной классификации, для ребра E_1 грань F_1 есть p -грань, а грань F_2 — i -грань. Используя заданную классификацию, можно ввести четкое различие четырех смежных ребер (E_2, E_3, E_4, E_5), на которые ссылается основное ребро (E_1). Два ребра, инцидентные вершине n , назовем n -ребрами. В нашем случае (см. рис. 3.58) это ребра E_2, E_3 . При обходе граней по часовой стрелке они являются последующими для i -грани и предыдущими для p -грани. Ребра, связанные с вершиной p , будем называть p -ребрами. В нашем случае это E_4, E_5 . По сравнению с i -вершинами они демонстрируют прямо противоположные свойства. При фиксированном направлении обхода они являются последующими для p -грани и предыдущими для i -грани. Эти четыре ребра называют крыльями (wings), от них и пошло название всей структуры данных.

Описанная структура данных предназначена для хранения данных об оболочках, состоящих из односвязных граней. Существует несколько модификаций этой

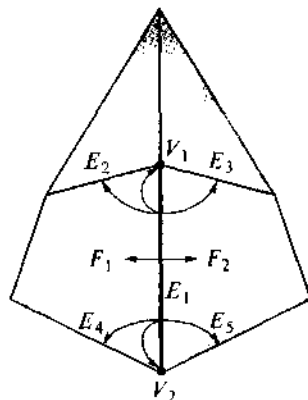


Рис. 3.58. Фрагмент многоугольника и ссылки структуры крыльевых ребер

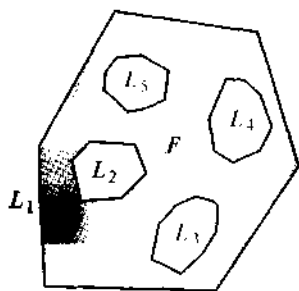


Рис. 3.59. Грань с несколькими отверстиями

структуры, позволяющих расширить ее возможности на грани с отверстиями. Например, можно хранить описание каждой грани в виде списка граничных циклов подобно тому, как это делается в структуре полуребер. Для каждой грани сохраняется указатель на внешний цикл, который, в свою очередь, ссылается на внутренний граничный цикл, если он имеется. Последний указывает на другой цикл при наличии нескольких отверстий в одной грани или на внешний цикл, в ином случае.

Пример такого представления иллюстрируют рис. 3.59 и 3.60. На рис. 3.59 показана грань с отверстиями, а на рис. 3.60 представлена списочная структура, которая задает топологию этой грани.

Выбор односвязного списка вместо двухсвязного не имеет принципиального значения. Односвязный список позволяет получить более экономную структуру, что достигается за счет некоторой потери эффективности расчета для некоторых классов задач.

Булевские операции в системе В-гер

Синтез новых тел из набора исходных объектов путем применения некоторого множества выбранных операций — мощный и естественный способ формообразования. В компьютерной геометрии известно несколько классов допустимых операций, применимых к трехмерным телам. Одними из самых распространенных являются так называемые булевские (булевы) операции. К их числу прежде всего относятся объединение, вычитание и пересечение двух тел. Можно утверждать, что не существует развитых систем геометрического моделирования, которые не поддерживают этот метод синтеза трехмерных и двумерных форм.

В общем случае результатом булевских операций могут быть не многообразные объекты, работа с которыми ограниченно поддерживается в современных системах машинной графики. Напомним, что i -мерным многообразием называется тело, расположенное в i -мерном пространстве, каждая точка которого (тела) имеет окрестность, топологически подобную « i -мерной сфере».

Существует условие, которое немного ограничивает разнообразие синтезируемых форм, но гарантирует их высокую «технологичность». Это дополнительное условие называется регуляризацией, его формулировка и основные свойства подробно рассмотрены в начале главы. Способ

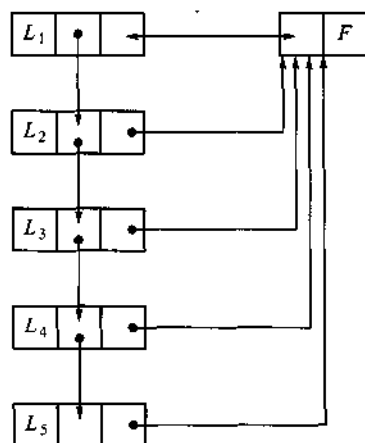


Рис. 3.60. Списочная структура, задающая грань с отверстиями

описания трехмерных тел также существенно ограничивает применение булевских операций. Существуют несовместимые пары булевских операций и видов описания геометрии.

В табл. 3.4 дан полный список регулярных операций, применимых к телам с граничным описанием, и сведение регулярных операций к обычным (полный вариант этой таблицы приведен в начале главы).

Таблица 3.4

Булевские операции	Регулярные булевские операции		
	$A \cup^* B$	$A \cap^* B$	$A -^* B$
$A_i \cap B_i$	-	+	-
$B_b \cap A_b$	-	+	+
$A_i - B_i$	+	-	+
$B_b - A_b$	+	-	-
$\#(A_i \cap B_i)$	+	+	-
$\sim(A_b \cap B_b)$	-	-	+

Примечание. Большими латинскими буквами обозначены трехмерные тела-операнды. Нижние индексы i и b означают соответственно внутренние и граничные области: $\#(A_i \cap B_b)$ — часть общей границы объектов A и B , относительно которой A_i и B_b лежат по одну сторону; $\sim(A_b \cap B_b)$ — часть общей границы, где A и B лежат по разные стороны; «+» — набор обычных операций, к которому сводится каждая регулярная булевская операция.

Результатом регулярного объединения двух тел $A \cup^* B$ является новое тело, содержащее точки, принадлежащие внутреннему объему первого или второго операнда. Результатом регулярного пересечения $A \cap^* B$ служит тело, содержащие точки, принадлежащие внутреннему объему первого и второго операндов. Тело, образованное вычитанием $A -^* B$, состоит из внутренних точек первого операнда, которые не входят во внутренний объем второго операнда.

Известно, что операцию вычитания можно свести к операции пересечения; для этого требуется обратить второй операнд (вывернуть его наизнанку) и найти множество точек, принадлежащее объему первого и второго операндов. Операцию обращения обозначают обычно через $\bar{\cdot}$. В обращенном теле внутренние грани становятся наружными, наружные — внутренними, кроме того, изменяется направление всех граничных циклов на противоположное. В результате внутренним объемом становится та часть пространства, которая первоначально располагалась за граничной оболочкой тела. Сведение операций можно записать в виде короткого математического выражения $\Gamma = \bar{\Gamma} - B = A \cap \bar{B}$.

Булевские операции над двухмерными и трехмерными объектами имеют множество общих черт. Это относится к их математической и программной реализации в пакетах КГ.

Техника булевого объединения

Рассмотрим операцию булевого сложения двух трехмерных тел. Суть операции достаточно проста. Требуется найти линии пересечения граничных оболочек двух тел, затем удалить часть первого тела, которая расположена внутри второго, и кусок второго тела, попавшего внутрь первого. Оставшиеся фрагменты образуют новое тело, которое представляет собой булевоое объединение.

Операцию можно разбить на три этапа. Первый этап состоит в получении линий пересечения граничных оболочек двух тел. По этим линиям строят новые ребра, которые будем называть ребрами пересечения. На втором этапе ищут точки пересечения новых ребер со старыми ребрами, присутствовавшими в описании границ операндов до операции. Эти ребра разрежем на части по всем новым вершинам. Третий этап заключается в перестройке граничных циклов пересекающихся граней. Рассмотрим содержание этапов более подробно.

Первый этап операции булевого объединения начинается с поиска линий пересечения каждой грани первого тела с каждой гранью второго тела, если таковые (линии) имеются. Технически этот шаг реализуется при помощи хорошо известного алгоритма поиска линии пересечения двух поверхностей, который рассмотрен в гл. 4, посвященной алгоритмическому обеспечению КГ.

Пусть построены искомые линии пересечения. На их базе создадим ребра пересечения. Всем новым ребрам припишем направление, которое совпадает с ориентацией векторного произведения нормали грани первого тела с нормалью грани второго тела. Будем считать положительной ориентацию нормали, направленной наружу тела (рис. 3.61). Ребра пересечения должны целиком лежать внутри граничных циклов исходных тел. Они могут соединяться со старой границей только своими концевыми вершинами. Это условие, примененное к примеру, показанному на рис. 3.61, требует разделения ребер старой границы на две части в точках A , B и C .

На втором этапе процедуры построения булевого объединения требуется разрезать ребра старой границы в точках касания новых ребер. Это разделение

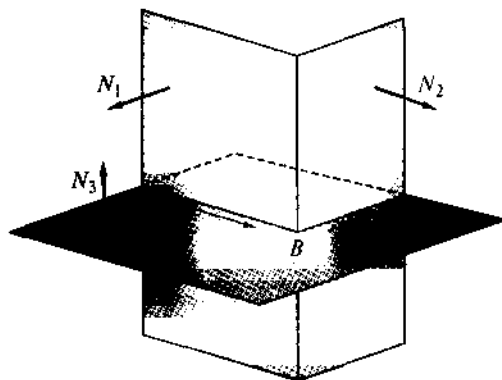


Рис. 3.61. Нормали граней и направления ребер пересечения

выполняется посредством рассечения кривых, которые являются носителями ребер. Из одной кривой получим две кривые, которые при соединении образуют исходную кривую. Одна из этих кривых останется геометрическим носителем разрезаемого ребра, а на базе второй построим новое ребро, которое наследует свойства исходного ребра. Граничные ребра строятся на основе кривой пересечения двух поверхностей. Рассечению подлежат два экземпляра этой двумерной кривой, лежащие на разных поверхностях.

Точки пересечения нового ребра со старым ребром грани ищем как точки пересечения двумерных кривых, заданных на общей для них плоскости параметров. От каждого ребра в формуле, определяющей точки пересечения линий, участвует по одной двумерной кривой, входящей в линию пересечения. Параметры точек пересечения ребер и сами координаты кривых, являющиеся параметрами поверхности, должны быть определены с заданной точностью. Поскольку каждое ребро исходных тел входит в циклы двух смежных граней, то после резки ребер исходных тел необходимо произвести корректировку этих циклов с учетом проведенного разрезания.

В результате выполнения первых двух этапов получим совокупность ориентированных ребер пересечения, которые соединяются друг с другом и со старыми ребрами тел-операндов только в вершинах.

Третий этап является заключительным в операции построения булевого объединения. Его цель состоит в разрезании граней и перестройке граничных циклов в соответствии с новой структурой граничных ребер.

На рис. 3.62 показаны грани двух пересекающихся тел, представленных на рис. 3.61. Тонкими линиями со стрелками изображены направления граничных циклов исходных граней, толстыми линиями со стрелками — ребра разрезания. Требуется перестроить граничные циклы таким образом, чтобы учесть новую ситуацию, созданную разрезанием. На рис. 3.63 показаны части граней, которые войдут в результирующее объединение. Как и ранее, тонкие линии со стрелками

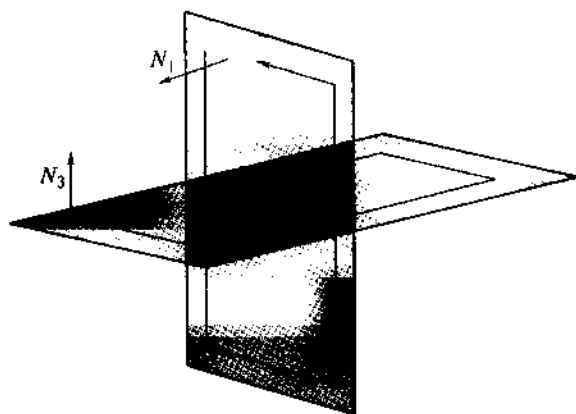


Рис. 3.62. Грани тел до разрезания

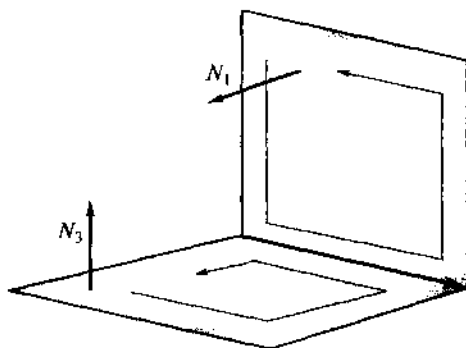


Рис. 3.63. Грани тел после разрезания

на концах изображают направления реструктурированных граничных циклов, каждый из которых представляется в виде списка ребер в порядке их следования и списка флагов ориентации ребер в цикле.

На рис. 3.63 видно, что при данной ориентации ребро пересечения войдет в граничный цикл грани первого тела с положительным флагом, а в список ребер грани второго тела — с отрицательным флагом. Все старые ребра тел-операндов, которые остались в результирующем теле, сохраняют свои исходные флаги.

Рассмотрим правила реструктуризации циклов пересекающихся граней. Начнем с пересекающейся грани, принадлежащей первому телу. Найдем любое ребро пересечения этой грани и, начиная с него, будем строить список ребер граничного цикла. Ребра пересечения входят в цикл с отрицательным флагом, поэтому цикл будет иметь направление, противоположное первому ребру. Чтобы найти продолжение цикла, следует рассмотреть все старые ребра и ребра пересечения, которые имеют с первым общую вершину. Из этого множества выберем такое ребро, которое лежит левее остальных, если смотреть вдоль цикла с наружной стороны грани.

Выбранное ребро включаем в состав граничного цикла. Если оно относится к старым, то сохраняем за ним положительное значение флага. Если оно является новым ребром (ребром пересечения), то присваиваем ему отрицательный флаг. В процессе включения ребер в состав цикла новые ребра имеют больший приоритет по сравнению со старыми. Иными словами, если кандидатами на включение оказались два совпадающих ребра — старое и новое, то предпочтение отдается ребру пересечения. Процесс построения цикла продолжается до его замыкания.

Если у обрабатываемой грани использованы не все ребра пересечения, то следует создать еще один цикл, причем можно начать с любого из оставшихся ребер пересечения. Реструктуризация циклов продолжается до полного исчерпания ребер пересечения. В результате на обрабатываемой грани будет создано несколько независимых граничных циклов.

Множество новых граничных циклов разделим на группы, каждая из которых состоит из одного внешнего цикла и всех входящих в него внутренних цик-

лов. Назовем фиксированными такие циклы старой границы, у которых ни одно ребро не вошло в перестроенные циклы. Среди старых фиксированных циклов выберем такие, которые целиком лежат внутри новых внешних циклов. Их следует включить в состав новой перестроенной границы грани.

Кроме того, следует принять решение о включении в состав границы внешних фиксированных циклов. Если существует новый цикл, который не принадлежит никакому новому внешнему циклу, а расположен внутри старого фиксированного цикла, то последний включается в состав перестроенной границы.

Описанную реструктуризацию граничных циклов следует выполнить для каждой грани первого тела, имеющей пересечение с гранями второго тела. Второе тело требуется обработать согласно приведенному описанию, но внести в алгоритм одно важное изменение — все ребра пересечения должны войти в перестроенные граничные циклы второго тела с положительным флагом.

Таким образом, задача реструктуризации граней пресекающихся тел решена. Для получения результирующего тела требуется добавить все грани, которые не участвовали в пересечении, но топологически связаны с измененными гранями. Для этого достаточно последовательно рассмотреть все ребра, входящие в оболочку нового тела, и включить в состав границы такие смежные с ребрами грани, которые отсутствуют в полученном описании.

3.4.6. Модели пространственного разбиения

Пространственное разбиение (spatial-partitioning) — представление трехмерных объектов в виде совокупности непересекающихся элементарных пространственных форм — примитивов. Примитивы выполняют функции геометрических строительных элементов и могут различаться типом, параметризацией, размерами, пространственной ориентацией и другими свойствами формы, положения и описания. Степень декомпозиции трехмерного объекта и выбор примитивов определяются сложностью формы и целями моделирования.

Декомпозиционная модель

Одним из наиболее общих видов описания, основанных на пространственном разбиении, является так называемая декомпозиционная модель (cell decomposition) трехмерной геометрии. В каждой системе моделирования декомпозиционного типа задается множество параметризованных геометрических примитивов и набор допустимых операций. Множество операций обычно ограничивается различными вариантами упрощенного булевского объединения объектов. Упрощение может состоять в обработке непересекающихся операндов или операндов, имеющих общую вершину, ребро или грань. В декомпозиционных системах моделирования трехмерных объектов подобные операции называются склеиванием. Генерация формы трехмерного тела реализуется по принципу снизу вверх,

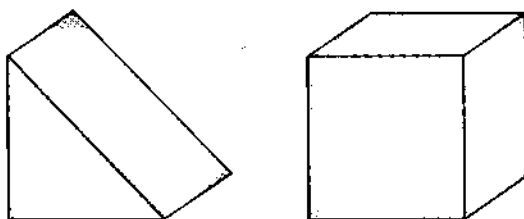


Рис. 3.64. Геометрические примитивы

когда тела, полученные склеиванием базовых примитивов, используются как операнды для синтеза более сложных трехмерных форм.

Представление трехмерного тела в виде композиции геометрических примитивов является вполне однозначным, однако обратный переход от формы к набору элементов не обладает этим свойством. Простой пример, приведенный на рис. 3.64 и 3.65, подтверждает этот тезис. Трехмерное тело, показанное на рис. 3.65, можно получить различными вариантами склейки параметризованных геометрических примитивов: призмы и параллелепипеда.

В общем случае моделирование пространственных тел посредством декомпозиции требует значительных вычислительных ресурсов. Достаточно сказать, что каждую пару примитивов, участвующих в образовании результирующего тела, иногда требуется проверить на непересечение. Существуют проектные ситуации, требующие и других проверочных мероприятий для декомпозиционных форм. Несмотря на отмеченные недостатки, этот способ моделирования нашел широкое применение в системах конечно-элементного анализа и автоматизированного проектирования.

Воксельное описание

Воксельное описание является частным случаем декомпозиционной модели, в которой элементами пространственной формы являются одинаковые примитивы, расположенные в узлах регулярной сетки. В этой модели примитивы выполняют функции неделимых, непараметризуемых элементов трехмерных форм и

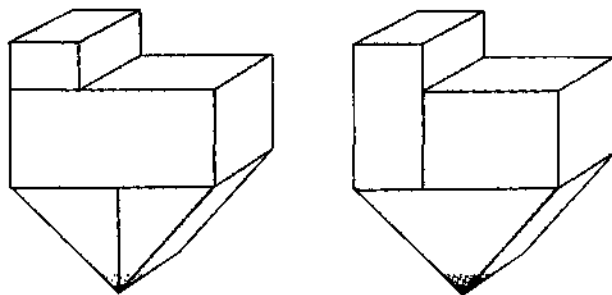
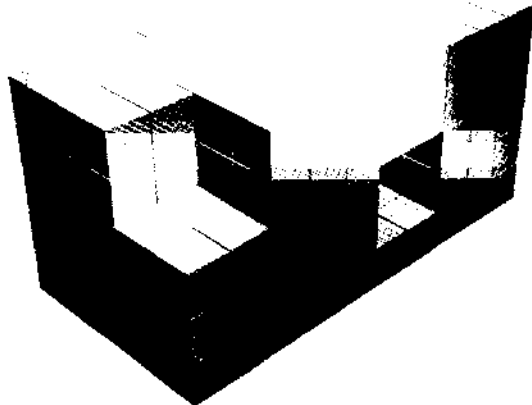


Рис. 3.65. Составные объекты, полученные склеиванием примитивов



• Рис. 3.66. Воксельное представление трехмерного тела

по аналогии с двумерной растровой графикой, где функции графических атомов выполняют пиксели, называются вокселями (voxels, volume elements).

Для описания трехмерной формы в этой модели следует принять решение о наличии или отсутствии элементов на позициях регулярной сетки. Это дает точное и единственное представление формы пространственного тела.

На рис. 3.66 показан пример описания трехмерного тела в терминах пространственных пикселей — пластинчатого тела с выбранным углом и отверстием.

Воксельная модель отличается не только предельной простотой определения, по сравнению с другими средствами трехмерной графики она характеризуется несколькими очевидными преимуществами в вычислениях. Например, элементарное решение в этой модели имеет задача определения принадлежности ячейки телу или проблема пересечения двух тел. Для объемного раstra просто рассчитать такие важные для любого технического проекта параметры, как массу и момент инерции. Они находятся простым сложением соответствующих параметров отдельных примитивов. Воксельная модель описывает геометрию объемного тела и при этом автоматически определяет часть пространства, расположенную за пределами объекта, что позволяет использовать данное описание для моделирования и расчета тел с полостями. Применяется она и для формализации среды в задачах моделирования поведения роботов и манипуляторов.

Достоинства и недостатки воксельного описания совпадают со свойствами плоских растровых моделей.

В большинстве случаев примитивами воксельной модели являются кубы небольшого размера. Сетка, составленная из таких ячеек, не дает точного описания многих трехмерных форм со сложными криволинейными оболочками. Для получения приемлемой аппроксимации необходимо уменьшить размеры примитивов и увеличивать их количество. Если размерность пространственного раstra равна n , то для описания геометрии может потребоваться вплоть до n^3 примити-

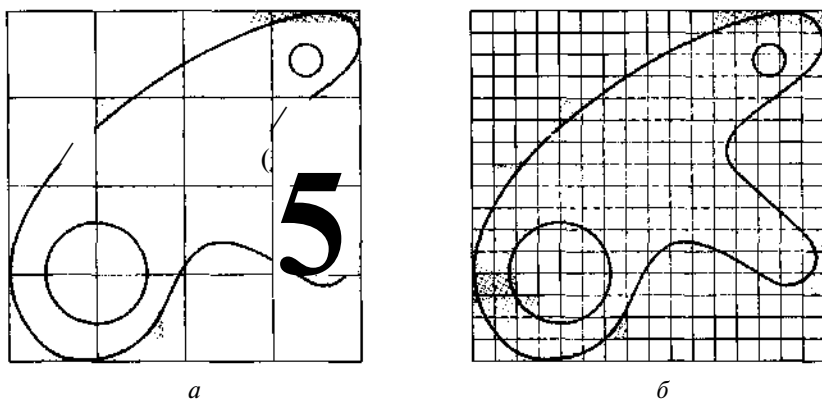


Рис. 3.67. Растровое пространство в процессе разбиения:
а — второй шаг; *б* — последний шаг

вов. С ростом размерности это число увеличивается очень быстро, в случае пространственных форм сложной конфигурации и большой размерности может привести к перерасходу вычислительных ресурсов, прежде всего оперативной памяти.

Несмотря на отмеченные недостатки, системы, основанные на воксельном описании, нашли широкое применение в биомедицинских приложениях — с их помощью представляются трехмерные сканы, полученные современными компьютерными томографами.

Восьмеричные деревья

Восьмеричные деревья — один из способов экономичного представления пространственного растра. Этот тип деревьев представляет собой расширение квадратичных деревьев, которые служат информационной моделью обычного плоского растра. Данный способ представления пространственного растра используется с начала 60-х годов прошлого века.

Квадратичные деревья описывают процедуру последовательного деления двухмерного массива точек на квадранты. Рассмотрим правила построения этих объектов на примере плоской детали, показанной на рис. 3.67. Пространство плоского растра будем последовательно делить на квадранты. Квадрант, закрашенный полностью, обозначим буквой *F* (**full**), закрашенный частично — *P* (**partial**), а свободный от закрашки — *E* (**empty**). Первое разбиение состоит из четырех больших квадрантов, граница которых проходит по самой середине массива точек. Все элементы первого шага разбиения являются частично окрашенными и получают пометку *P*.

Порядок деления, сохраняющийся на всех операциях декомпозиции растрового поля, показан на рис. 3.68. Именно в этом порядке записываются вершины квадратичного дерева.

Результаты второго шага разбиения представлены на рис. 3.67, а. Рисунок показывает, что на этом этапе появляются полностью закрашенные квадранты. Деление таких объектов не дает новой информации, поскольку элементы закрашенных квадрантов всегда являются закрашенными, а элементы пустых — пустыми. Процедуру разбиения растрового поля на квадранты принято представлять в виде квадратичного дерева. Его вершиной является исходное поле, а висячими вершинами служат закрашенные или свободные от закрашки квадранты, записанные в последовательности, изображенной на рис. 3.68.

3	4
1	2

Рис. 3.68. Порядок перечисления квадрантов

На рис. 3.69 показан фрагмент квадратичного дерева, описывающего двухмерный растр из примера, изображенного на рис. 3.67. Полный вид этого дерева довольно громоздок, поэтому некоторые частично заполненные вершины не получили продолжения до уровня висячих вершин.

Путем обрезания дерева в полных и пустых вершинах достигается заметная экономия выразительных средств и памяти вычислительной системы. Тем не менее растры высокой размерности для описания могут потребовать очень больших квадратичных деревьев. Исследователями в области машинной геометрии предложено несколько способов сокращения их размеров. Один из возможных подходов состоит в аппроксимации квадрантов с частичным заполнением. В большинстве проектных ситуаций область со значительным преобладанием заполненных ячеек можно считать полной, и наоборот, квадрант, имеющий небольшое заполнение, рассматривается как пустой. Принять решение о квадрантах с частичным заполнением позволяют пороговые значения плотностей.

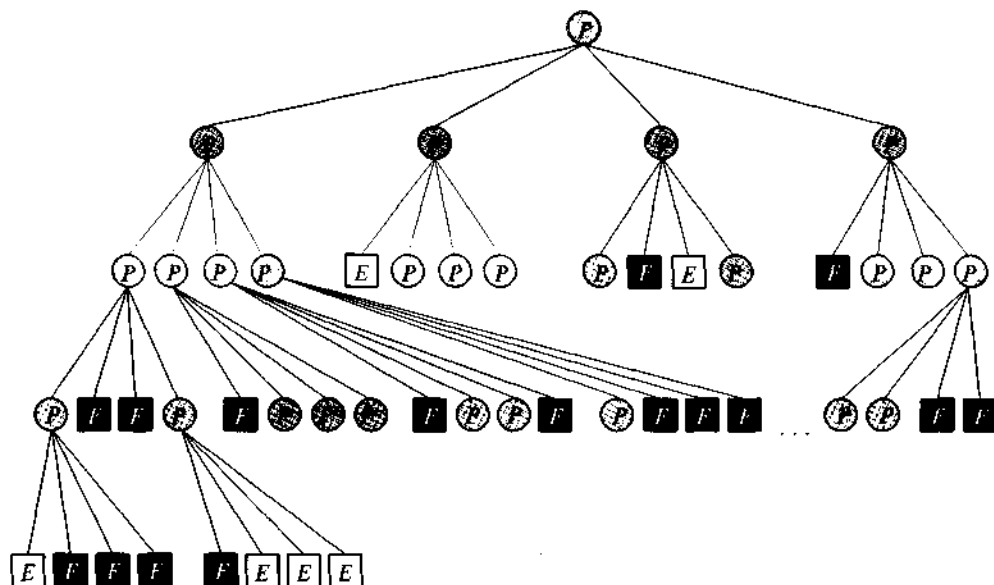


Рис. 3.69. Фрагмент квадратичного дерева

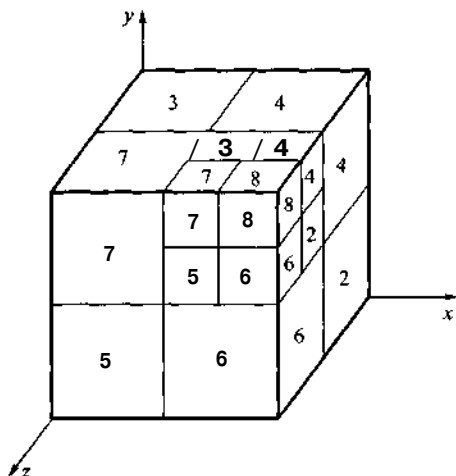


Рис. 3.70. Нумерация октантов

Описанный подход можно распространить и на пространственный растр. Дискретное трехмерное пространство подвергается последовательному делению на кубы. Для описания процедуры пространственной декомпозиции используются так называемые восьмеричные, или октантные, деревья. Корнем восьмеричного дерева служит исходный трехмерный растр, а его висячими вершинами — целиком заполненные или пустые октанты.

Примем соглашение о наименовании октантов. Поскольку не существует общепринятой системы нумерации октантов, будем различать их по направлениям, которые они занимают относительно центра родительского объекта. Обозначим направления

координатных осей через L — налево (left), R — направо (right), U — вверх (up), D — вниз (down), F — вперед (front), B — назад (back). Теперь октанты можно обозначить в соответствии с занимаемой позицией: LUF , LUB , LDF , LDB , RUF , RUB , RDF и RDB . Для экономии изобразительных средств зададим направлениям порядковую нумерацию, начинающуюся с единицы. Соответствие номеров и октантов показано на рис. 3.70; октанты с номером 1 здесь не показаны.

Между периметром плоского тела и числом узлов четвертичного дерева существует очевидная зависимость. Можно показать, что за исключением нескольких предельных случаев эта зависимость описывается прямо пропорционально. В самом деле, область внутри границы и за ее пределами не требует уточнения. В четвертичном дереве соответствующие вершины являются висячими и не нуждаются в уточнении. Описания требуют только частично заполненные квадранты, лежащие на границе. Аналогичная зависимость существует между площадью поверхности объемного тела и числом вершин восьмеричного дерева.

Булевские операции и преобразования растров

Исследователями в области КГ были выполнены изыскания в области разработки эффективных алгоритмов для расчета и хранения четвертичных и восьмеричных деревьев. Оказалось, что многие задачи вычислительной геометрии допускают простые и экономичные решения в терминах пространственных и плоских растров. Так, невысокой трудоемкостью расчета обладают булевские операции объединения, вычитания и пересечения.

Рассмотрим способ нахождения булевого объединения двух растровых представлений. Пусть двухмерные тела A и B заданы в виде квадратичных деревьев. Обозначим C их булевскую сумму. Чтобы получить квадратичное дерево

тела C , будем просматривать деревья операндов параллельно от корня к листьям. По заполнению одноименных узлов операндов легко найти состояние соответствующего узла результата.

Если хотя бы один из узлов операндов является черным (заполненным), то в данной области плоского раstra существует непустое булевское объединение. Если один из узлов является белым (незаполненным), то у дерева объекта C создается узел соответствующего положения, цвет которого зависит от цвета узла другого операнда. Если оба исходных узла оказались серыми (частично заполненными), то к дереву фигуры C добавляется соответствующая серая вершина, а алгоритм рекурсивно применяется к потомкам обработанных узлов обоих операндов. Если в процессе создания дерева C все потомки некоторой вершины получили черный (белый) цвет, то они удаляются из C , а их родительская вершина окрашивается в черный (белый) цвет. Алгоритм прекращает работу после просмотра всех вершин исходных деревьев.

На рис. 3.71 показаны два простых двухмерных растровых объекта со своими квадратичными деревьями, а на рис. 3.72 приведены результаты булевого объединения и вычитания этих операндов и древесные модели результирующих объектов.

Многие задачи геометрических преобразований объектов, заданных в виде квадратичных и восьмеричных деревьев, имеют простые алгоритмические решения. Так, для реализации поворота объекта на 90° требуется применить процедуру рекурсивной обработки ко всем потомкам вершин каждого уровня. Просто решается и задача зеркального отражения и масштабирования тела с любым целым коэффициентом преобразования. Сложнее выполняется расчет перемещения и сглаживания объектов.

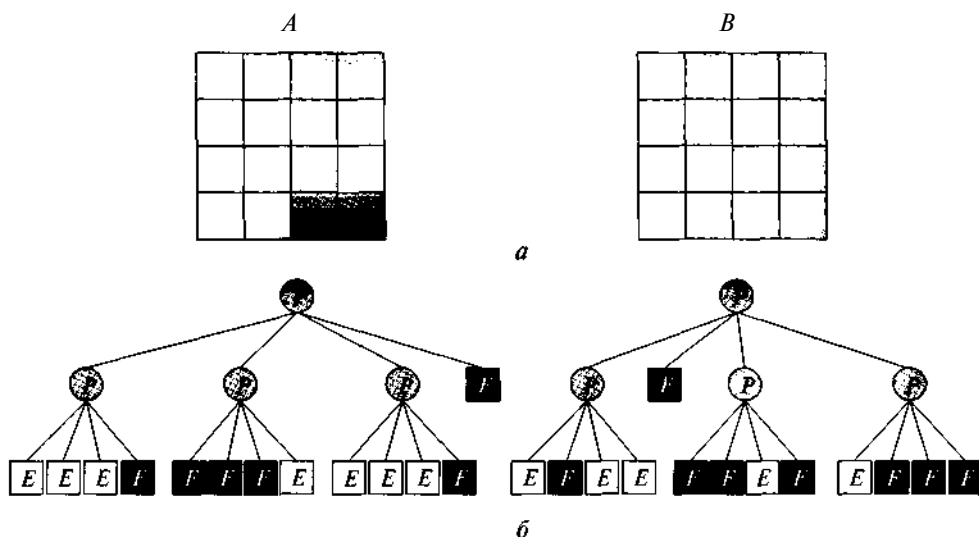


Рис. 3.71. Двухмерные растровые объекты (а) и их квадратичные деревья (б)

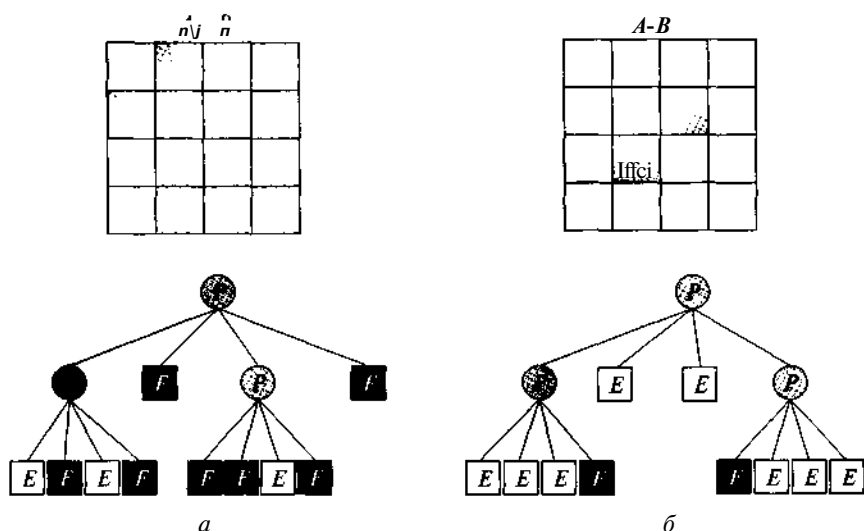


Рис. 3.72. Представление булевских операций в виде квадратичных деревьев:

a — булевское сложение; *b* — булевское вычитание

Поиск соседей

Топология реальных объектов изобилует неоднородностями. Вклад в негомогенный характер растрового описания вносит разбиение формы на грани, ребра и вершины. Если представить плоский или пространственный растр в виде дерева, то сведения о соседстве пикселей или вокселей будут недоступны для непосредственного использования. Для восстановления информации о совокупности растровых элементов, образующих грань или ребро, требуется разработать алгоритм нахождения соседей выбранной вершины. С этой задачей сталкиваются во многих ситуациях, связанных с обработкой квадратичных или восьмиричных деревьев.

Элемент квадратичного дерева в общем случае может иметь восемь соседей, расположенных в разных направлениях на плоском поле растра. Следуя английской традиции, обозначим четыре основных направления первыми буквами сторон света: *N* — север (North), *S* — юг (South), *E* — восток (East), *W* — запад (West). Диагональные направления получают естественные обозначения: *NW*, *NE*, *SW* и *SE*. Элементы восьмиричных деревьев могут иметь до 26 возможных соседей по числу направлений в пространственном растре: 6 — в направлении граней, 12 — в направлении ребер и 8 — в направлении вершин.

Во всех случаях решением задачи является один из потомков родительской вершины, общей для данной вершины и ее соседа. Процедура поиска вершины, соседствующей с выбранной в данном направлении, основана на исследовании дерева. Сначала ищется ближайший общий предок текущей вершины и ее сосе-

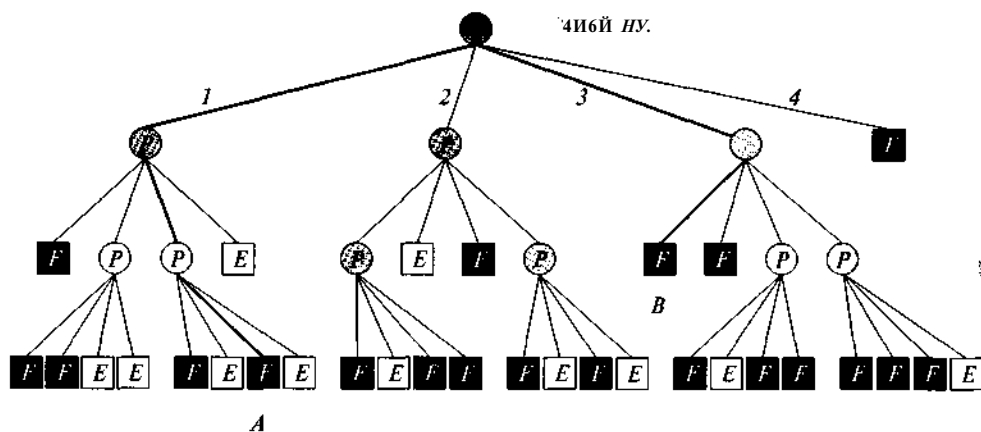


Рис. 3.73. Поиск соседей по квадратичному дереву

да, затем поддереву просматривается в направлении сверху вниз для определения искомого соседа.

Рассмотрим самый простой случай, когда в восьмеричном дереве поиск ограничен направлениями нормалей к граням куба, которые будем обозначать L (Left), R (Right), U (Up), D (Down), F (Front) и B (Back). Обозначим выбранное направление поиска d . Будем подниматься по дереву вверх, начиная от выбранной вершины. Общий предок представляет собой первую вершину, которая не может быть достигнута из своего потомка в направлении d . Пусть для примера поиск ведется в направлении левого соседа L . Тогда первый общий предок представляет собой первую вершину, которая не может быть достигнута из потомков, лежащих в направлениях LUF , LUB , LDF или LDB . Это утверждение справедливо, поскольку вершина, достижимая из своих потомков в указанных направлениях, не может иметь левого соседа.

После нахождения общего предка определяется поддереву, имеющее этого предка в качестве своего корня. Искомое решение лежит на пути, который ведет из корня поддереву и является зеркальным отражением маршрута, проложенного из стартовой вершины.

Рассмотрим этот метод на примере квадратичного дерева (рис. 3.73), задающего двухмерный объект, показанный на рис. 3.74. Цифрами 1-4 помечены основные направления двухмерного поля (см. рис. 3.68). Пусть в этой ситуации требуется найти северного соседа вершины A .

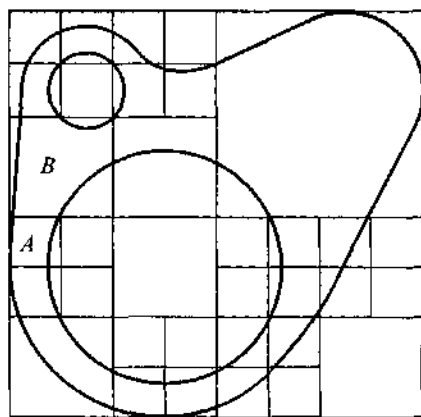


Рис. 3.74. Двухмерный растровый объект

Восходящая фаза поиска начинается с вершины A (см. рис. 3.73). Эта вершина является AW -предком (северо-западным) для своего непосредственного родителя. Весь путь до искомой родительской вершины показан на рис. 3.73 толстой линией. Его образуют два предка, лежащих в направлении NW , и один в направлении SW . В нашем случае корневая вершина дерева служит искомым предком, поскольку она достигнута в направлении SW . В этом направлении даже частично не используется направление N , выбранное для поиска соседа.

Далее для поиска решения следует проложить маршрут в нисходящем направлении. Он начинается в корневой вершине и представляет собой зеркальное отражение восходящего маршрута (показанного на рис. 3.73 толстой линией) относительно меридиана NS .

Решением задачи поиска северного соседа вершины A служит вершина B квадратичного дерева.

3.4.7. Конструктивная твердотельная геометрия

Конструктивной твердотельной геометрией (Constructive Solid Geometry — CSG) называется способ описания трехмерных объектов, основанный на некотором наборе базовых геометрических примитивов и операциях над ними. Программные реализации этого подхода используют различные множества операций и операндов, но чаще всего в набор базовых форм входят простые геометрические тела, имеющие точное математическое описание, а разрешенными преобразованиями служат регулярные булевы операции.

Любой объект, полученный методом конструктивной геометрии, можно представить в виде дерева, в котором висячими вершинами являются базовые примитивы, внутренние вершины представляют собой промежуточные формы и результаты применения булевских операций, а корень соответствует готовому объекту. В общем случае булевы операции не являются коммутативными, поэтому вершины каждого уровня CSG-дерева считаются упорядоченными.

На рис. 3.75 приведен пример простого трехмерного объекта, который представляет собой плоскую деталь с одним сквозным отверстием.

Дерево, показанное на рис. 3.76, описывает геометрию детали в терминах базовых форм и булевских операций над ними.

Физические свойства корневого объекта и его графическое представление зависят от примитивов, поставленных в соответствие листьям CSG-дерева. Для их нахождения требуется применить стратегию исследования дерева, на-

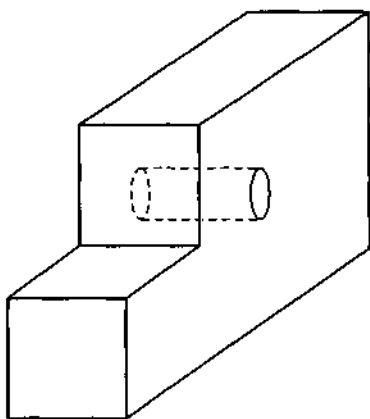


Рис. 3.75. Пример трехмерного объекта

поминающую известный алгоритм поиска в глубину. При этом возможность получения геометрического представления и трудоемкость этой процедуры зависит от способа описания геометрических примитивов. Если, например, исходные объекты описаны в системе граничного представления (B-гер), то расчет результатов регулярных булевских операций становится нетривиальной алгоритмической задачей. Если примитивы представляют собой простые объекты с точным аналитическим описанием геометрии, то проблема генерации финального образа существенно упрощается.

Некоторые программные реализации конструктивной твердотельной геометрии оперируют с простыми геометрическими примитивами, в описании которых используются полупространства. Например, куб представляется в виде пересечения шести полупространств, конечный цилиндр задается как бесконечный цилиндр, ограниченный плоскостями сверху и снизу, и т. п.

Использование полупространств влечет за собой как очевидные удобства, так и скрытые недостатки, главным из которых является проблема целостности описаний. Понятно, что некоторые операции с полупространствами способны породить формы, не относящиеся к замкнутым твердотельным объектам.

Конструктивная твердотельная геометрия — это не точное математическое описание с глубоко разработанным аппаратом. Она представляет собой парадигму, которая допускает значительные вариации базовых примитивов, производящих операций и их описаний. Базовые соглашения этой системы столь широки, что позволяют считать частными случаями рассмотренные ранее декомпозиционную и воксельную модели. В этих моделях для генерации трехмерных форм используется только одна операция — булевское сложение объектов, имеющих нулевое пересечение внутренних областей. В общем случае конструктивная геометрия тел не позволяет получить строго единственное описание геометрии объекта. Это подтверждает простой пример, показанный на рис. 3.77. Здесь одна результирующая форма может быть представлена разными CSG-деревьями.

Еще одним источником неопределенности в конструктивной геометрии тел также является использование геометрических примитивов, которые допускают параметрическую настройку. Эту ситуацию демонстрирует рис. 3.78 — измене-

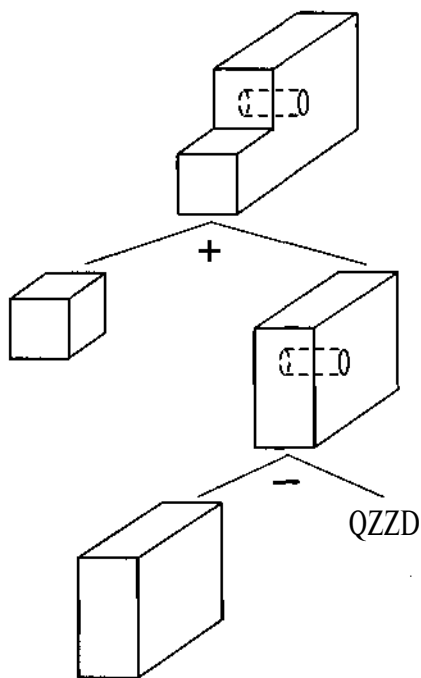


Рис. 3.76. Представление пространственного тела в виде CSG-дерева

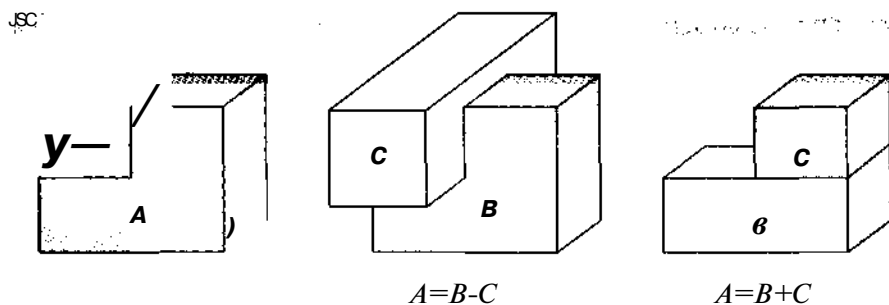


Рис. 3.77. Пример неоднозначного описания объекта в системе CSG

ние высоты параллелепипеда одного из операндов дает различный результат операции булевского вычитания.

Назовем и другие недостатки конструктивной геометрии тел:

- дерево CSG предназначено для записи применения булевских операций в их хронологической последовательности. По этой причине к CSG-объектам не могут быть применены иные средства локального или глобального формообразования, например снятие фаски или экструдирования;

- конструктивное представление геометрии не включает эксплицитной информации о граничных поверхностях, ребрах и связях между этими элементами описания. Для генерации таких сведений требуется выполнить большой объем сложных вычислений по CSG-дереву. Поэтому данный способ описания не очень хорошо подходит для интерактивного представления трехмерных объектов и манипуляций с ними. Составление чертежа по готовой модели объекта, моделирование перемещений режущего инструмента, имитация поведения робота — примеры задач, которые в парадигме конструктивной геометрии тел решаются со значительными трудностями.

Нельзя не упомянуть и о достоинствах конструктивной геометрии тел:

- предоставляет пользователю простой способ проектирования объектов сложной геометрии, основанный на операциях сложения, вычитания, замены геометрических примитивов. Информация о структуре проекта хранится в компактной и наглядной форме CSG-дерева;

- разрешает выполнять преобразования дерева, что является еще одним мощным ресурсом формообразования;

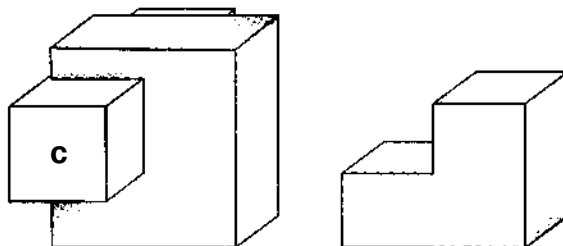


Рис. 3.78. Пример неоднозначности CSG-описания, вызванной параметрической настройкой

- объемное тело, описываемое CSG-деревом, всегда является корректным, т. е. его внутренний объем однозначно отделен от внешнего. За исключением нескольких предельных случаев операции с примитивами в CSG синтезируют трехмерное многообразие;

- по CSG-описанию всегда может быть получено представление геометрии объекта в системе В-гер. Такой переход часто требует трудоемких вычислений, но в большинстве случаев принципиально возможен. Это позволяет организовать взаимодействие систем моделирования, которые базируются на CSG, с чертежными системами и анимационными пакетами, основанными главным образом на представлении В-гер.

Конструктивная геометрия тел является основным способом описания трехмерной геометрии в современных пакетах геометрического моделирования различной отраслевой принадлежности и целевого назначения.

S
f

3.4.8. Сравнение представлений

Обсудим основные достоинства и недостатки рассмотренных в данной главе систем описания трехмерной геометрии.

Точность. Все модели пространственного разбиения и методы, основанные на описании границ полигонами, представляют собой аппроксимации реальных объектов. Только в немногих частных случаях они дают точное описание геометрии. Эту особенность нельзя считать серьезным недостатком, поскольку многие приложения не требуют высокой точности моделирования. В качестве примера достаточно назвать такую бурно развивающуюся отрасль машинной графики, как программирование игр. Системы, в которых требуется высокая точность моделирования, чаще всего построены на основе конструктивной геометрии тел с неполиэдральными геометрическими примитивами. Средствами параметрического моделирования можно создавать объекты с геометрией высокой точности, но такая разработка требует высоких затрат ручного труда.

Область применения. Области применения объектов, созданных при помощи параметрического моделирования и посредством замещения, являются весьма ограниченными. Наоборот, средствами техники пространственного разбиения можно создавать любые формы и объекты, но в общем случае только как приближение к реальной или идеальной геометрии образца. В настоящее время наиболее распространены системы граничного описания, в которых разрешается использовать многоугольники и прямые линии. Понятно, что выразительные возможности такой системы моделирования являются ограниченными. Если для описания границ использовать двухмерные и одномерные многообразия общего вида, то подобными средствами можно представить множество различных форм.

Однозначность. Большая часть рассмотренных систем моделирования не обладает свойством однозначности. Иными словами, они не дают строго единственного описания геометрии объекта. Это утверждение относится ко всем моделям за исключением пространственного разбиения и параметрического моделирования,

которые могут синтезировать уникальное описание при соблюдении некоторых дополнительных условий.

Корректность. Среди всех моделей граничное представление отличается самой высокой трудоемкостью вычислений, необходимых для подтверждения корректности. Корректность CSG-описания можно установить посредством очень простой синтаксической проверки древесной структуры. Модели пространственного разбиения вообще не нуждаются в такой процедуре, поскольку их корректность гарантируется автоматически.

Вопросы для самоконтроля

1. Назовите способы представления полигональных сеток.
2. Что понимают под согласованностью полигональных сеток?
3. Что такое нормаль к плоскости и как ее координаты связаны с уравнением плоскости?
4. Выведите зависимости коэффициентов, задающих уравнение несущей плоскости многоугольника, от его проекций.
5. Назовите три основных подхода к описанию пространственных кривых, перечислите достоинства и недостатки этих способов описания.
6. Перечислите основные принципы кусочно-нелинейной аппроксимации пространственных кривых.
7. Какие преимущества дает использование кубических полиномов для задания аргументов фрагмента пространственной кривой?
8. Дайте определения геометрической непрерывности класса G^0 и G^1 .
9. Дайте определения непрерывности классов C^0 , C^1 и C^2 .
10. В чем заключается физический смысл касательного вектора $\frac{dQ(t)}{dt}$ пространственной кривой?
11. Дайте определение геометрического вектора и стыковочной функции.
12. Дайте определение кривых Эрмита. Назовите их достоинства и недостатки.
13. Дайте определение кривых Безье. Перечислите их достоинства и недостатки.
14. Дайте определение базиса Бернштейна и назовите его основные особенности.
15. Дайте определение кубического В-сплайна. В чем состоит отличие узлов сплайна от его контрольных точек?
16. Назовите главное отличие однородных сплайнов от неоднородных.
17. Какие сплайны называются рациональными? В чем состоит основное преимущество этого типа параметрических кубических кривых?
18. Назовите основные способы визуализации пространственных кривых.
19. Дайте определение поверхностей Эрмита.
20. Назовите основные подходы к решению задачи визуализации параметрических бикубических поверхностей.
21. Какое описание твердого тела называется каркасной моделью?
22. Перечислите преимущества, которые дает применение регулярных булевских операций.
23. Назовите основные принципы, лежащие в основе параметрического подхода к моделированию геометрии пространственных тел.

Вопросы для самоконтроля

24. Назовите основные положения и преимущества граничного моделирования трехмерных тел.
25. Запишите формулу Эйлера для полиэдров.
26. Перечислите структуры данных, предназначенных для описания тел в системе В-гер.
27. Назовите основные реализации декомпозиционного подхода к твердотельному моделированию.
28. В каком порядке рассматриваются квадранты плоского растра при разбиении вершин квадратичного дерева?
29. Как представляется форма трехмерного объекта в твердотельной конструктивной геометрии?
30. Оцените достоинства и недостатки различных способов описания трехмерных объектов.

4. МЕТОДЫ, АЛГОРИТМЫ И ФОРМАТЫ ФАЙЛОВ КОМПЬЮТЕРНОЙ ГРАФИКИ

Рассматриваются алгоритмические основы КГ и форматы файлов обмена графической информацией между различными программными системами КГ. Описываются методы и алгоритмы двухмерной (2D) графики и геометрии, методы и алгоритмы трехмерной (3D) графики и геометрии, а также алгоритмы анимации трехмерных моделей, форматы векторных и растровых файлов в программных системах КГ.

4.1. Методы и алгоритмы двухмерной компьютерной графики

В зарубежной литературе по КГ традиционно принято классифицировать методы и алгоритмы КГ на три основные группы:

- компьютерное зрение (ввод графических изображений и синтез графической информации) — Computer Vision;
- обработка изображений (обработка графической информации) — Image Processing;
- компьютерная графика (вывод графических изображений) — Computer Graphics.

Выпускаются и соответствующие журналы, например «Computer Vision, Graphics and Image Processing». В настоящее время в КГ есть такой важный раздел, как геометрическое моделирование, который отнесли к КГ, поэтому в данной книге использован другой подход к классификации методов и алгоритмов КГ.

Любой компьютерный алгоритм предполагает наличие входных, выходных данных и методов обработки входных данных для получения нужных выходных данных. В области КГ разработано огромное количество алгоритмов. Для двухмерной компьютерной графики и геометрии данные подразделяются на два класса: векторные и растровые. Напомним, что векторные данные представляют собой множество опорных (ключевых) точек для математического описания кривых линий, а также информацию об их атрибутах (цвет, толщина и другие параметры) и признаках заполнения замкнутых областей, ограниченных кривыми линиями; растровые данные представляют собой набор численных значений, определяющих яркость и цвет отдельных пикселей в заданной двухмерной области. В связи с этим алгоритмы двухмерной (2D) графики можно классифицировать по четырем основным группам:

- с растровыми входными и выходными данными (цифровая обработка изображений);
- с растровыми входными данными и векторными выходными данными (векторизация, распознавание образов);
- с векторными входными данными и растровыми выходными данными (растеризация);
- с векторными входными и выходными данными (геометрическое моделирование, двумерные геометрические преобразования, решение метрических задач на плоскости, параметрическая и вариационная геометрия).

В литературе принято описывать алгоритмы с использованием блок-схем, псевдоязыков или самих языков программирования (обычно на основе алгоритмических языков Paskal, Modula, C или C++). В связи с тем, что количество конкретных алгоритмов двумерной графики очень велико, далее будут рассмотрены только основные, фундаментальные идеи и алгоритмические основы тех или иных классов алгоритмов, а более подробно, но без привлечения блок-схем и языков программирования будут рассмотрены только те алгоритмы, которые де-факто стали классическими.

Большинство графических систем программирования содержат процедуры и функции для реализации многих алгоритмов машинной графики и геометрии, причем некоторые алгоритмы уже реализованы аппаратно, однако надо знать особенности этих алгоритмов для грамотного использования соответствующих процедур и функций и, возможно, для написания аналогичных.

4.1.1. Входные и выходные данные растровые

Для преобразований растровых данных в растровые обычно используют термин «цифровая обработка изображений». Цифровая обработка изображений — область машинной графики, связанная с обработкой растровых данных изображений. Она охватывает большой спектр методов, которые имеют широкое применение. Цифровая обработка изображений применяется, когда необходимо:

- повысить качество изображения или модифицировать исходное изображение для того, чтобы выделить некоторые аспекты информации, содержащиеся в изображении, как подготовительный этап в задачах распознавания образов, рассмотренных в подразд. 4.1..2;
- классифицировать, сравнить или измерить элементы в изображении;
- скомбинировать части изображений или реорганизовать элементы изображения;
- получить видимое изображение цифровых данных, усиленных определенным образом для выделения некоторых аспектов данных. Такие типы обработки изображения используются, например, в медицинском оборудовании, гидролокаторах, ультразвуковом оборудовании, тепловизионном оборудовании и т. д.

Можно выделить три основных класса алгоритмов обработки растровых данных:

- точечные алгоритмы, когда изменяют значения пикселей, основываясь на исходных значениях самих пикселей и возможную их позицию в изображении;
- пространственные алгоритмы, когда изменяют значения пикселей, основываясь на исходных значениях самих пикселей и пикселей вокруг них;
- алгоритмы геометрических преобразований, когда изменяется расположение или местонахождение пикселей в изображении, основываясь на некоторых геометрических преобразованиях.

В каждом из перечисленных выше классов разработано множество различных алгоритмов обработки изображения. Важно знать, что применение этих алгоритмов не коммутативно, т. е. очень важен порядок их применения.

Далее достаточно подробно описаны только базовые алгоритмы, но сначала рассмотрим алгоритмические понятия, которые лежат в основе этих алгоритмов.

Пусть $A = \{a_{ij}\}$ — растровое изображение, представляющее собой прямоугольную матрицу размером $m \times n$, $i = \overline{1, m}$, $j = \overline{1, n}$; a_{ij} — элемент изображения (пиксел).

Если ПЦ $\in \{0, 1\}$ и принимает только два значения, то изображение называется бинарным и состоит только из черных (black) B и белых (white) W пикселей.

Если ПЦ $\in \{0, 1, \dots, N-1\}$ и принимает N значений, то изображение называется полутоновым и каждый пиксел может принимать N оттенков серого (градаций яркости).

Если $d_{ij} \in \left\{ \begin{pmatrix} x_{ij}^{(1)} \\ x_{ij}^{(2)} \\ x_{ij}^{(3)} \end{pmatrix} \right\}$, где $x_{ij}^{(k)} \in \{0, 1, \dots, N-1\}$, то изображение называется

цветным, каждый пиксел может иметь любое из $N^{(k)}$ возможных значений цвета, определяемое соответствующими ему координатами $(x_{ij}^{(1)}, x_{ij}^{(2)}, x_{ij}^{(3)})$ в цветовом пространстве.

Если $a_{ij} \in \left\{ \begin{pmatrix} d_{ij} \\ x_{ij}^{(2)} \\ x_{ij}^{(3)} \\ \dots \\ x_{ij}^{(M)} \end{pmatrix} \right\}$, где $x_{ij}^{(k)} \in \{0, 1, \dots, N-1\}$, то изображение называется

многоканальным. Такое изображение состоит из совокупности M полутоновых изображений $x_{ij}^{(k)}$. Каждому пикселу соответствует M -компонентный вектор со

значениями яркости, соответствующими ему во всех M изображениях. Рассмотрим алгоритмические основы обработки бинарных и полутоновых изображений.

Пусть B и W — два множества соответственно черных (объект) и белых (фон) пикселей, составляющих бинарное изображение. Каждый пиксел изображения имеет восемь соседних пикселей, которые нумеруются в соответствии со следующей схемой (рис. 4.1): $S^*(au)$ — множество всех соседей au (кроме собственно au), называемое 8-связными соседями au .

Соседи с нечетными номерами называются прямыми соседями au , или 4-связными соседями, обозначаемые как $S_4(au)$; соседи с четными номерами — это не прямые соседи au , которые обозначаются $S_D(a, j)$ (диагональные соседи). В общем случае под понятием соседства понимается $S\%(au)$. Множество $S_8(a, >)$ называется 8-окрестностью $a_{x,y}$, а множество $S_4(au)$ — 4-окрестностью au . Многие алгоритмы на пиксельной плоскости основаны на отношениях соседства.

Пиксел из B , имеющий всех соседей из B , называется внутренним пикселем. Совокупность всех внутренних пикселей B называется ядром, или внутренностью B . Все пиксели B , не являющиеся внутренними, называются контурными пикселями (рис. 4.2).

Одним из важных определений на бинарном изображении является расстояние. Расстояние между двумя пикселями a_{i_1, j_1} и a_{i_2, j_2} — длина наикратчайшего пути, соединяющего эти пиксели на соответствующем графе. Расстояние может быть определено путем выбора подходящего отношения соседства и подходящего определения длины пути.

Евклидово расстояние определяется формулой

$$D(a_{i_1, j_1}, a_{i_2, j_2}) = \sqrt{(i_1 - i_2)^2 + (j_1 - j_2)^2}.$$

Основной проблемой при использовании евклидова расстояния для обработки изображений является то, что оно представляется действительными числами. Вот почему используют многочисленные целочисленные аппроксимации евклидова расстояния, описываемые следующими формулами:

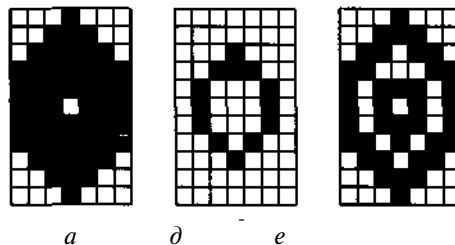


Рис. 4.2. Исходный объект (а), его ядро (б) и контуры (в)

a_4	a_3	a_2
"5"	a_{ij}	"1"
a_6	a_7	"8"

Рис. 4.1. 8-связные соседи пиксела pi_i

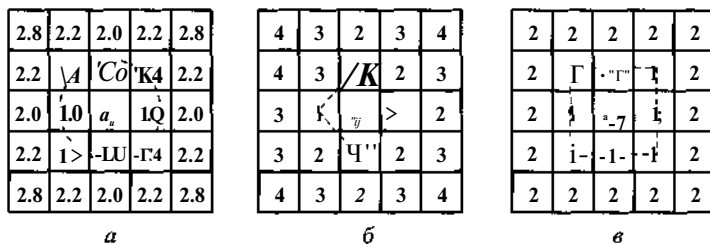


Рис. 4.3. Типы расстояний относительно пиксела:

a — евклидово; b — городское; v — шахматное

$$D(a_{i_1 j_1}, a_{i_2 j_2}) = \begin{cases} w_0 |i_1 - i_2| + w_1 |j_1 - j_2|, & \text{если } |i_1 - i_2| < |j_1 - j_2|; \\ w_1 |i_1 - i_2| + w_0 |j_1 - j_2|, & \text{если } |i_1 - i_2| \geq |j_1 - j_2|, \end{cases}$$

где w — весовые коэффициенты, $w_0, w_1 > 0$.

В зависимости от значений w_0 и w_1 могут быть получены различные типы расстояний. Например, если $(w_0, w_1) = (1, 1)$, то полученное расстояние называется городским расстоянием (city block), если $(w_0, w_1) = (1, 0)$, то полученное расстояние называется шахматным расстоянием (chessboard).

Каждое расстояние имеет свою собственную графическую форму, определяемую множеством всех пикселей, эквидистантных от нескольких центральных пикселей. Таким образом, евклидово расстояние в качестве характеристической формы имеет круг, а городское и шахматное расстояния имеют ромб и квадрат соответственно в качестве характеристической формы (рис. 4.3).

Два пиксела (B или W) называются связными, если они являются соседями (расстояние между ними равно единице) в выбранной метрике.

Связная компонента изображения — связанное множество пикселей в соответствии с выбранным типом метрики. В зависимости от выбранного типа метрики существуют 8-связные линии (рис. 4.4, a), 4-связные линии (рис. 4.4, b) и линии со смешанным типом связности (рис. 4.4, v).

Метрика зависит от расстояния, поэтому по аналогии с типами расстояний выделяют городскую, шахматную и другие метрики. Для сохранения связности объекта и фона используют различные типы связности. Обычно 8-связность применяют для пиксела B , а 4-связность — для W , или наоборот.

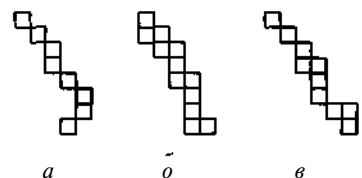


Рис. 4.4. Вид линии в зависимости от вида связности:

a — 8-связная; b — 4-связная; v — со смешанным типом связности

Многие алгоритмы цифровой обработки изображений основаны на математической морфологии, суть которой заключается в том, что исходное изображение рассматривается как множество, и к нему применяются теоретико-множественные операции. Слово «морфология» (от греч. *morphe* — форма) означает форму, структуру. Математическая морфология предназначена для исследования

структуры некоторых множеств однотипных объектов. Любые растровые данные изображения также представляются в виде множеств пикселей, поэтому операции математической морфологии могут быть применены и для исследования некоторых свойств изображения, его формы и структуры, а также для его обработки.

Пусть имеется двоичное изображение, представленное в виде упорядоченного набора (упорядоченного множества) пикселей, причем черным пикселям соответствует цифра 1, а белым — цифра 0. Под областью изображения понимается некоторое подмножество цифр 1 на фоне цифр 0. Каждая операция двоичной морфологии является некоторым преобразованием этого множества. В качестве исходных данных принимаются исходное двоичное изображение A и некоторый структурирующий элемент S . Результатом операции также является результирующее двоичное изображение.

Структурирующий элемент — некоторое двоичное изображение, имеющее определенную геометрическую форму. Чаще всего используются симметричные структурирующие элементы, примеры которых показаны на рис. 4.5. В каждом элементе выделяется особая точка, называемая начальной. Она может быть расположена в любом месте элемента, хотя в симметричных элементах это обычно центральный пиксел.

Сначала результирующая пиксельная плоскость заполняется цифрами 0, образуя полностью белое изображение. Затем осуществляется сканирование исходной пиксельной плоскости пиксел за пикселем с помощью структурирующего элемента. Для зондирования каждого пикселя на изображение накладывается структурирующий элемент так, чтобы совместились зондируемая и начальная точки. Далее проверяется некоторое условие на соответствие пикселей структурирующего элемента и точек изображения под ним. Если условие выполняется, то на результирующем изображении в соответствующем месте ставится цифра 1 (в некоторых случаях будет добавляться не один единичный пиксел, а все единичные пиксели из структурирующего элемента).

По рассмотренной схеме выполняются базовые операции математической морфологии — расширение (dilation) и сужение (erosion). Однако более полезными оказались производные операции — некоторая комбинация базовых операций, выполняемая последовательно. Основными из них являются открытие (opening) и закрытие (closing).

Операция расширения двоичного изображения A на структурирующий элемент S записывается в виде $A @ S$ и определяется как $A @ S = \{c | \exists a \in A, \text{ se } S: c = a + b\}$. На рис. 4.6, а приведен пример расширения исходного изображения.

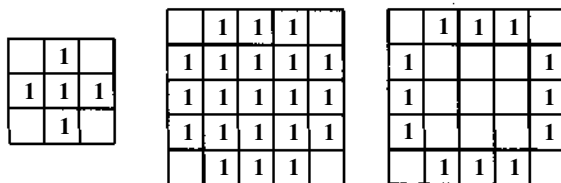


Рис. 4.5. Примеры структурирующих элементов

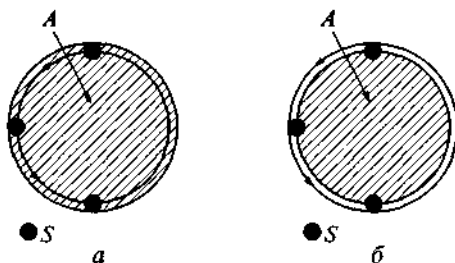


Рис. 4.6. Операции расширения (а) и сужения (б):

A — исходное двоичное изображение; S — структурирующий элемент

Операция сужения двоичного изображения A на структурирующий элемент S записывается в виде $A(-)S$ и определяется как $A(-)S = \{c \mid \forall c \in s \in A, se S\}$. На рис. 4.6, б приведен пример сужения исходного изображения.

Первая основная производная операция — операция закрытия двоичного изображения A на структурирующий элемент S записывается в виде $(A \circ S) - S$. На рис. 4.7, а приведены примеры операции закрытия исходного изображения. Операция закрытия заполняет небольшие внутренние области, принадлежащие изображению, и убирает углубления по краям области.

Вторая основная производная операция — операция открытия двоичного изображения A на структурирующий элемент S определяется как $A \circ S = (A - S) \circ S$. На рис. 4.7, б приведены примеры открытия исходного изображения. Операция открытия удаляет небольшие внутренние области, не принадлежащие изображению, небольшие кусочки изображения, выходящие за границу области изображения, а также расширяет пустые области в изображении.

Следует отметить важное свойство математической морфологии — в результате выполнения морфологических операций в исходном изображении остаются лишь те существенные особенности формы, которые присутствуют в структурирующем элементе, поэтому чтобы не внести в изображение новые ис-

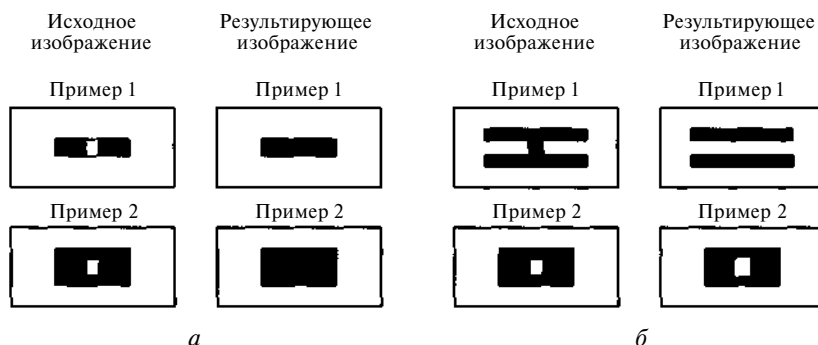


Рис. 4.7. Операции закрытия (а) и открытия (б) изображения

кажения, структурирующий элемент должен быть близок к кругу. В этом случае в изображении останутся все его симметрии.

Полутоновые изображения характеризуются большим в отличие от бинарного количеством градаций яркости. Как и для бинарного изображения, основным понятием для полутонового изображения является понятие объекта и фона, однако в силу природы изображения их определение более сложно по сравнению с бинарными. Обозначим $f(X)$ — функцию значения яркости произвольной строки полутонового изображения вдоль оси X . На рис. 4.8 показаны графики такой функции для бинарного и полутонового изображений.

Объект, как правило, отличается значением яркости от фона. Область I — область фона, область III — область объекта. В силу того, что изображение полутоновое, граница II между объектом и фоном является размытой. Граница на полутоновом изображении — область, ограничивающая объект, при этом I и III — области постоянных значений, а II — область изменения значения яркости — соответствует перепаду яркости, т. е. области, в которой производная df/dX отлична от нуля. Наиболее распространенной задачей в обработке полутоновых изображений является выделение границ — областей II. Результат представляется в виде бинарного изображения, в котором областям I и III соответствует белый цвет, а области II — черный. Линии черного цвета на бинарных изображениях, соответствующие границам объекта на полутоновых изображениях, называются контурными линиями.

Контурным представлением объекта полутонового изображения называется его описание с помощью контурной линии, ограничивающей объект. Контурная линия рассматривается как 8-связная линия на бинарном изображении, которой соответствует черный цвет, фону — белый. Понятия соседства и связности относятся к контурным (черным) пикселям. Среди контурных линий особо выделяют контурную линию шириной в один пиксел, которая наиболее часто применяется в алгоритмах обработки изображений.

Важными являются понятия низкочастотных и высокочастотных компонентов изображения. Преобладание низкочастотных компонентов в некоторой области означает, что все значения яркости пикселей изображения в этой области постоянны или изменяются очень медленно, эти компоненты определяют общую, среднюю яркость изображения, тогда как области изображения с преобладанием высокочастотных компонентов содержат пиксели с быстрым и резким изменением значения их яркости, т. е. с большим количеством мелких объектов в изображении.

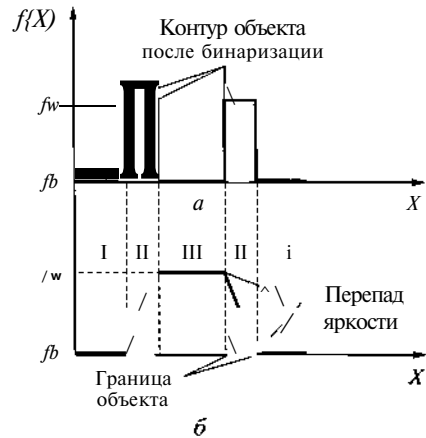


Рис. 4.8. Изменение яркости пикселей для бинарного (а) и полутонового (б) изображений:

I, II, III — области изменения значений яркости

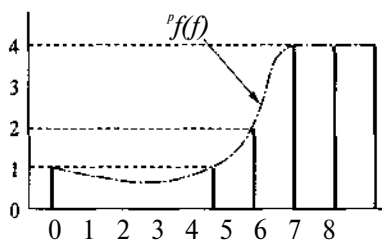


Рис. 4.9. Гистограмма яркости изображения

Полезным средством для анализа и редактирования растровых изображений является гистограмма яркости изображения. Гистограмма яркости изображения — столбчатый график зависимости плотности распределения Pf вероятностей значений яркости от значения / яркости пикселей изображения $P(if)$. На рис. 4.9 показана типичная гистограмма яркости изображения для условного изображения, содержащего 16 пикселей, с восемью возможными значениями градации яркости пикселей. Эта

гистограмма показывает, что в данном изображении преобладают черные и темные пиксели, а пиксели со значениями яркости 1, 2, 3 отсутствуют.

Другим полезным средством при обработке растровых данных является использование таблиц преобразований. Таблицы преобразований оказались особенно удобными для точечных алгоритмов. При использовании таблиц преобразований максимально увеличивается скорость выполнения алгоритмов обработки изображения. Основная идея состоит в замене вычислений по некоторым формулам на поиск нужного значения в таблице преобразований. Таблица предварительно заполняется значениями, которые отражают преобразования, выполненные на растровых данных, проходящих через таблицу преобразований.

Точечные алгоритмы. Точечные алгоритмы — базовые операции обработки изображения. Они очень простые и их наиболее часто используют в алгоритмах обработки изображения. Они полезны как сами по себе, так и в сочетании с другими классами алгоритмов. Точечные алгоритмы — алгоритмы, которые изменяют значение яркости одного пикселя в изображении и основаны только на этом значении и иногда на положении пикселя. Никакие другие значения не включаются в преобразование. Индивидуальные значения яркости пикселя изображения заменяются новыми значениями, которые алгоритмически связаны с исходным значением яркости пикселя. Как результат алгоритмических отношений между исходным и новым значением элемента изображения точечные алгоритмы могут в общем случае выполняться в обратном порядке. Алгоритмы точечных процессов сканируют изображение пиксел за пикселем, осуществляя преобразование пикселей изображения. Если преобразование зависит только от исходного значения яркости пикселей, то этот процесс лучше реализовать с помощью таблиц преобразований. Если преобразование в точечных алгоритмах также учитывает и расположение пикселей, то для преобразования используется какая-либо формула или формула в сочетании с таблицей преобразований. В общем случае точечные алгоритмы не изменяют пространственные соотношения в пределах изображения. Поэтому точечные алгоритмы не могут модифицировать детали, содержащиеся в изображении. Применение точечных алгоритмов к изображениям имеет смысл только для бинарных и полутоновых изображений.

Самые простые алгоритмы этого класса — алгоритмы получения негативных изображений или алгоритмы инверсии яркости пикселей. Негативное изображение создается путем вычитания исходного значения яркости каждого пиксела изображения $f(X, Y)$ из максимально возможного значения яркости f_w (значение яркости белых пикселей):

$$g(X, Y) = f_w - f(X, Y),$$

где $g(X, Y)$ — результирующее значение яркости пиксела X, Y .

В данном случае эффективно использовать таблицы преобразований. Например, диапазон значений яркости 8-битного полутонового изображения составляет от 0 до 255. Соответствующая формула преобразования имеет вид $f(X, Y) = 255 - f(X, Y)$, а таблица преобразования представлена в табл. 4.1.

Таблица 4.1

Индекс	0	1	2	...	255
Значение	255	254	253	...	0

Получение негативных изображений полезно для более сложных преобразований, а также для выделения ярких частей изображения, поскольку глаз человека гораздо более восприимчив к деталям в темной области изображения, чем в светлой.

Другая важная группа алгоритмов этого класса — алгоритмы бинаризации полутоновых изображений, т. е. превращение их в черно-белое (двухградационное) изображение. Такое преобразование осуществляется для того, чтобы сократить информационную избыточность изображения, оставив в нем только ту информацию, которая нужна для решения конкретной задачи (например, для выделения очертания объектов), и исключив несущественные особенности (фон). Эти алгоритмы основаны на пороговой обработке полутонового изображения, которая заключается в разделении всех пикселей изображения на два класса по признаку яркости: пиксели объекта и фона. При этом выполняется поэлементное преобразование вида

$$g(X, Y) = \begin{cases} f_w, & \text{если } f(X, Y) > f_o; \\ f_b, & \text{если } f(X, Y) < f_o, \end{cases}$$

где f_o — некоторое пороговое значение яркости (рис. 4.10).

Основной проблемой здесь является выбор порога. Очень важно выбрать правильное значение порога, чтобы убедиться, что в ходе порогового отсекаания не будет теряться полезная информация.

Пусть исходное полутоновое изображение содержит интересные нас объекты одной яркости на

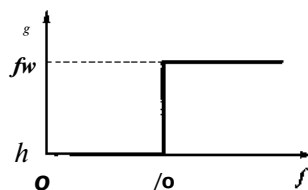


Рис. 4.10. Пороговое значение яркости изображения

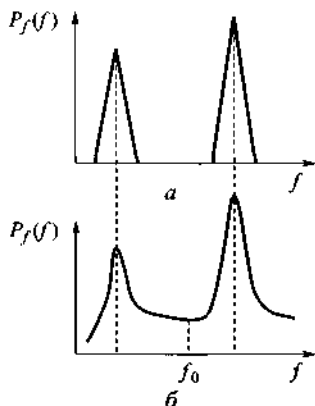


Рис. 4.11. Выбор порога преобразования:

а — для четких изображений;
б — для зашумленных изображений

фоне другой яркости (типичные примеры: машинописный текст, чертежи, медицинские снимки и т. д.). Тогда плотность распределения вероятностей яркости должна выглядеть как два узких пика, как показано на рис. 4.11, а. В этом случае задача установления порога тривиальна: в качестве f_0 можно взять любое значение между пиками. Однако обычно изображение зашумлено, кроме того, как для объектов, так и для фона характерен некоторый разброс яркостей. В результате функция плотности распределения вероятностей яркости размывается (рис. 4.11, б). Часто бимодальность распределения тем не менее сохраняется. В такой ситуации можно выбрать порог f_0 , соответствующий положению минимума между максимумами (модами). Значения элементов изображения ниже определенного значения порога превращаются в черные, в то время как значения элементов изображения, большие или равные значению порога, превращаются в белые. Пороговые алгоритмы

исключают из изображения ненужную информацию, которая может, например, нарушить процесс обнаружения краев объектов.

На уровне точечных алгоритмов можно решить задачу увеличения контраста изображения. Изображения с низким контрастом могут характеризоваться либо как в основном светлые, либо как в основном темные. Изображения с высоким контрастом, с другой стороны, имеют как темные, так и светлые области. Другими словами, изображения с высоким контрастом используют весь диапазон яркости, соответствующий полутоновым изображениям.

Изображения с низким контрастом состоят из тонов ограниченного диапазона (оттенки серого) и обычно либо слишком яркие, либо слишком темны. На гистограммах это условие регистрируется, когда все образцы изображения группируются друг с другом и занимают только малую часть возможных значений яркости изображения. Если эта группировка значений элементов изображения направлена в левую сторону гистограммы (по направлению к низким значениям яркости), изображение будет черным. Если эта группировка направлена к большим значениям яркости изображения, изображение будет ярким. Гистограммы изображений с хорошим контрастом регистрируют относительно однородное распределение значений яркости элементов изображения без больших пиков или впадин. Как и изображения с хорошим контрастом, изображения с высоким контрастом содержат широкий диапазон серых оттенков. Однако изображения с высоким контрастом имеют большие площади, на которых преобладает темный цвет, и большие площади, имеющие светлую окраску. Основным методом увеличения контраста изображения является использование гистограммы или функции плотности распределения яркости пикселей изображения.

В малококонтрастных изображениях реальный диапазон яркости оказывается намного меньше допустимого (градационной шкалы яркости). Значение яркости исходного изображения обозначим $f(X)$, а значение яркости результирующего изображения — $g(X)$. Допустимый диапазон яркости заключен между значениями f_b и f_w . Задача повышения контраста заключается в растягивании диапазона яркости исходного изображения (рис. 4.12, а) на всю шкалу (рис. 4.12, б).

Эту задачу можно решить при помощи точечного преобразования яркости по формуле

$$g(X) = af(X) + b,$$

где a и b — постоянные.

Алгоритмы увеличения контраста используют гистограммы для того, чтобы определить, где находится группировка значений элементов в изображении с низким контрастом. Для увеличения контраста сканируется гистограмма с самого нижнего значения яркости пикселей до самого высокого и от самого высокого верхнего значения яркости изображения до самого низкого. Затем находятся постоянные a и b и корректируются значения яркости пикселей изображения. В результате значения яркости между двумя порогами преобразуются в полный диапазон яркостей. Результатом этого процесса будет более контрастное изображение.

Пространственные алгоритмы. Пространственные алгоритмы используют группы пикселей изображения для извлечения информации об изображении. Этим пространственные процессы отличаются от точечных, которые основаны на информации только для одного пиксела при реализации точечного алгоритма. Группа пикселей изображения, используемых в пространственных алгоритмах, называется областью примыкания. Область примыкания — двухмерная матрица значений яркости пикселей изображения, которая обычно имеет нечетное число строк и столбцов. Преобразуемые пиксели (старое значение которых заменяется на новое как результат работы алгоритма) обычно расположены в центре области примыкания.

Большинство пространственных алгоритмов используют пространственные фильтры. В качестве примера рассмотрим задачу повышения резкости изображения с помощью пространственного фильтра. При вводе в компьютер изображения подвергаются действию нескольких искажающих факторов. Обычно эти искажения заключаются в ослаблении верхних пространственных частот спек-

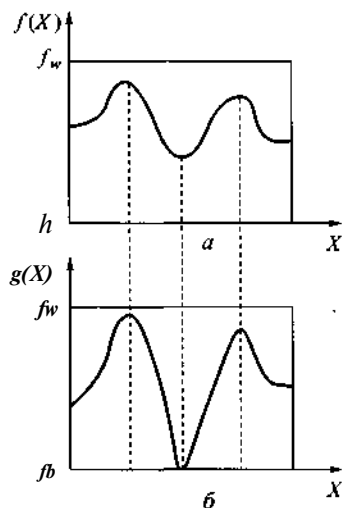


Рис. 4.12. Повышение контраста изображения растяжением диапазона яркостей:

a — диапазон яркости исходного изображения; b — диапазон яркости изображения с увеличением контрастности

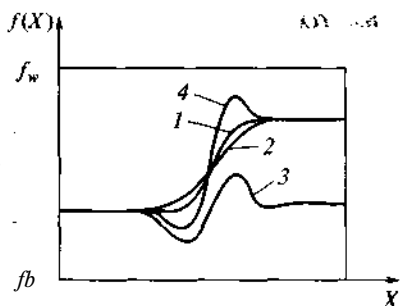


Рис. 4.13. Повышение резкости с помощью высокочастотной фильтрации:

1 — функция яркости исходного изображения; 2 — функция яркости сглаженного изображения; 3 — функция яркости разностного изображения; 4 — функция яркости результирующего более резкого изображения

ра изображения. Визуально они воспринимаются как ухудшение резкости изображения, при которой становятся плохо видимыми мелкие детали. Следовательно, повышение резкости должно заключаться в подъеме уровня высоких частот спектра изображения, т. е. в его высокочастотной фильтрации. В результате этой фильтрации происходит подчеркивание границ объектов, улучшается различимость мелких деталей (ранее размытых). Повышение резкости заключается в усилении высокочастотных составляющих пространственного спектра изображения. Простой метод повышения резкости основан на пространственной линейной обработке пикселей изображения скользящим окном небольшого размера. Это окно перемещается по изображению, и при каждом его положении формируется значение яркости выходного поля яркости (обычно это

значение соответствует центру окна). В данном случае алгоритм повышения резкости реализуется как двухмерный фильтр с конечной импульсной характеристикой. Размеры и форма окна определяют область ненулевых значений импульсной характеристики фильтра. Рассмотрим, как для одномерного случая качественно строится фильтр, подчеркивающий границы. Пусть $f(X)$ — функция яркости вдоль строки исходного нерезкого изображения. На рис. 4.13 кривая 1 представляет собой функцию яркости исходного изображения в области расфокусированной границы объекта.

Процедуру обработки можно разбить на несколько шагов. Сначала осуществляется низкочастотная фильтрация, т. е. дополнительное сглаживание сигнала (обозначим сглаженный сигнал $f_1(X)$, рис. 4.13, кривая 2). Далее из исходного вычитается сглаженный сигнал, в результате чего формируется разностный сигнал — высокочастотное изображение (рис. 4.13, кривая 3): $f_2(X) = f(X) - f_1(X)$. Затем этот разностный сигнал суммируется (с некоторым коэффициентом) с исходным. Полученный результат $g(X)$ — изображение с повышенной резкостью (рис. 4.13, кривая 4). В спектре этого изображения низкочастотные компоненты не изменились (т. е. общий уровень яркости остался прежним), а высокочастотные усилились (т. е. подчеркнуты локальные особенности — границы, мелкие детали). Рассмотрим эту процедуру для двухмерного случая. Низкочастотная фильтрация (сглаживание) осуществляется усреднением значения яркости в центре окна:

$$f_1(X, Y) = \sum_{(K_1, K_2) \in D} h(K_1, K_2) f_1(X - K_1, Y - K_2),$$

где D — некоторая конечная область в пространстве аргументов X, Y , определяющая окно $((K_1, K_2) \in D)$.

Очевидно, что записанное выражение задает двухмерную свертку дискретного сигнала с импульсной характеристикой $h(K_1, K_2)$ сглаживающего фильтра. Сначала выбираются значения $h(K_1, K_2)$ так, чтобы получить сглаживание (усреднение) значений яркости пикселей. К процедуре сглаживания предъявляется требование — она не должна изменять среднее значение (постоянную составляющую, среднюю яркость) изображения, т. е. необходимо выполнение следующего условия:

$$\sum_{(K_1, K_2) \in D} h(K_1, K_2) = 1.$$

Часто все коэффициенты импульсной характеристики берутся одинаковыми, при этом получается усреднение значений яркости изображения по окну.

Далее вычисляются высокочастотное изображение и изображение с повышенной резкостью, в результате можно получить выражение в виде свертки:

$$g(X, Y) = \sum_{(K_1, K_2) \in D} h(K_1, K_2) \cdot f(X - K_1, Y - K_2),$$

где $h(K_1, K_2)$ — импульсная характеристика фильтра, осуществляющего повышение резкости.

На практике из соображений простоты берут обычно центрированное квадратное окно малого размера (3x3 или 5x5). При этом $h(K_1, K_2)$ удобно задавать в форме так называемой маски свертки фильтра. Например, типичная маска размером 3x3 для повышения резкости изображений имеет вид

$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{pmatrix}.$$

Обозначим P_{ij} значение яркости некоторого пиксела в окне преобразования

$$\begin{pmatrix} P_0 & P_1 & P_2 \\ P_3 & P_4 & P_5 \\ P_6 & P_7 & P_8 \end{pmatrix},$$

а K_{ij} — значение соответствующего коэффициента в маске свертки

$$\begin{pmatrix} K_0 & K_1 & K_2 \\ K_3 & K_4 & K_5 \\ K_6 & K_7 & K_8 \end{pmatrix}.$$

В результате работы алгоритма вычисляется новое значение яркости центрального пиксела в окне, т. е. P_4 :

$$g(P_4) = \sum_i P_i K_i.$$

Размеры и структура маски свертки и значения коэффициентов, содержащихся в ядре свертки, определяют тип пространственного алгоритма и применяются к данным изображения. Изменение весового фактора в пределах ядра свертки влияет на величину и, возможно, на знак общей суммы и, таким образом, воздействует на значение рассматриваемого пиксела изображения.

Большой размер маски повышает гибкость процесса свертки. К сожалению, некоторые детали простого расчета весовой суммы свертки усложняют его реализацию. Первая и основная трудность связана с кромками изображения. По мере того как мы перемещаем маску свертки (и соответственно рассматриваемый пиксел изображения) по изображению пиксел за пикселем, возникают сложности, поскольку на границе изображения рассматриваемый пиксел не имеет соседей со всех сторон. Другими словами, маска свертки выходит краем за границы изображения. Такое искажение происходит на верхней, правой, левой и нижней границах изображения. Для устранения этого явления используют два метода:

- данные на краях изображения не учитываются;
- на краях изображения можно копировать пиксели маски для того, чтобы синтезировать дополнительные данные границы.

Рассмотрим другие примеры применения алгоритмов свертки. Низкочастотные пространственные фильтры оставляют низкочастотные компоненты изображения нетронутыми и ослабляют высокочастотные компоненты. Такие фильтры используются для понижения визуального шума, содержащегося в изображении, а также для удаления высокочастотных компонентов из изображения с тем, чтобы можно было тщательнее исследовать содержание низкочастотных компонентов. По мере уменьшения содержания высокочастотных компонентов, могут быть идентифицированы более слабые изменения низкой частоты. Частота отсечки низкочастотного фильтра определяется размером и коэффициентами масок свертки. Ниже приведены три различные низкочастотные маски:

$$\begin{pmatrix} \sqrt{9} & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{pmatrix}; \quad \begin{pmatrix} 1/10 & 1/10 & 1/10 \\ 1/10 & 1/5 & 1/10 \\ 1/10 & 1/10 & 1/10 \end{pmatrix}; \quad \begin{pmatrix} 1/16 & 1/8 & 1/16 \\ 1/8 & 1/4 & 1/8 \\ 1/16 & 1/8 & 1/16 \end{pmatrix}.$$

Сумма значений коэффициентов для всех низкочастотных фильтров равна единице. Рассмотрим часть изображения, не содержащую высокочастотных компонентов. Это означает, что все значения элементов изображения постоянны или они изменяются очень медленно. Когда низкочастотная маска проходит через область изображения, новое значение яркости преобразуемых элементов изображения вычисляется как сумма коэффициентов маски, умноженных на

значения пикселей изображения в области примыкания. Если все значения яркости пикселей изображения в области примыкания одинаковые (постоянные), новые значения яркости пикселей изображения будут такими же, как и старые. По этой причине сумма коэффициентов выбрана равной единице, при этом содержание низкочастотных компонентов сохраняется. По мере того как маска движется по области изображения с содержанием высокочастотных компонентов, любые быстрые изменения яркости усредняются оставшимися пикселями изображения в области примыкания, тем самым понижая содержание высокочастотных компонентов. Визуальным результатом низкочастотной фильтрации является слабая нерезкость изображения.

Высокочастотные фильтры выделяют высокочастотные компоненты изображения, оставляя содержание низкочастотных компонентов нетронутым. Содержание низкочастотных компонентов увеличивается по отношению к содержанию высокочастотных компонентов. Области изображения с высокой частотой будут хорошо освещены (станут ярче), а части с низкой частотой станут черными. Иногда резкость изображения увеличивается после высокочастотной фильтрации за счет выделения шума изображения. При использовании высокочастотной фильтрации возможно усиление края изображения. Ниже приведены две различные высокочастотные маски:

$$\begin{pmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{pmatrix}; \begin{pmatrix} 1 & -2 & 1 \\ -2 & 5 & -2 \\ 1 & -2 & 1 \end{pmatrix}.$$

Большое значение коэффициента центра маски определяет действие высокочастотного фильтра. По мере того как центр маски с большим значением перемещается по области изображения с высокой пространственной частотой (что означает большие изменения яркости элементов изображения), новое значение рассматриваемых элементов изображения многократно увеличивается. Меньшие отрицательные значения коэффициента маски, сгруппированные вокруг центра, уменьшают эффект большого весового фактора. В конечном итоге большие изменения яркости элементов изображения усиливаются, а области постоянной яркости элементов изображения останутся неизменными. Другими словами, области постоянной яркости элементов изображения (области низких частот изображений) не подвергаются этим преобразованиям.

Отдельно следует рассмотреть медианные фильтры. С помощью этих фильтров выполняется усредненное фильтрование. Усредненное фильтрование — пространственный алгоритм, который не попадает под категорию свертки. Усредненное фильтрование использует значение пикселей, содержащихся в области примыкания, для определения нового значения рассматриваемого пикселя. Однако новое значение яркости пикселя не вычисляется алгоритмически из значений яркости пикселей в области примыкания. Вместо этого пиксели в области примыкания сортируются в возрастающем порядке и отбирается среднее значение.

ние яркости как новое значение рассматриваемого пиксела. Например, надо изменить значение яркости центрального пиксела в следующем окне преобразования:

$$\begin{pmatrix} 27 & 22 & 29 \\ 42 & 255 & 29 \\ 27 & 25 & 29 \end{pmatrix}.$$

Значения яркости пикселей, отсортированные в возрастающем порядке, будут иметь вид

$$22, 25, 27, 29, 29, 29, 42, 255.$$

В результате среднее значение яркости 29 заменяет 255 в выходном изображении. Если 255 было значением пиксела шума, создавшего белое пятно на исходном изображении, то оно будет отфильтровано.

Результатом усредненного фильтрования является то, что любой случайный шум, содержащийся в изображении, будет эффективно устранен. Это происходит потому, что любое случайное резкое изменение в яркости пиксела в пределах области примыкания сортируется, т. е. оно помещается либо на вершину, либо на нижнюю часть отсортированных значений области примыкания. ч

Алгоритмы геометрических преобразований. Алгоритмы геометрических преобразований растровых данных изображений изменяют пространственное местоположение и/или структуру пикселей в изображении, основываясь на некоторых геометрических преобразованиях. Геометрические преобразования не обязательно изменяют значения яркости элементов изображения, но они всегда изменяют положение пикселей изображения. Другими словами, значения яркости для данных пикселей изображения перемещаются в новую позицию. В этом состоит основное отличие алгоритмов геометрических преобразований от точечных и пространственных алгоритмов. Каждый из этих классов алгоритмов обработки изображения всегда изменял значение яркости пикселей изображения, в данных алгоритмах яркость изменять необязательно, хотя во многих алгоритмах яркость пикселей изменяется. Геометрические преобразования растровых данных широко применяются:

- как предварительный шаг в подготовке изображения для последующей точечной, пространственной обработки;
- для обеспечения работы алгоритмов распознавания, например после сканирования страниц текста с некоторым поворотом строк текста относительно горизонтали;
- для обеспечения работы алгоритмов наложения текстуры;
- для обеспечения различных эффектов в изображениях.

Геометрические преобразования растровых данных изображения включают перемещение, масштабирование, сдвиг и поворот изображения.

Перемещение, масштабирование и сдвиг изображения. Перемещение изображения имеет смысл, только если это изображение является подызоб-

ражением некоторого большего изображения. Предположим, что мы хотим переместить массив из n строк и k столбцов пикселей (изображение-источник) с координатами левого верхнего углового пиксела $\{a, b\}$ в новое положение с координатами левого верхнего углового пиксела $\{c, d\}$ (изображение-цель). Это преобразование очень простое; мы копируем пиксели из исходного положения до положения цели и (если хотим) заменяем все исходные пиксели, которые не являются пикселями цели, цветом фона. При условии, что надо предотвратить изменение пикселей изображения источника, когда изображение цели накладывается на него, и при условии, что четыре числа a, b, c и d — это целые числа, проблем не возникает (рис. 4.14).

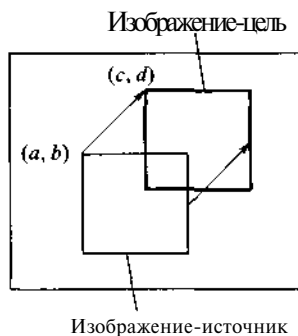
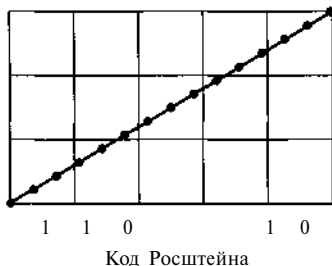


Рис. 4.14. Простое перемещение изображения

Однако если координаты начального и конечного положения не являются целыми числами, то необходимо выполнить дискретизацию координат. Аналогичная проблема имеет место и для всех других видов преобразований. Для решения этой проблемы были разработаны алгоритмы на основе кода Росштейна (Rothstein).

В качестве примера рассмотрим изменение размеров изображения только по горизонтали. Предположим, что пиксельная плоскость разделена на квадраты (один квадрат на один пиксел), и средняя яркость в квадрате имеет значение, назначенное для яркости соответствующего пиксела. Будем растягивать изображение, состоящее из серых квадратов, яркость которых определена значениями соответствующих пикселей. Обозначим коэффициент масштабирования (сдвига) plq , где p и q — простые целые числа. Тогда для растяжения пиксельной плоскости, состоящей из массива квадратов, с этим коэффициентом надо брать q столбцов квадратов изображения источника и разместить их так, чтобы охватить p столбцов квадратов изображения цели. Выполнение дискретизации по центрам квадратов полученной области даст в результате изображение цели.



Код Росштейна

Рис. 4.15. Получение кода Росштейна для линии с наклоном $qlp = 3/5$:

- — метка для определения цифр кода

Для эффективной реализации этого алгоритма на первом этапе генерируется код Росштейна для числа plq . Этот код представляет собой двоичную последовательность, описывающую линию, наклон которой равен qlp (любой сканирующий преобразователь линии может быть использован для генерации подобного кода). На рис. 4.15 показана линия с наклоном $3/5$ и с 15 (pxq) метками для фиксации пересечений этой линии с горизонтальными линиями сканирования.

Когда линия сканирования проходит слева направо, она пересекает горизонтальные линии сетки. Если столбец содержит такое пересечение сетки, он

отмечается цифрой 1; иначе он отмечается цифрой 0. Каждый столбец содержит три метки. Каждая строка содержит пять меток. Если метка принадлежит и горизонтальной, и вертикальной линии и находится в левой части столбца, столбец отмечается цифрой 1, а в правой части столбца — цифрой 0. Проверяется каждая пятая метка. Таким образом, интервал между меткой 9 и меткой 12 отмечается цифрой 1, так как метка 10 находится в пределах столбца, а интервал между меткой 12 и меткой 15 отмечается цифрой 0, так как метка 15 лежит в правой части столбца. Код Росштейна можно рассматривать как механизм для того, чтобы распределить q цифр, равных 1, равномерно среди p двоичных цифр. Поэтому его можно использовать для распределения каждого из q столбцов изображения источника среди p столбцов изображения цели.

На втором этапе после дискретизации результирующего изображения происходит назначение яркости пикселям. Эта задача имеет много возможных решений (например, метод ближайшего соседа, когда яркость пикселя результирующего изображения назначается яркости ближайшего к нему пикселя исходного изображения), но чаще всего используют фильтры с различными масками свертки. Обычно выбирают симметричную функцию фильтра, которая отлична от нуля только в пределах маски 3×3 . Всякий раз, когда квадрат маски пересекает результирующий многоугольник изображения цели, текущий пиксел считается расположенным в изображении цели. Этот квадрат называется ограничивающим квадратом для пикселя цели, а поддержка функции фильтра называется маской свертки пикселя цели (рис. 4.16).

Квадрат маски каждого пикселя цели переносится с обратным преобразованием в пространство исходного изображения, где он становится четырехугольником. Пиксеты, которые в исходном изображении лежат в этом четырехугольнике, преобразуются в изображение цели, и только те пиксеты, которые попадают в пределы ограничительного квадрата маски цели, сохраняются и используются для вычисления яркости очередного пикселя.

Поворот изображения. Поворот изображения на произвольный угол, не кратный 90° , всегда происходит с погрешностями, поскольку дискретность координатной сетки порождает нежелательные эффекты при использовании тра-

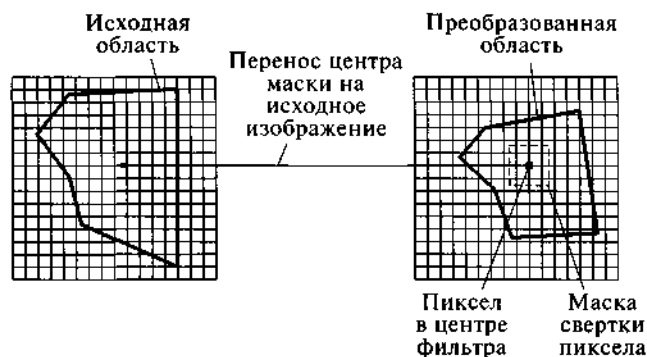


Рис. 4.16. Назначение яркости пикселу

диционных алгоритмов поворота в плоскости. Рассмотрим поворот исходного изображения вокруг начала координат на угол α против часовой стрелки. Традиционные преобразования координат для поворота точек на непрерывной плоскости запишутся следующим образом:

$$P' = AP,$$

где $P' = \begin{bmatrix} x' \\ y' \end{bmatrix}$ — координаты точки в повернутой системе координат; $P = \begin{bmatrix} x \\ y \end{bmatrix}$ —

координаты в исходной системе координат; $A = \begin{bmatrix} \cos \alpha & \sin \alpha \\ -\sin \alpha & \cos \alpha \end{bmatrix}$ — матрица поворота.

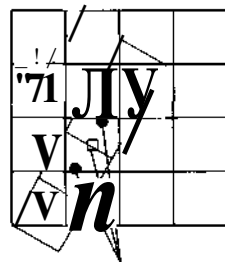
Операция поворота пикселей должна выглядеть следующим образом: на основании значений яркости пикселей исходного изображения необходимо получить значения яркости пикселей с такой же координатной сеткой для повернутого изображения. При решении задачи будем основываться на методе реализации поворота непрерывного изображения. В отличие от поворота на непрерывной плоскости необходим переход к дискретным координатам:

$$P' = \begin{bmatrix} x' \\ y' \end{bmatrix} \rightarrow P_{ij} = \begin{bmatrix} x_i \\ y_j \end{bmatrix},$$

где x_i и y_j — целые числа. Однако для произвольного угла α функции \sin и \cos не являются целыми, поэтому необходимо округлять $\begin{bmatrix} x' \\ y' \end{bmatrix}$ до ближайшего целого.

Рассмотрим действие такого алгоритма. На рис. 4.17 приведен пример поворота столбца пикселей на некоторый угол и результат дискретизации положения повернутых пикселей. На основании значений яркости в узлах исходной координатной сетки необходимо получить значения яркости в узлах результирующей координатной сетки. Следовательно, число пикселей дискретизированного изображения не должно изменяться при повороте: одному пикселу исходного изображения должен соответствовать один пиксел результирующего. Однако в данном случае такое условие не выполняется. На рис. 4.17 видно, что существуют узлы, которым соответствует более одного пиксела исходного изображения (сдвоенные точки) и узлы, которым не соответствует ни одного пиксела исходного изображения (необработанные точки). Пусть a — шаг исходной (следовательно, и результирующей) сетки. Тогда зона пиксела будет иметь вид квадрата со стороной a . Значение наибольшего отрезка, который можно вписать в квадрат, равно его диа-

Необработанная точка



Сдвоенная точка

Рис. 4.17. Округление координат пикселей при повороте:

• — центр пиксела исходного изображения; • — центр пиксела повернутого изображения

гонали, т. е. $a\sqrt{2}$. А поскольку сторона исходной координатной сетки также равна a , то возможен случай, когда один из отрезков, образованных координатной сеткой, целиком попадает в зону какого-либо пиксела. Это будет соответствовать появлению «сдвоенной» точки. Таким образом, поворот дискретизированного изображения при помощи алгоритма с округлением неизбежно приводит к появлению необработанных точек на изображении, что вносит искажения в исходное изображение.

Возможен следующий способ устранения этого эффекта. Поскольку операция преобразования координат выполняется для каждой точки исходного изображения, то число пар координат точек до обработки и после одинаково, однако на изображении присутствуют необработанные точки. Их наличие обусловлено тем, что нескольким точкам исходного изображения ставится в соответствие единственная точка повернутого изображения, что и приводит к появлению необработанных точек. Причиной этого является то, что исходная и результирующая координатные сетки имеют одинаковый шаг a . Наличие сдвоенных точек является следствием неравенства $a < a\sqrt{2}$. Для устранения этого эффекта можно использовать различные шаги исходной и результирующей сеток путем введения промежуточной системы координат. Для любой точки промежуточной системы координат вычисляются соответствующие точки результирующей и исходной системы координат. В соответствии с чем производится присваивание значения яркости точкам результирующей системы координат. Таким образом, для построения взаимосвязи между исходной и повернутой системой координат, вводится промежуточная система координат с шагом координатной сетки в 2 раза меньшим. Это можно рассматривать как продвижение по исходному изображению с нецелым шагом, меньшим, чем шаг координатной сетки. Получаемый результат не требует никакой корректировки и выполняется за один проход. Однако это связано с увеличением вычислительных затрат, поскольку поворот производится для сетки с меньшим шагом, число узлов в которой в 2 раза больше, чем в исходной.

Более экономичными являются многопроходные алгоритмы геометрических преобразований растровых данных изображений. Предположим, что имеется исходное изображение, показанное на рис. 4.18, *а*. Выполним преобразование сдвига только для вертикальных рядов пикселей (рис. 4.18, *б*) и затем повторим аналогичный сдвиг только для горизонтальных рядов (рис. 4.18, *в*).

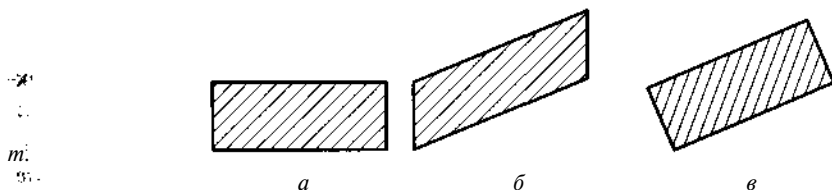


Рис. 4.18. Поворот изображения за два прохода:

а — исходное изображение; *б* — сдвиг по вертикали; *в* — сдвиг по горизонтали

Если преобразования вертикального и горизонтального сдвига выполнены правильно, то в результате можно повернуть изображение, как показано на рис. 4.18, в. Двухпроходный метод быстрее, чем прямое применение формул поворота в однопроходных алгоритмах, так как этот метод оперирует только одним рядом вертикальных или горизонтальных пикселей в одном проходе, и вычисления в пределах каждого ряда могут быть выполнены инкрементно. Следует отметить, что во многих случаях для устранения нежелательных эффектов дискретизации необходима фильтрация изображения, которая также может быть выполнена отдельно для горизонтальных и вертикальных рядов пикселей.

Выполнение двух проходов (или многих проходов) преобразования может быть разделено на две подзадачи:

- определение преобразований для индивидуальных проходов по вертикальным и горизонтальным рядам пикселей;
- применение фильтрации для устранения эффектов дискретизации результатов алгебраических вычислений при генерации новых пикселей.

Идеи многопроходных алгоритмов можно использовать и в других алгоритмах обработки растровых данных изображений.

4.1.2. Входные данные растровые, выходные данные векторные

Иногда преобразования растровых данных в векторные называют векторизацией. Алгоритмы данного класса нашли применение в системах обработки изображений и распознавания образов. Одновременно с теоретическими разработками алгоритмов векторизации уделялось большое внимание разработке проблемно-ориентированных практических систем. Наибольшие успехи были достигнуты при создании систем распознавания печатного текста, которые обеспечивают почти 100%-ное правильное автоматическое распознавание при качественном сканировании текста, например программная система FineReader. Ввиду чрезвычайной сложности задач выделения двумерных объектов и связей между ними на основе растровых бинарных или полутоновых изображений этих объектов, все системы имеют строгую предметную направленность. Кроме систем распознавания текста следует выделить системы:

- распознавания чертежных документов в САПР;
- технического зрения;
- распознавания географических карт;
- распознавания топологии интегральных схем;
- распознавания аэро- и космических снимков;
- обработки медицинских снимков;
- идентификации фотографий лица человека;
- системы идентификации отпечатков пальцев.

Области применения алгоритмов данного класса постоянно увеличиваются в связи с расширением использования растровых, сканирующих устройств вво-



Рис. 4.19. Векторизация:
а — вручную; *б* — полуавтоматический

да графической информации. Отметим основные принципы обработки растровых данных, используемые в подобных системах:

- исходное растровое изображение всегда предварительно обрабатывается для удаления случайных помех и элементов изображения, не представляющих интерес для конкретной задачи. Методы и алгоритмы для такой обработки растровых изображений были рассмотрены в подразд. 4.1.1;
- предусмотрено три основных режима работы с изображением:
 - ручной, при котором опорные точки векторного изображения и векторные контуры изображения наносятся и рисуются поверх пиксельного изображения вручную с помощью какого-либо графического редактора, как показано на рис. 4.19, *а*;
 - полуавтоматический, при котором можно вручную контролировать процесс распознавания, а также контролировать выделение областей для распознавания изображения; на рис. 4.19, *б* показан результат полуавтоматического выделения областей в программе Photoshop по команде пользователя;
 - полностью автоматический.

В алгоритмах векторизации реализуются четыре основных принципа распознавания изображений:

- последовательное распознавание от простых элементов изображения к сложным, от анализа бинарного представления к анализу полутонового;
- последовательный переход от локального анализа элементов изображения к глобальному анализу;
- максимальное использование пространственно-логических отношений между элементами различных уровней представления графической информации;
- обязательный учет специфики решаемой задачи.

Рассмотрим более подробно алгоритмы векторизации на примере распознавания технических чертежей в САПР, представленных в виде бинарных растровых изображений. Чертежом называют документ, содержащий изображение предмета и другие данные, необходимые для его изготовления и контроля. Чертеж обычно содержит проекции предмета, которые в зависимости от их содержания делятся на виды, разрезы, сечения и сведения, необходимые для его изготовления (рис. 4.20).

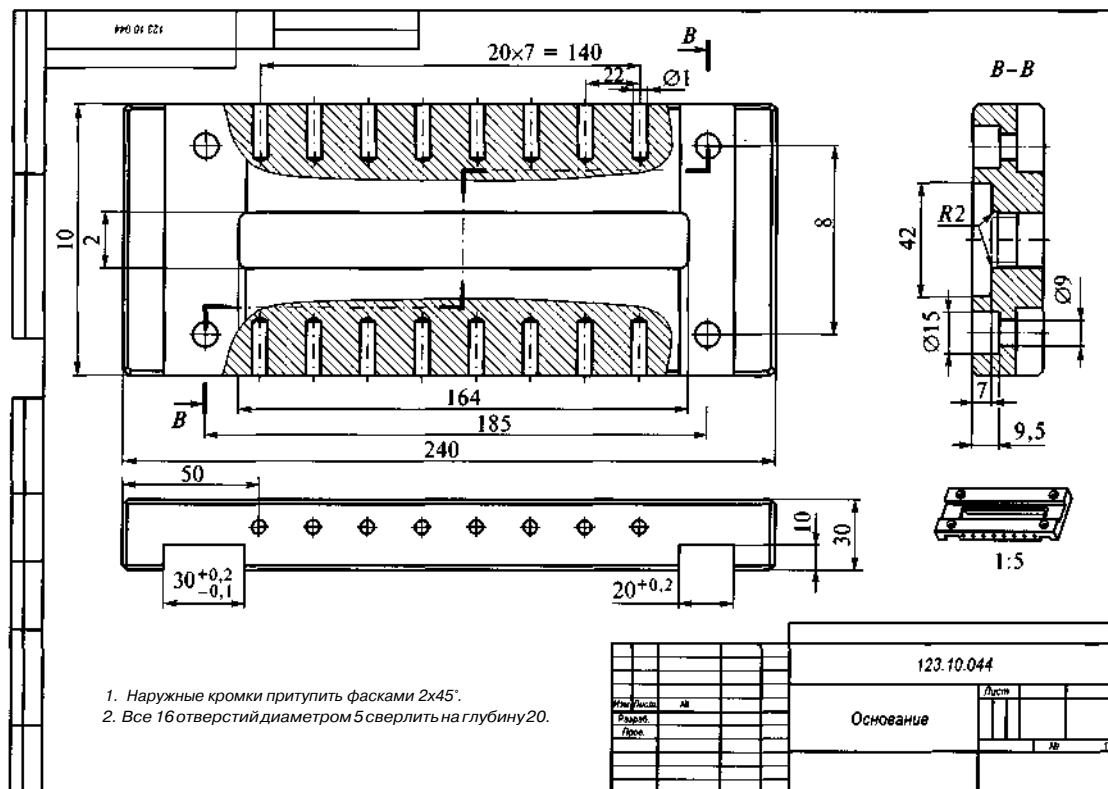


Рис. 4.20. Пример растрового изображения чертежа

Ручной ввод технических чертежей с помощью дигитайзеров очень медленный и дорогой процесс. Использование для этих целей сканеров позволяет быстро получить растровый формат изображений, однако необходимо иметь программы для преобразовывания изображения в векторную форму и затем в геометрические модели достаточно высокого уровня.

Основными элементами каждого чертежа являются линии, размеры, различная дополнительная информация. Каждый из этих элементов, в свою очередь, может быть представлен как составленный из различных частей, называемых примитивами. Основные примитивы приведены на рис. 4.21.

Цель распознавания изображений чертежей заключается в автоматическом формировании описания чертежа в примитивах, использующихся в системах САПР и которыми оперирует пользователь. Основными элементами чертежа считаются контурные (обычно утолщенные) линии, линии симметрии (штрихпунктирные линии), невидимые линии контура (обычно штриховые линии), сечения (заштрихованные области), размеры различных типов (линейные, радиальные и т. п.), обозначения допусков и подобные им элементы, которые характеризуются своей формой. Эти элементы в дальнейшем могут использоваться как основа для понимания чертежа, т. е. для его описания в терминах двумерных объектов, характерных для конкретной области применения, содержащихся в специальных библиотеках САПР (например, болты, валы, штифты и т. п.), а также для восстановления трехмерных моделей деталей. Следует отметить, что формируемый уровень описания чертежа в терминах простейших графических примитивов недостаточен для современных САПР, и объем дальнейших доработок по доведению описания до нужного уровня остается весьма существенным. Автоматическое преобразование чертежно-графических изображений в формат САПР можно разделить на три основных этапа, не считая сканирования документа и получения растрового изображения:

- векторизация изображения для получения векторного представления в терминах простейших графических примитивов;
- распознавание векторной модели с целью получения описания чертежа в терминах универсальных элементов чертежа (примитивов чертежа);
- интерпретация описания для представления специализированных двумерных объектов, имеющих в библиотеках САПР, с параметрами и отношениями, как правило, в формате DXF, который будет рассмотрен в подразд. 4.3.2.

Существующий уровень теории распознавания изображений не позволяет полностью автоматизировать процесс распознавания чертежа, т. е. обойтись без вмешательства человека. Поэтому основная цель, которая должна быть достигнута, — минимизация объема рутинного ручного труда по кодированию графической информации.

Будем рассматривать процесс распознавания как последовательное преобразование графических данных от одного уровня представления к другому, имеющему более высокий уровень обобщения информации об исходном изображении. Конечная цель в данном случае достигается путем последовательного



Рис. 4.21. Примитивы чертежа

обобщения данных до уровня, необходимого для представления чертежа в системах САПР.

Рассмотрим в качестве примера три уровня представления графической информации: исходное растровое изображение, промежуточное контурное представление и промежуточное скелетное представление. Единственным элементом исходного растрового представления является пиксел, который может принимать два значения: либо 1 (элемент объекта), либо 0 (элемент фона). Пример объекта исходного бинарного изображения приведен на рис. 4.22, а.

Растровое изображение характеризуется максимальной детализацией по сравнению с другими уровнями представления. Первым уровнем преобразования исходного растрового представления в векторное является контурное представление. Контурная форма представления (С-форма) содержит координаты внешних и внутренних границ (контуров) связанных компонентов изображения, а также параметры самих связанных компонентов (площадь, периметр, габариты и т. п.). Контурная форма бинарного объекта изображения приведена на рис. 4.22, б. Эта форма достаточно просто и однозначно может быть получена из растрового изображения. Такое представление весьма далеко от требуемого выходного представления и недостаточно структурировано, однако его удобно использовать для извлечения некоторых элементов чертежа (таких, как изолированные символы, потенциальные стрелки размерных линий, заполненные области и т. п.).

Скелетная форма представления (5-форма) используется как основа для распознавания элементов изображения. Она формируется в результате векторизации утоньшенного изображения, которое представляет собой множество растровых линий единичной ширины и содержит предпосылки для структуризации связанных компонентов изображения. Скелетная форма бинарного объекта изображения приведена на рис. 4.22, в. 5-форма содержит два основных типа данных, используемых для дальнейшего распознавания: отрезки или сегменты (части связанных компонентов утоньшенного изображения, ограниченные концевыми

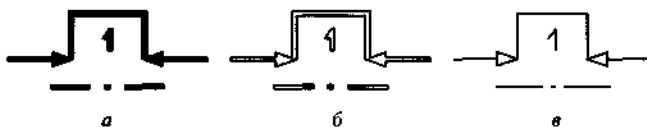


Рис. 4.22. Пример исходного бинарного (а), контурного (б) и скелетного (в) представлений

и узловыми точками) и узлы, описывающие связи между отрезками. На физическом уровне 5-форма представляется в виде трех файлов: файла паспортов, файла метрики и файла связей. В файле паспортов хранится описание каждого сегмента в виде ограниченного множества характеристик (длина; средняя ширина; угол наклона; координаты описываемого прямоугольника; тип сегмента: прямая, ломаная, дуга; кривизна для дуг окружностей; ссылки на файлы метрики и связей и др.). В файле метрики хранится метрическое описание сегментов в виде координат точек. В файле связей содержится информация о связях сегментов, стыкующихся в узловых точках. Все файлы взаимосвязаны между собой посредством ссылок. Хотя данное представление более близко к требуемому выходному представлению, оно еще остается чрезмерно детализированным (например, в скелетной форме один отрезок прямой может быть представлен в виде множества отрезков) и требует большого объема интерактивной работы для практического использования в САПР. Для достижения практически полезных результатов надо распознавать более сложные примитивы и объекты.

Собственно процесс векторизации растровых данных (после предварительной обработки) включает в себя следующие этапы:

- сегментация бинарных растровых данных или выделение областей;
- выделение контуров;
- выделение средних линий объектов;
- векторизация скелетных изображений;
- полигональная аппроксимация.

Сегментация бинарных растровых данных. Предполагается, что на документах (чертежах или описаниях) существуют блоки информации двух типов: текста и графики, размещенных, как правило, горизонтально и вертикально. Одной из первых задач (после фильтрации шумов, которая необходима для всех типов исходных растровых данных) в технологии векторизации является задача сегментации областей изображений на текст и графику.

Рассмотрим три метода сегментации изображения документа на компоненты: расширения, выделения профилей и преобразование Хоха (Hough).

Метод расширения включает просмотр черно-белых изображений. Алгоритм заменяет белые элементы изображения на черные между двумя любыми черными элементами, если расстояние между ними меньше, чем некоторый порог. Другие белые элементы изображения остаются неизменными. Алгоритм применяется сначала к строкам, а затем к столбцам. Два результирующих бинарных изображения объединяются путем применения логического И к каждому пикселу. Пороги в направлениях X и Y необязательно одинаковы. Сегментация приводит к расширению каждого блока, который будет состоять из одного типа данных (текст или графика).

Метод выделения профилей основан на том, что содержимое документа первоначально компоновалось в виде прямоугольных блоков, и изображение может быть рекурсивно разрезано на прямоугольные блоки. В результате документ представляется в форме XF-дерева с гнездами прямоугольных блоков.

Применение метода основывается на очертаниях фрагментов изображений. ХУ-дерево получается в результате использования горизонтального или вертикального разделения на альтернативных уровнях дерева. Корень — это весь анализируемый документ и он может рассматриваться как единственное горизонтальное или вертикальное выделение. Выделенные вершины и уровни формируются в процессе альтернативного выделения. Число горизонтальных и вертикальных выделений на каждом уровне различно, поэтому итоговое дерево является бинарным.

Преобразование Хоха — метод обнаружения параметрически изображаемых форм, например прямых линий на зашумленных бинарных изображениях. Сегментация на основе преобразования Хоха основана на ряде предположений о документе. Документ обычно содержит несколько прямых линий. Таблицы содержат сплошные прямые линии, чертежи также преимущественно имеют прямые линии. Столбцы текста, как правило, разделены прямыми участками белого пространства. Текст обычно состоит из нескольких параллельных текстовых строк. Преобразование Хоха позволяет находить на бинарном изображении плоские кривые, заданные параметрически, например прямые, окружности, эллипсы и т. д. Обозначим 0 — точки фона, 1 — точки интереса. Задача преобразования Хоха состоит в выделении кривых, образованных точками интереса.

Рассмотрим семейство кривых на плоскости, заданное параметрическим уравнением:

$$F(a_1, a_2, \dots, a_n, X, Y) = 0,$$

где F — некоторая функция; a_1, a_2, \dots, a_n — параметры семейства кривых; X, Y — координаты на плоскости. Параметры семейства кривых образуют фазовое пространство, каждая точка которого (для определенных значений параметров a_1, a_2, \dots, a_n) соответствует некоторой кривой.

Идея преобразования Хоха состоит в поиске кривых (точек параметрического пространства), которые проходят через достаточное число точек интереса. Ввиду дискретности машинного представления и входных данных требуется перевести непрерывное параметрическое пространство в дискретное. Для этого вводим сетку на параметрическое пространство. Каждой ячейке сетки ставим в соответствие счетчик. Значение счетчика каждой ячейки устанавливаем равным числу точек интереса, через которые проходит хотя бы одна кривая, параметры которой принадлежат данной ячейке. Анализ счетчиков ячеек позволяет найти на изображении кривые, на которых лежит наибольшее число точек интереса.

Например, прямую на плоскости (рис. 4.23) можно задать следующим образом:

$$X \cos T + Y \sin T = R,$$

где R — длина перпендикуляра, опущенного на прямую из начала координат; T — угол между перпендикуляром к

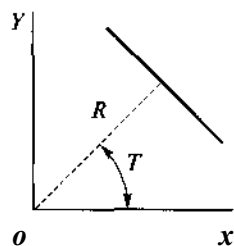


рис. 4.23. Параметрическая прямая

прямой и осью OX . Угол T изменяется в пределах от 0 до 2π , а R ограничен размерами входного изображения.

Таким образом, функция, задающая семейство прямых, имеет вид

$$F(R, T, X, Y) = x \cos T + y \sin T - R.$$

В общем случае алгоритм поиска прямой на изображении при помощи преобразования Хоха имеет следующий вид:

- 1) обнулить счетчики всех ячеек;
 - 2) для каждой точки интереса на изображении проверить каждую прямую, проходящую через данную точку, и увеличить соответствующий счетчик;
 - 3) выбрать ячейки со значением счетчика, превышающим заданный порог.
- Гистограммы расстояний вдоль линий и их перпендикуляров могут быть использованы для получения расстояний между словами и между строками. Обобщая эти данные, определяют пороги для алгоритмов группового размазывания, которые группируют вместе большие блоки данных.

Ряд алгоритмов сегментации основан на последовательном уточнении исходных изображений и чаще используется для анализа текстовых документов. Изображение сначала обрабатывается для выделения отдельных связанных компонентов. На нижнем уровне анализа выделяются отдельные символы и большие значки. Затем символы сливаются в слова, слова — в строки, строки — в параграфы, параграфы — в большие блоки, если такое слияние возможно. Этими алгоритмами обычно определяют: являются ли связанные компоненты частями текста, чертежа, областей полутоновых изображений, ограниченных порогом. Возможными характеристиками для выполнения такой классификации являются размер, ветвящиеся структуры, топология и мера формы.

Для каждой области можно подсчитать набор простейших числовых характеристик: площадь, периметр, компактность, статические моменты и т. п.

Площадь соответствует числу пикселей в области.

Периметр соответствует числу пикселей, принадлежащих границе области. При определении периметра рассматриваются два случая:

- пиксел лежит на границе области, если он сам принадлежит области и хотя бы один из его соседей области не принадлежит (внутренняя граница);
- пиксел лежит на границе области, если он сам не принадлежит области и хотя бы один из его соседей области принадлежит (внешняя граница).

Компактность — отношение квадрата периметра P к площади A ; $C = P/A$. Например, для наиболее компактной фигуры — круга — $C = 2\pi$.

Статические моменты выделенных областей определяют центр тяжести области. Например, дискретный центральный момент M_u области определяется следующим образом:

$$M_u = \sum_{(x, y) \in \text{Reg}} (x - x_c)^u (y - y_c)^v;$$

$$x_c = \frac{x}{n} \sum_{j \in \text{Reg}} x; \quad y_c = \frac{y}{n} \sum_{j \in \text{Reg}} y,$$

где n — общее число пикселей в области; Reg — выделенная область; x, y — координаты пикселей в пиксельной матрице.

На основе этих характеристик можно классифицировать получаемые области.

Графическая информация на чертежах включает три основных класса объектов: линии, области и дискретные объекты. Поскольку эти классы объектов обычно обрабатываются и распознаются различными методами, то необходимо на начальной стадии провести их сегментацию на различные классы. Данная задача может быть эффективно решена с использованием морфологических операций эрозии и расширения. Действительно, выполняя эрозию структурирующим элементом, имеющим форму круга радиуса, равного половине максимальной ширины линий плюс один, удаляют все линии на изображении. Сохранятся лишь эрозированные области и, возможно, шум. Если решается некоторая конкретная задача распознавания (например, выделение стрелок на чертежах), то, исследуя форму выделенных объектов, можно отделить области заданной конфигурации от шума. Выполнив затем операцию расширения и вычитания из исходного изображения, получают изображение, содержащее только области. При выделении областей предполагается, что известна максимальная ширина линии. Ее также можно определить либо в автоматическом, либо в интерактивном режиме.

Второй часто встречаемой при векторизации задачей является сегментация объектов на линии (протяженные объекты) и дискретные объекты. Эта задача может решаться и на растровом представлении, как это было показано выше, если дискретные объекты имеют толщину, значительно превосходящую толщину линии. В более общем случае сегментация объектов может быть выполнена на их контурном представлении.

Этапы алгоритма разделения объектов на классы по их контурному представлению:

- 1) оконтуривание всех объектов изображения и их преобразование в векторную форму;
- 2) разделение всех объектов на два класса: линейные объекты и все остальные;
- 3) выделение средних линий из линейных объектов;
- 4) полигональная аппроксимация объектов и запись информации в базу данных.

Для разделения объектов на различные классы используется средняя толщина связанного компонента, рассчитываемого на основе предположения о том, что связанный компонент состоит из множества трапеций, по следующей формуле:

$$w = \frac{P - \sqrt{P^2 - 16S}}{4}$$

где S и P — площадь и периметр связанного компонента соответственно.

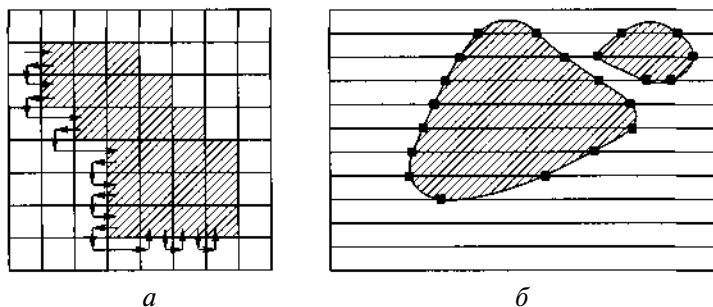


Рис. 4.24. Выделение контура отслеживающим (а) и сканирующим (б) алгоритмом

Если подкоренное выражение превосходит некоторое заданное пороговое значение, то объект является линейным. Если площадь или периметр меньше соответствующих пороговых значений, то объект представляет собой шум.

Выделение контуров. Алгоритмы выделения контуров можно условно разбить на две группы: отслеживающие и сканирующие.

Отслеживающие алгоритмы основаны на том, что на изображении отыскивается объект (первый встретившийся пиксел объекта) и контур объекта отслеживается и векторизуется. Достоинством данных алгоритмов является простота, к недостаткам можно отнести их последовательную реализацию и некоторую сложность при поиске и обработке внутренних контуров. Пример отслеживающего алгоритма — алгоритм «бегающий муравей» — приведен на рис. 4.24, а.

На первом этапе алгоритма осуществляется сканирование изображения до нахождения первого пиксела объекта (первой черной точки). Затем «муравей» начинает двигаться вокруг этой точки по 8-связной окрестности против часовой стрелки. Первой анализируемой точкой 8-связной окрестности является точка, с которой он попал в центр окрестности. Движение по 8-связной окрестности осуществляется до тех пор, пока не будет найден новый пиксел объекта. Эта точка принимается за центр окрестности и все опять повторяется. Таким образом осуществляется движение по контуру до тех пор, пока контур не замкнется, т. е. очередная точка не совпадет с первой точкой контура, с которой началось отслеживание. Затем осуществляется поиск другого объекта и отслеживание повторяется.

Сканирующие алгоритмы основаны на просмотре (сканировании) всего изображения и выделения контурных точек без отслеживания контура конкретных объектов. На рис. 4.24, б показан пример выделения контура сканирующим алгоритмом.

Выделение средних линий объектов изображения. Задача выделения средних линий (скелетов) изображений является одной из основных задач предварительной обработки изображения. Средние линии позволяют описывать геометрические особенности объектов и удобны для последующей обработки. Термин

«утонынение» — общий термин для обозначения процесса преобразования линий или других объектов изображения, имеющих ширину в несколько пикселей, в линии единичной ширины. Другими наиболее употребляемыми терминами для обозначения этой операции являются термины «скелетизация», «преобразование средних осей», «преобразование осей симметрии» и др.

К операции утонышения предъявляются, как правило, три основных требования:

- связность объектов изображения и фона должна быть сохранена;
- концы средней линии должны располагаться как можно ближе к их истинному положению;
- центральные линии объектов должны быть выделены достаточно точно.

Как правило, все существующие алгоритмы удовлетворяют этим требованиям. Алгоритмы утонышения можно условно разбить на несколько групп на основе идеи или метода, заложенных в них. Самая большая группа алгоритмов основана на идее итеративного удаления внешних слоев или контурных точек объектов до тех пор, пока на изображении останутся только точки скелета. Итеративные алгоритмы используют маску (как правило, размером 3×3), которая перемещается по всему изображению и в каждый момент времени она сопоставляется с соответствующим участком изображения, чтобы определить новое значение центрального пиксела. Таким образом, в результате просмотра всего изображения удаляется один (или несколько) из внешних слоев объекта. Количество просмотров изображения и как следствие время работы итеративных алгоритмов зависят от максимальной ширины объектов изображения.

Алгоритмы данной группы можно разделить на два класса: параллельные и последовательные. В параллельных алгоритмах окно располагается одновременно во всех пикселах изображения и при его обработке не используются новые (полученные на данной итерации) значения пикселей. При работе последовательных алгоритмов пиксели обрабатываются последовательно.

Векторизация скелетного представления исходных растровых данных.

На данном этапе утонышенное, скелетизированное растровое изображение преобразуется в векторную структуру, сохраняющую топологию объектов исходного графического материала. Другими словами, объект (рис. 4.25, а) должен быть преобразован в соответствующий ему граф (рис. 4.25, б).

Методы получения векторного представления можно разделить на следующие группы: отслеживающие, сканирующие и их комбинация.

Комбинация сканирующих и отслеживающих методов достигается тем, что вначале в результате сканирующего просмотра изображения осуществляется его предварительная разметка. Каждый пиксел утонышенного раstra анализируется на основании его связности с восемью соседними пикселями и помечается определенным

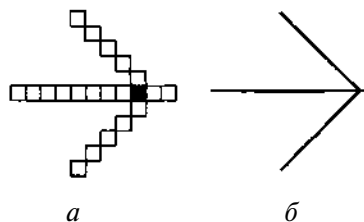


рис. 4.25. Объект скелетизированного изображения (а) и его граф (б)

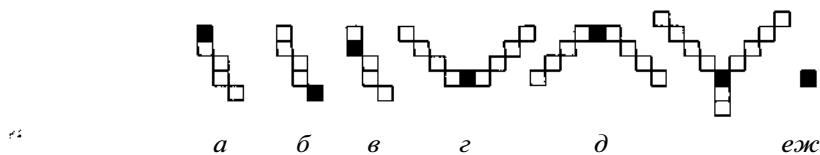


Рис. 4.26. Типы ситуаций сканирования:

а — начало; *б* — конец; *в* — продолжение; *г*, *д* — слияние; *е* — разветвление;
ж — изолированная точка

образом. Все черные (единичные) пиксели утоньшенного растра в зависимости от расположения окружающих их пикселей в определенной окрестности классифицируются как конечная точка, элемент линии и узловая точка. Из такого размеченного растра выбираются пиксели, помеченные как конечные и узловые. Их координаты запоминаются в соответствующем файле. Далее, начиная с этих точек, отслеживаются объекты изображения и запоминаются в файле объектов. При этом создается векторное описание исходного изображения. Если при первоначальном сканировании исходного растра проставить для каждого пикселя число итераций утоньшения, то в векторное линейное описание можно ввести дополнительный атрибут — ширину линии. Объединение всех связанных пикселей утоньшенного растра в линейные структуры осуществляется за один просмотр исходного файла. В результате идентифицируются все линии изображения и места их пересечений, ветвлений и слияний, а на изображении выделяются особые точки (конечные или узловые) и отрезки связных компонентов (сегменты), ограниченные особыми точками.

Векторизация выполняется за один просмотр входного изображения и состоит из двух основных операций: обработки ситуаций и отслеживания отрезков в режиме сканирования. Типы ситуаций сканирования приведены на рис. 4.26. После просмотра всего изображения выполняется полигональная аппроксимация векторизованной информации, одновременно вычисляются характеристики отрезков, такие как средняя ширина, длина, габаритные размеры и т. д.

Полигональная аппроксимация. Полученное векторное представление объектов является избыточным и должно быть сжато. Для сжатия известно много методов, которые на первом уровне можно подразделить на две группы: методы, основанные на функциях первого порядка (полигональная аппроксимация), и методы, основанные на функциях второй и более степени.

Полигональная аппроксимация является средством компактного и эффективного представления линий для анализа формы и классификации образов. Она позволяет сократить количество входных данных и сгладить незначительные искажения линии. Аппроксимация должна обладать некоторыми свойствами:

- хорошо сохранять информацию, т. е. все существенные особенности формы не должны устраняться;

*— • не занимать значительных объемов памяти;

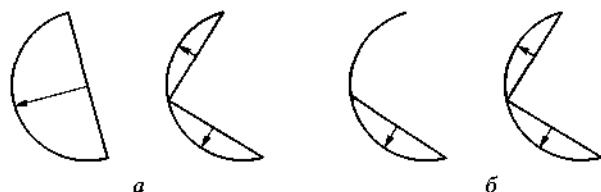


Рис. 4.27. Итерационная аппроксимация от концевых точек (а) и от одной начальной точки (б)

- быть нечувствительна к локальным шумам и преобразованиям, таким как параллельный перенос, поворот, масштабирование и т. д.

Существует много методов полигональной аппроксимации, среди которых можно выделить два наиболее известных:

- метод итерационной аппроксимации от концевых точек включает следующие основные этапы (рис. 4.27, а):

- концевые точки соединяются прямой и измеряется расстояние от прямой до максимально удаленной точки,
- если расстояние превосходит заданный порог, то линия разбивается на две и строятся две прямые, где опять измеряется расстояние,
- так продолжается до тех пор, пока все расстояния не будут меньше заданного порога;

- метод аппроксимации от одной начальной точки начинает работать от начальной точки, строит прямую от нее до ближайших точек, измеряет аналогичные расстояния, и если расстояние меньше порога, то данная точка выбирается в качестве новой точки (рис. 4.27, б).

Методы аппроксимации, основанные на функциях более высокого порядка, обычно позволяют получать более точный результат. Но они более дорогие в вычислительном отношении, чем полигональная аппроксимация. Очень часто вместе с аппроксимацией необходимо выравнивать прямые линии в местах, где они встречаются, т. е. делать почти вертикальные (горизонтальные) линии строго вертикальными (горизонтальными).

4.1.3. Входные данные векторные, выходные данные растровые

Для алгоритмов этой группы иногда используется термин «растеризация». Растеризация выполняется при выводе результатов обработки графических изображений и результатов геометрического моделирования разнообразных объектов на растровые устройства вывода, в первую очередь на графические дисплеи.

Будем рассматривать растровые данные в виде прямоугольного массива числовых значений (пиксельная карта — *pixmap*). При этом будем считать, что каждое числовое значение в пиксельной карте состоит из фиксированного числа бит, называемых глубиной цвета (*color depth*) пиксела. Если глубина цвета пик-

села равна cd , то он может принимать 2^{cd} различных значений и поэтому способен отобразить 2^{cd} различных цветов. Если $cd = 1$, то возможны только два цвета — белый — W (white) и черный — B (black) (битовая карта — *bitmap*). Векторные данные — фактически множество точек, кривых линий и замкнутых областей на плоскости. Отображение этих данных на пиксельной карте требует решения двух фундаментальных задач:

- закрашивание пикселей вдоль кривой в координатной плоскости XY пиксельной карты;
- заполнение замкнутых областей пикселями заданного цвета.

Основные фундаментальные идеи, реализованные в конкретных алгоритмах, заключаются в следующем.

1. Для определения положения каждого пиксела вдоль кривой используется простейшее уравнение кривой на плоскости:

$$Y = F(X).$$

Наклон касательной этой кривой к оси X определяется производной dF/dX и при значении этой производной больше единицы возможно влияние ошибок округления на определение положения очередного пиксела вдоль кривой, поэтому как только значение этой производной превысит единицу, следует использовать другое уравнение кривой на плоскости:

$$X = F_{\text{обр}}(Y),$$

где $F_{\text{обр}}$ — обратная функция по отношению к F .

При этом производная $dF_{\text{обр}}(Y)/dY$ не будет превышать единицу. Например, для окружности эту идею можно реализовать, определив нужные пиксели только в первой половине первого квадранта относительно центра окружности, помещенного в начало координат пиксельной плоскости, а затем использовать операции последовательного отражения и перемещения результата относительно соответствующего центра и осей координат.

2. Для определения положения очередного пиксела вдоль непрерывной кривой следует использовать информацию об уже размещенных пикселях (инкрементные алгоритмы).

3. Поскольку важнейшим требованием к конечным программам является максимальное быстродействие, то следует заменять математические операции с вещественными числами на операции целочисленной арифметики и сдвига или выбирать результат из заранее вычисленных таблиц (например, для тригонометрических функций).

4. Заполнение областей выполняется либо заливкой, начиная с известного «затравочного» пиксела внутри области, либо построчным сканированием областей, ограниченных замкнутыми кривыми линиями.

Первой фундаментальной задачей в алгоритме растеризации является задача отображения отрезка прямой линии на пиксельной плоскости XY .

Перед рассмотрением конкретных алгоритмов сформулируем общие требования к изображению отрезка:

- концы отрезка должны находиться в заданных точках;
- отрезки должны выглядеть прямыми;
- яркость вдоль отрезка должна быть постоянной и не зависеть от длины и наклона;
- алгоритм должен быть строго детерминированным (всегда отображать одни и те же пиксели для одинаковых отрезков);
- результат должен быть одинаков независимо от того, в каком направлении отображается отрезок.

Первые три требования не могут быть точно выполнены на пиксельной плоскости в силу того, что изображение строится из пикселей конечных размеров, а именно:

- концы отрезка в общем случае располагаются на пикселях, лишь наиболее близких к требуемым позициям, и только в частных случаях координаты концов отрезка точно совпадают с координатами пикселей;
- отрезок аппроксимируется набором пикселей и лишь в частных случаях вертикальных и горизонтальных отрезков они будут выглядеть гладкими прямыми без ступенек. На рис. 4.28 показано отображение пяти отрезков, проведенных из верхнего левого угла пиксельной плоскости (16×16 пикселей), полученное по алгоритму Брезенхема; на рис. 4.28, *а* — горизонтального, вертикального и под углом 45° , а на рис. 4.28, *б* — под углом больше 45° и под углом меньше 45° ;
- яркость для различных отрезков и даже вдоль отрезка в общем случае различна, так как, например, расстояние между центрами пикселей для вертикального отрезка и отрезка под углом 45° различно.

Чтобы понять основные идеи, рассмотрим самый простой (и неэффективный) метод, а затем классический алгоритм Брезенхема для растеризации отрезков прямых линий. Предположим, что мы хотим задать значения пикселей так, что отрезок прямой линии появляется между точками *a* (с координатами X_a, Y_a) и *b* (с координатами X_b, Y_b). На рис 4.29 показано, какие пиксели отображаются вдоль идеального отрезка такой прямой.

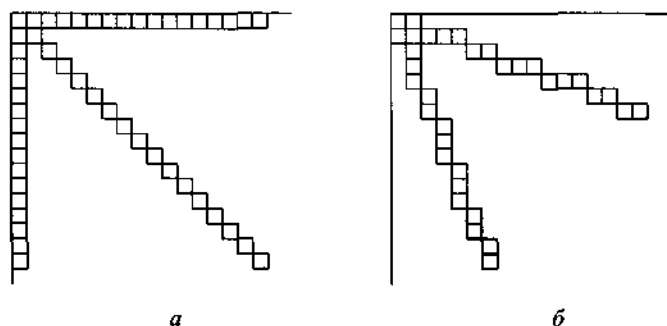


Рис. 4.28. Отображение отрезков прямых линий на пиксельной плоскости:

а — горизонтального, вертикального и под углом 45° ; *б* — под углом больше и меньше 45°

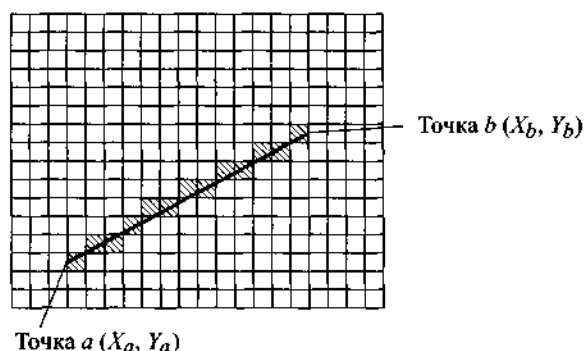


Рис 4.29. Отображение отрезка прямой между двумя точками:

■ — отображаемый пиксел

Данная прямая описывается уравнением

$$Y = mx(X - X_a) + Y_a,$$

где X изменяется от X_a до X_b , а величина mx определяет наклон прямой к оси X :

$$mx = \frac{dY}{dX} = \frac{Y_b - Y_a}{X_b - X_a}.$$

Для прямых с углом наклона больше 45° следует использовать уравнение

$$X = my(Y - Y_a) + X_a.$$

Самый простой метод растеризации заключается в том, чтобы двигаться по X от X_a до X_b с единичным (в масштабе пиксельной сетки) шагом по оси X и на каждом шаге округлять соответствующее значение Y до ближайшего целого числа. Данный алгоритм неэффективен, так как арифметические операции с вещественными числами и округление надо выполнять для каждого отображаемого пикселя.

Алгоритм Брезенхема для отображения отрезка прямой линии. Алгоритм Брезенхема (Bresenham) имеет значительные преимущества по сравнению с простым методом, поскольку в нем отсутствует арифметика с плавающей точкой и округление. Его можно также использовать для отображения окружностей и эллипсов на пиксельную плоскость, так как он является классическим примером инкрементного алгоритма, в котором вычисляется положение каждого пикселя на основе информации о предыдущем пикселе. Рассмотрим вариант этого алгоритма, известный под названием алгоритм средней точки.

Предположим, что дан отрезок прямой между точками a и b и необходимо найти наилучшую последовательность отображаемых пикселей. Для упрощения изложения алгоритма рассмотрим частный случай, когда $X_a < X_b$ и наклон прямой находится между 0 и 1. Определим ширину W и высоту H для границ отрезка прямой по X и Y :

(высота) $H = Y_b - Y_a$

$$W = X_b - X_a;$$

$$H = Y_b - Y_a.$$

В силу принятых ограничений H и W положительны и $H < W$. Уравнение прямой для данного отрезка можно переписать в следующем виде:

$$W(Y - Y_a) + H(X - X_a) = 0.$$

Левая часть этого уравнения равна нулю для всех точек (X, Y) , лежащих на прямой, если она меньше нуля, то точка (X, Y) лежит выше прямой, а если она больше нуля, то точка (X, Y) лежит ниже прямой. Поскольку в дальнейшем нужен будет только знак этого выражения, то домножим его на 2, чтобы далее использовать только целочисленную арифметику:

$$F(X, Y) = -2W(Y - Y_a) + 2H(X - X_a).$$

На рис. 4.30 показаны некоторые пиксели вблизи прямой.

Предположим известно, что для прямой наилучшим отображением будет пиксел P . Нужно определить наилучшее значение Y для следующего значения X , равного $(X_p + 1)$, т. е. хотим определить, находится ли прямая в точке $(X_p + 1)$ ближе к точке $L = (X_p + 1, Y_p)$ или к точке $U = (X_p + 1, Y_p + 1)$. На рис. 4.30 показано одно из возможных положений прямой. Решение об отображении пиксела U или L будем принимать в зависимости от того, лежит идеальная прямая выше или ниже средней точки $M = (X_p + 1, Y_p + 1/2)$, расположенной посередине между U и L . Вычислив функцию $F(X, Y)$ в точке M , по ее знаку определим, проходит прямая выше или ниже точки M :

- если $F(X_M, Y_M) < 0$, то точка M лежит выше прямой и выбираем точку L ;
- если $F(X_M, Y_M) > 0$, то точка M лежит ниже прямой и выбираем точку U .

Таким образом, если $F(X_M, Y_M) > 0$, то включаемый пиксел находится выше, чем предыдущий, поэтому надо увеличить Y на единицу, в противном случае Y оставить без изменения. Суть метода Брезенхема состоит в том, чтобы вычислять эту функцию инкрементно, основываясь на том, насколько ее значение должно измениться от одного шага к следующему.

Для точки M можно записать выражение

$$F(X_M, Y_M) = -2W(X_p + 1/2 - Y_a) + 2H(X_p + 1 - X_a). \quad (4.1)$$

Рассмотрим изменение функции $F(X, Y)$ при переходе от $X = X_p + 1$ к следующему значению $X = X_p + 2$. Как показано на рис. 4.30, следующей средней точкой M будет либо точка M' , либо точка M'' . Если мы не прибавили к Y единицу на предыдущем шаге, то точкой M станет точка $M' = (X_p + 2, Y_p + 3/2)$, а если прибавили, точкой M станет точка $M'' = (X_p + 2, Y_p + 3/2)$.

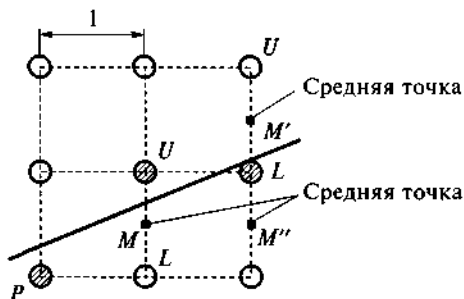


Рис. 4.30. Схема для получения алгоритма средней точки:

⊙ — центр отображаемого пиксела

Рассмотрим два возможных случая вычисления функции F .

Случай 1. Если функция P на предыдущем шаге была отрицательна, т. е. значение Y не изменялось, то получим равенство

$$F(X_p + 2, Y_p + 1/2) = -2W(Y_p + 1/2 - Y_a) + 2H(X_p + 2 - X_a).$$

Чтобы определить, насколько последнее равенство больше, чем $F(X_M, Y_M)$, вычтем из него равенство (4.1):

$$F(X_p + 2, Y_p + 1/2) = F(X_M, Y_M) + 2H.$$

Случай 2. Если функция F на предыдущем шаге была положительна, т. е. значение Y было увеличено на 1, то

$$F(X_p + 2, Y_p + 3/2) = -2W(Y_p + 3/2 - Y_a) + 2H(X_p + 2 - X_a) = F(X_M, Y_M) - 2(W - H).$$

В каждом из случаев к исходному значению функции добавляется некоторая константа: $2H$, если мы не увеличивали Y , и $-2(W - H)$ — если увеличивали.

Осталось определиться, с чего начинать процесс растеризации. Известно, что при $X = X_a$ $Y = Y_a$. Поэтому первый «экземпляр» средней точки $M = (X_a + 1, Y_a + 1/2)$ и тогда

$$F(X, Y) = -2W(Y_a + 1/2 - Y_a) + 2H(X_a + 1 - X_a) = 2H - W.$$

Если бы мы предварительно не умножили функцию на 2, то результат был бы $H - 0,5W$, что сделало бы невозможным использование целочисленных значений всех переменных.

Таким образом, перед началом процесса примем, что $F = 2H - W$, $X = X_a$, $Y = Y_a$. Затем на каждом шаге делаем следующее:

- 1) присвоим пикселу (X, Y) нужное значение цвета;
- 2) увеличим X на 1; если $F < 0$, то добавим к F значение $2H$; в противном случае увеличим Y на 1 и добавим к F значение $-2(W - H)$.

При программировании этого алгоритма следует отдельно рассматривать горизонтальные и вертикальные прямые, так как они встречаются в двухмерной графике очень часто. Для этих прямых следует использовать упрощенные алгоритмы, например для горизонтальных прямых закрашивать сразу все биты вдоль строки развертки.

Второй фундаментальной задачей в этой группе алгоритмов является задача заполнения многоугольника заданными координатами его вершин, пикселями одинакового цвета. Простейший способ заполнения многоугольника, заданного координатами вершин, заключается в определении принадлежности пиксела внутренней части многоугольника — если принадлежит, то пиксел закрашивается. Определить принадлежность пиксела многоугольнику можно, например, подсчетом суммарного угла с вершиной на пикселе при обходе контура многоугольника.

Из рис. 4.31, а следует, что если пиксел внутри многоугольника, то суммарный угол будет равен $APB + BPC + CPD + DPE + EPA = 360^\circ$, а из рис. 4.31, б следует, что если пиксел вне многоугольника, то этот угол будет равен

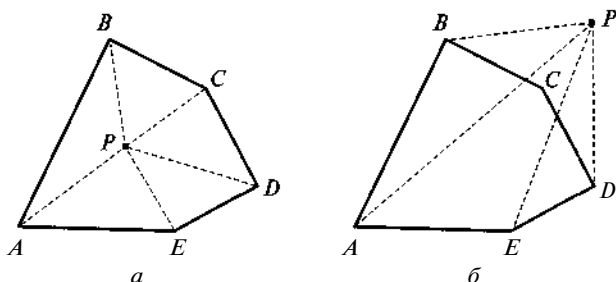


Рис. 4.31. Определение принадлежности пиксела:
а — внутри многоугольника; б — вне многоугольника

$DPE + EPC + CPA + APB - BPD = 0$, поскольку угол BPD отсчитывается в противоположном направлении по отношению к другим углам и равен их сумме.

Вычисление принадлежности в этом алгоритме надо выполнять для всех пикселей экрана, поэтому данный способ неэффективен.

Алгоритм построчного заполнения многоугольника. Более эффективны алгоритмы построчного заполнения. Предположим, что область, подлежащая заполнению, представляет собой многоугольник, заданный совокупностью пикселей $P_i = (X_i, Y_i)$, которые определяют вершины многоугольника. На рис. 4.32 приведен пример для многоугольника, заданного пятью вершинами.

Для заполнения области внутри многоугольника просматривают пиксели по строке, закрашивая нужные участки последовательных пикселей внутри многоугольника. При этом для каждой строки развертки пикселей находят пересечения этой строки с ребрами многоугольника E_i и сортируют пересечения по возрастанию X . Затем серии пикселей между всеми парами пересечений закрашивают.

На рис. 4.32 строка развертки при $Y = 3$ пересекает четыре ребра — E_3, E_4, E_5, E_2 . Четыре абсциссы точек пересечения округляются до ближайшего целого и сортируются, в результате получается последовательность координат по X закрашиваемых пикселей — 1, 2, 7, 9. Затем закрашиваются следующие серии пикселей: первая от столбца 1 до столбца 2 и вторая от столбца 7 до столбца 9. При группировании отсортированных пересечений ребер в пары используется тест — находится внутри или снаружи. В процессе движения вдоль строки развертки пикселей мы при каждом пересечении попадаем или внутрь многоугольника, или наружу от него, причем это состояние все время меняется. Пребывание внутри многоугольника называют четностью. При этом можно говорить об изменении четности при каждом пересечении. Если мы попадаем внутрь многоугольника, то эта нечет-

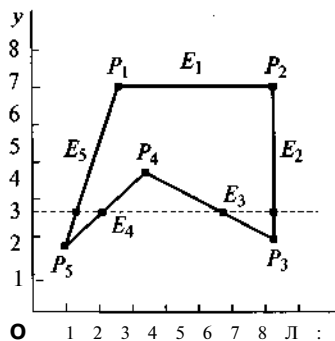


Рис. 4.32. Заполнение области
внутри многоугольника

ная последовательность пикселей будет закрашиваться; если попадаем наружу многоугольника, то четная последовательность не закрашивается. Если многоугольник целиком лежит по правую сторону от начала каждой строки развертки пикселей, то будет нечетная последовательность пикселей.

Обычно необходимо закрашивать нескольких многоугольников, причем некоторые из них (например, два) располагаются рядом и в силу этого имеют общее ребро. Это называется примыканием многоугольников, при этом в рассмотренном алгоритме возможно присвоение пикселям общего ребра сначала цвета одного многоугольника, а затем другого, следовательно, алгоритм должен решить, какому многоугольнику отнести пиксели примыкающего ребра. Общепринятое правило заключается в том, чтобы многоугольник владел своими левыми (для боковых) и своими нижними (для горизонтальных) ребрами. Тогда в случае примыкающих многоугольников ребро будет принадлежать только правому или верхнему многоугольнику. На рис 4.33 приведен пример заполнения рассмотренной выше области внутри многоугольника с учетом данного правила. Пиксели сверху и справа этого многоугольника не закрашиваются.

Часто бывает, что строка развертки проходит точно через угловую точку многоугольника. Для обеспечения правильного изменения четности это принимается за пересечение. В действительности существует много различных случаев и в программе потребуется реализовать сложный набор правил.

Практические алгоритмы основаны также на том, что соседние пиксели в строке скорее всего одинаковые и меняются только там, где строка пересекается с ребром многоугольника. Это называется когерентностью растровых строк. При этом достаточно определить координаты X пересечений строк сканирования с ребрами многоугольника. Пары отсортированных точек пересечения зададут интервалы закрашивания пикселей. Кроме того, если какие-либо ребра пересекались i -й строкой, то они скорее всего будут пересекаться также и строкой $i+1$. Это называется когерентностью ребер. При переходе к новой строке легко вычислить новую координату X точки пересечения ребра, используя координату X старой точки пересечения и тангенс угла наклона ребра:

$$X_{i+1} = X_i + \frac{1}{mx} = X_i + dX,$$

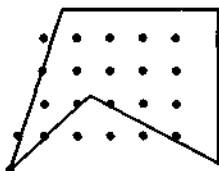


Рис. 4.33. Заполнение области с учетом примыкания:

— центр отображаемого пикселя

mx — тангенс угла наклона ребра, $mx = dy/dx$, так как $dy = 1$, то $1/mx = dx$.

Смена числа интервалов закрашивания пикселей происходит только тогда, когда в строке сканирования появляется вершина.

Учет когерентности строк и ребер позволяет построить различные высокоэффективные алгоритмы построения сканирования для заполнения многоугольников. Для увеличения эффективности любого алгоритма следует найти ту его часть, которая отнимает больше всего времени. В дан-

ном случае это большой объем вычислений, связанных с растром пересечений строк развертки с ребрами. Для снижения этих затрат времени следует построить и ввести простой список активных ребер, в результате чего можно быстро определять точки пересечений. Этот список позволяет алгоритму использовать когерентность ребер. Для каждой строки сканирования рассматриваются только те ребра, которые пересекают строку. При переходе к следующей строке для пересекаемых ребер пересчитываются координаты X пересечений. При появлении в строке сканирования вершин выполняется перестройка списка активных ребер. Ребра, которые перестали пересекаться, удаляются из списка, а все новые ребра, пересекаемые строкой, заносятся в него.

Общая схема алгоритма, динамически формирующего список активных ребер и заполняющего многоугольник снизу вверх, имеет следующий вид:

- 1) подготовить служебные целочисленные массивы Y координат вершин и номеров вершин;
- 2) совместно отсортировать координаты Y по возрастанию и массив номеров вершин для того, чтобы можно было определить исходный номер вершины;
- 3) определить пределы заполнения по оси Y — Y_{\min} и F_{\max} . Стартуя с текущим значением $Y_{\text{тек}} = F_{\min}$, исполнять пп. 4-9 до завершения закрашивания области;
- 4) определить число вершин, расположенных на строке $Y_{\text{тек}}$ — текущей строке сканирования;
- 5) если вершины есть, то для каждой из вершин дополнить список активных ребер, используя информацию о соседних вершинах;
- 6) для каждого ребра в список активных ребер заносятся:
 - максимальное значение координаты Y ребра,
 - приращение координаты X при увеличении Y на 1,
 - начальное значение координаты X .
- 7) если обнаруживаются горизонтальные ребра, то они просто закрашиваются и информация о них в список активных ребер не заносится;
- 8) если после этого обнаруживается, что список активных ребер пуст, то закрашивание заканчивается;
- 9) по списку активных ребер определяется $Y_{\text{след}}$ — координата Y ближайшей вершины (вплоть до $Y_{\text{след}}$ можно не заботиться о модификации списка активных ребер, а только менять координаты X пересечений строки сканирования с активными ребрами).

В цикле от $Y_{\text{тек}}$ до $Y_{\text{след}}$ для каждой координаты Y :

- выбрать из списка активных ребер и отсортировать координаты X пересечений активных ребер со строкой сканирования;
- определить интервалы и выполнить закрашивание пикселей;
- перевычислить координаты пересечений для следующей строки сканирования;
- проверить, не достигли ли максимальной координаты Y . Если достигли, то заполнение области закончено, иначе выполнить п. 4;

• очистить список активных ребер от ребер, закончившихся на строке $Y_{\text{след}}$ и перейти к п. 4.

Алгоритмы заполнения области с затравкой. В алгоритмах заполнения области с затравкой задается заполняемая область, код пиксела, которым будет выполняться заполнение, и начальная точка в области, начиная с которой начнется заполнение. В этом состоит основное отличие заливки области с затравкой от построчного заполнения многоугольника. В последнем случае сразу имеем всю информацию о предельных размерах части экрана, занятой многоугольником. Поэтому определение принадлежности пиксела многоугольнику базируется на быстро работающих алгоритмах, использующих когерентность строк и ребер. В алгоритмах же заполнения области с затравкой вначале надо прочесть пиксел, затем определить, принадлежит ли он области, и если принадлежит, то перекрасить.

Закрашиваемая область или ее граница — это некоторое связное множество пикселей. По способам доступа к соседним пикселям области делятся на 4- и 8-связные. В 4-связных областях доступ к соседним пикселям осуществляется по четырем направлениям — горизонтально влево и вправо и вертикально вверх и вниз. В 8-связных областях к этим направлениям добавляются еще четыре диагональных. Используя связность, мы можем, двигаясь от точки затравки, просмотреть и закрасить все пиксели области. При этом для 4-связной прямоугольной области граница 8-связная и, наоборот, у 8-связной области граница 4-связная. В общем, 4-связную область можно заполнить как 4-, так и 8-связным алгоритмом. Обратное же неверно.

С использованием связности областей и стека можно построить алгоритмы заливки гранично-определенных областей.

Рассмотрим простой алгоритм заливки гранично определенной 4-связной области:

- 1) поместить координаты затравки в стек;
- 2) пока стек не пуст, извлечь координаты пиксела из стека;
- 3) перекрасить пиксел;
- 4) для всех четырех соседних пикселей проверить, является ли он граничным или уже перекрашен. Если нет, то занести его координаты в стек и перейти к п. 2.

На рис. 4.34, *а* показан выбранный порядок перебора соседних пикселей, а на рис. 4.34, *б* — соответствующий ему порядок закрашки простой гранично определенной области.

Рассмотренный алгоритм легко модифицировать для работы с 8-связными гранично определенными областями или же для работы с внутренне определенными областями.

Недостатком алгоритмов, непосредственно использующих связность закрашиваемой области, являются большие затраты памяти на стек, так как на каждый закрашенный пиксел в стеке по максимуму будет занесена информация о еще трех соседних. Кроме того, информация о некоторых пикселях может записываться в стек многократно. Это приводит не только к перерасходу памяти, но и к потере быстродействия за счет многократной раскраски одного и того же пик-

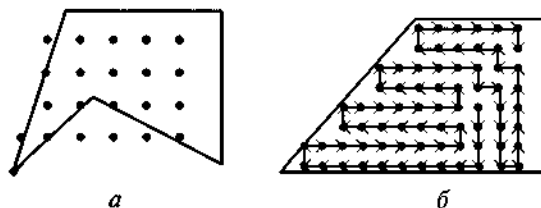


Рис. 4.34. Закрашивание 4-связной области:

a — порядок перебора соседних пикселей; *б* — порядок закрашки; • — начальный пиксел; *x* — центр отображаемого пиксела

села. Более экономичным является построчный алгоритм заливки, который использует пространственную когерентность:

- пиксели в строке меняются только на границах;
- при перемещении к следующей строке размер закрашиваемой строки скорее всего или не изменяется, или меняется на один пиксел.

Таким образом, на каждый закрашиваемый фрагмент строки в стеке хранятся координаты только одного начального пиксела, что приводит к существенно уменьшению размера стека.

Последовательность работы алгоритма для гранично определенной области следующая:

1) координата затравки помещается в стек, затем до исчерпания стека выполняются пп. 2-4;

2) координата очередной затравки извлекается из стека и выполняется максимально возможное закрашивание вправо и влево по строке с затравкой, т. е. пока не попадется граничный пиксел. Пусть это $X_{лев}$ и $X_{прав}$ соответственно;

3) анализируется строка ниже закрашиваемой в пределах от $X_{лев}$ до $X_{прав}$ и в ней находятся крайние правые пиксели всех незакрашенных фрагментов. Их координаты заносятся в стек;

4) то же самое проделывается для строки выше закрашиваемой.

Сглаживание ступенчатости. Появление ступенчатости, или лестничного эффекта (например, при растеризации линий с наклоном, не равным 90° и 0 на рис. 4.28), связано с дискретной природой пикселей. На рис. 4.35 показан пример растеризации черного прямоугольника при условии, что каждый пиксел имеет только два значения яркости — черный и белый. Подобные примеры

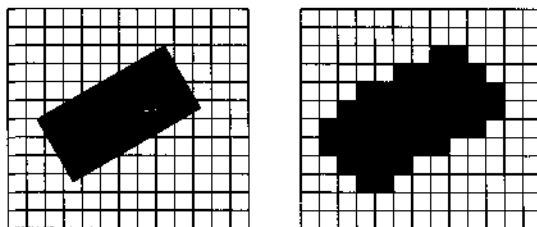


Рис. 4.35. Ступенчатость прямоугольника



Рис. 4.36. Эффекты искажения мелких объектов

можно привести и при дискретизации уровней цветности пикселей. Другим проявлением ступенчатости является некорректная визуализация тонких деталей или тонкой текстуры изображений. Если объект меньше размера пиксела и не покрывает центр пиксела, служащий для оценки атрибутов пиксела, то он не учитывается в результирующем изображении. С другой стороны, если малый объект покрывает этот центр, то он может слишком сильно повлиять на атрибуты пиксела. На рис. 4.36 изображены эффекты искажения мелких объектов. Только крайний слева маленький объект будет отображен, остальные маленькие объекты будут проигнорированы, так как они не затрагивают центры пикселей.

Основная причина появления ступенчатости, или лестничного эффекта, заключается в том, что отрезки, ребра многоугольника, цветовые границы, маленькие замкнутые области имеют непрерывную природу, тогда как растровые данные всегда дискретны. Погрешность дискретизации непрерывных линий и областей имеет ту же природу, что и погрешность дискретизации непрерывных сигналов в теории обработки сигналов, откуда и пришел сам термин «ступенчатость» (aliasing). Рассмотрим, например, быстроизменяющийся сигнал, изображенный на рис. 4.37, а, измеряемый посредством равномерных опросов с интер-

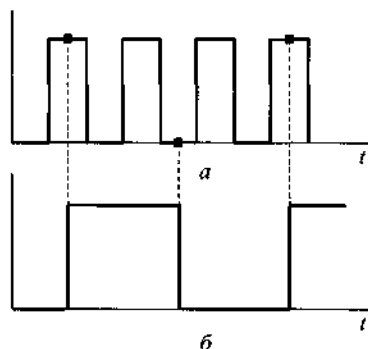


Рис. 4.37. Дискретизация сигнала с недостаточной частотой: а — исходный сигнал; б — результирующий сигнал, формируемый по точкам стробирования

валами, показанными точками. Если основываться только на этих опросах, то исходный сигнал будет выглядеть как сигнал с более низкой частотой (рис. 4.37, б).

Для высокочастотного сигнала выборка проведена с недостаточной частотой. Согласно теореме Котельникова—Найквиста, для предотвращения искажения следует проводить выборку сигнала с частотой, по крайней мере вдвое превышающей наибольшую частоту сигнала. Недостаточная частота следования пикселей приводит к такого же рода искажениям.

Можно выделить два подхода к сглаживанию ступенчатости:

- увеличение частоты следования пикселей (разрешения растрового изображения);

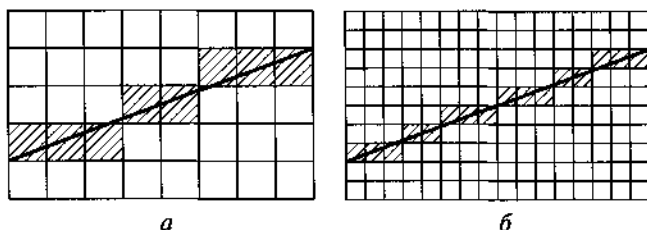


Рис. 4.38. Растреризация отрезка прямой:

a — с одинарным разрешением; *б* — с увеличенным разрешением

• размытие линий или границ областей при заданной частоте следования пикселей.

На рис. 4.38 приведен пример растреризации отрезка прямой с одинарным и удвоенным разрешением.

Если разрешение выходного изображения ограничено возможностями графических устройств, то следует все вычисления проводить с высоким разрешением, а изображать с более низким, используя усреднение некоторого типа для получения атрибутов пикселей с более низким разрешением. Хотя результат увеличения разрешения выглядит лучше, это улучшение потребует в 4 раза больше памяти и увеличения времени растреризации, поэтому этот подход дорогостоящий и полностью проблему сглаживания ступенчатости не решает.

Уменьшить ступенчатость путем размытия пикселей можно только в том случае, если каждый пиксел будет иметь больше двух градаций яркости. Для полутоновых изображений — это количество возможных значений яркости. Для бинарных изображений используется технология аппроксимации полутонами или псевдотонирование. Чтобы получить несколько градаций яркости в бинарном изображении с фиксированным разрешением, несколько пикселей объединяют в конфигурации. При этом за счет ухудшения пространственного разрешения можно улучшить качество растрового изображения.

На рис. 4.39 изображена одна из возможных групп конфигураций для бинарного изображения. Для каждой клетки используется девять пикселей. Они дают десять уровней (с 0 по 9) яркости клетки. Клетки конфигураций не обязательно должны быть квадратными, а размеры круглых черных точек могут быть разными.

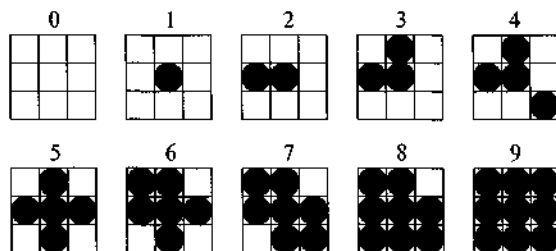


Рис. 4.39. Десять градаций яркости для бинарного изображения

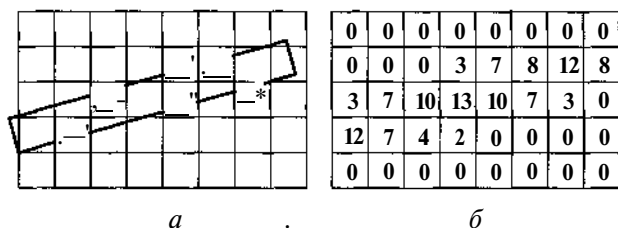


Рис. 4.40. Сглаживание ступенчатости с использованием части площади пикселей, покрываемой объектом:

a — исходная линия; *б* — градации яркости растеризованной линии

В отличие от увеличения разрешения, когда пиксел рассматривается как геометрическая точка, при размытии линий пиксел трактуется как конечная область.

В простейших алгоритмах сглаживания ступенчатости яркость пикселей вычисляется на основе покрытия объектом пикселей, т. е. вычисляется часть площади пикселя, которая покрывается объектом. В этом случае линия, которая в идеале имеет нулевую толщину, представляется как линия шириной в один пиксел. Предположим, что каждый пиксел имеет 16 градаций яркости (0 — черный цвет, 15 — белый цвет). На рис. 4.40 показано использование части площади пикселя, покрываемой линией толщиной один пиксел.

Геометрические вычисления, необходимые для определения степени покрытия каждого пикселя, могут занять много времени, поэтому разработано много эффективных алгоритмов, которые используют инкрементные вычисления и арифметику целых чисел, т. е. модификации алгоритма Брезенхема.

В более сложных алгоритмах сглаживания ступенчатости используется фильтрация полученных по простым алгоритмам данных, т. е. значение яркости каждого пикселя вычисляется как средневзвешенное значение соответствующего набора соседних пикселей (обычно из восьми ближайших соседей). На каждый ненулевой пиксел поочередно накладывается квадратная маска свертки некоторого фильтра, которую называют оконной функцией. Затем вес каждой клетки умножается на соответствующее значение пикселя, девять произведений складываются и образуют значение яркости обрабатываемого пикселя. На рис. 4.41 приведены примеры масок, используемых в практических алгоритмах. Во всех масках вес центрального пикселя превышает вес соседей, а веса линейно возрастают по мере движения от краев маски к ее центру. Иногда используют большие маски размером 5 x 5 и даже 7 x 7.

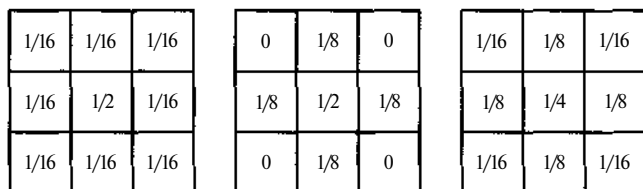


Рис. 4.41. Примеры масок свертки для сглаживания ступенчатости

4.1.4. Входные и выходные данные векторные

Обработка векторных данных является фундаментом КГ, поскольку векторные данные обеспечивают оптимальный способ хранения и передачи графических и геометрических объектов в компьютерных системах, а также позволяют легко модифицировать эти объекты и представлять их в нужном масштабе и в нужном расположении на плоскости. Можно выделить пять основных групп алгоритмов векторной плоской графики:

- геометрическое моделирование на плоскости;
- двумерные геометрические преобразования;
- отсечение двумерных геометрических объектов;
- решение метрических задач на плоскости;
- параметрические и вариационные геометрические модели.

Геометрическое моделирование на плоскости. Если для растровых данных базовым элементом является пиксел, то для векторных данных аналогичным базовым элементом является точка. Точки — основные строительные блоки векторных данных, именно поэтому конкретные алгоритмы с векторными данными в значительной степени зависят от способа представления точек. В общем случае местоположение точки на плоскости можно определить с помощью системы координат двумя способами: в декартовой системе координат или в полярной системе координат.

В декартовой системе координат точке P на плоскости ставится в однозначное соответствие пара чисел $P = (x_P, y_P)$ — координат точки по соответствующим осям координат (рис. 4.42, а). В полярной системе координаты точки задаются радиусом r_P геометрического вектора точки, который представляет собой направленный отрезок прямой линии, начинающийся в точке $(0, 0)$ и заканчивающийся в данной точке P и углом φ_P поворота этого вектора относительно оси x (рис. 4.42, б). Положительное направление угла отсчитывается против часовой стрелки.

В большинстве алгоритмов используется декартова система координат, но иногда для описания различных фигур на плоскости, которым присуща осевая симметрия, удобнее использовать полярную систему координат. Декартовы и полярные координаты связаны между собой следующими соотношениями:

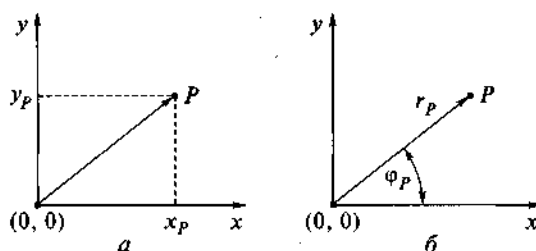


Рис. 4.42. Декартова (а) и полярная (б) системы координат

$$\begin{cases} x_p = r_p \cos(\varphi_p), & y_p = r_p \sin(\varphi_p); \\ r_p = \sqrt{x^2 + y^2}, & \varphi = \arctg(y/x). \end{cases}$$

Координаты точки могут быть описаны вещественными (с плавающей точкой) или целыми числами. Представление координат точек целыми числами требует минимальных затрат памяти и дает значительное преимущество в быстрой работе конечных программ. Однако при этом возникает множество проблем, связанных с ограниченным числом разрядов для описания координат точки. Для представления целых координат, как правило, используется 32-разрядное машинное слово. Наибольшее целое число, занимающее такое слово и необходимое для представления величин со знаком из положительного и отрицательного диапазонов, равно 214 783 647. Для многих приложений этого бывает вполне достаточно. Например, в известной графической системе MicroStation до недавнего времени использовалось именно такое представление точки. Этого вполне хватало для создания двумерных чертежей, однако когда требуется координата точки, которую невозможно представить с помощью такого слова, возникают серьезные проблемы. Эту трудность можно преодолеть, воспользовавшись относительными координатами. Для этого в абсолютной системе координат устанавливаются точки при заданных ограничениях, а затем с помощью относительных координат и тех же ограничений строится точка с большими значениями координат.

Другой способ преодоления этого недостатка — использование однородных координат. Этот способ вызывает некоторые осложнения при программировании и уменьшает скорость выполнения программы, но все эти недостатки оправдываются преимуществами, которые дает представление больших чисел при использовании целочисленной арифметики. Напомним, что в системе однородных координат двумерное пространство представляется трехмерным, т.е. двумерные координаты, однозначно задаваемые вектором (x, y) , выражаются через три координаты (h_x, h_y, h) , где h — произвольный множитель. Если все координатное пространство будет меньше приведенного выше ограничения, то h будет равно единице, а координаты будут представляться путем прямых преобразований. При этом если хотя бы одна из координат превысит это ограничение, то преимущества использования однородных координат становятся очевидны. Установим, например, $h = 1/2$ и координаты точки теперь можно представить в диапазоне в 2 раза большем, чем для $h = 1$. Следует отметить, что кардинальным решением этих проблем является переход к представлению координат точек вещественными числами с плавающей точкой, но за счет снижения быстродействия конечных программ и повышения требований к затратам памяти.

При работе с точками помимо определения их координат часто встает задача определения расстояния между двумя точками. Предположим, что имеем две точки $P_1 = (x_1, y_1)$ и $P_2 = (x_2, y_2)$, тогда расстояние d между ними в декартовой системе координат можно определить по формуле

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}.$$

Если координаты точек заданы в полярной системе координат $P_1 = (r_1, \varphi_1)$ и $P_2 = (r_2, \varphi_2)$, то соответствующая формула имеет следующий вид:

$$d = \sqrt{r_1^2 + r_2^2 - 2r_1r_2 \cos(\varphi_1 - \varphi_2)}.$$

Другой часто встречающейся задачей при работе с точками является задача определения коллинеарности трех точек. Три точки $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2)$ и $P_3 = (x_3, y_3)$ коллинеарны, т. е. лежат на одной прямой, если определитель матрицы

$$\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} = 0.$$

Базовым элементом векторных данных является кривая линия на плоскости. Напомним, что кривая линия в большинстве случаев описывается либо уравнением линии в неявной форме $f(x, y) = 0$, либо уравнениями линии в параметрической форме $P(t) = (x(t), y(t))$, $0 \leq t \leq 1$, где $P(t)$ — точка, принадлежащая линии. Важнейшим понятием при моделировании является понятие ориентации линии. Используя это понятие, легко формализовать передачу позиционной информации о взаимном расположении точек и линий (слева, справа, внутри, снаружи). Применение ориентированных линий позволяет получить важные практические результаты как при выборе рациональных способов описания точек и линий, так и при разработке рациональных алгоритмов решения геометрических задач. Линия, определяемая неявным уравнением $f(x, y) = 0$, делит всю плоскость на две области: положительную и отрицательную. Положительной относительно кривой называется область, все точки которой удовлетворяют неравенству $f(x, y) > 0$, а отрицательная область определяется неравенством $f(x, y) < 0$. Уравнение $f(x, y) = 0$ считается уравнением кривой, ориентированной в сторону определяемой им положительной области. На плоскости xOy линию $f(x, y) = 0$ всегда будем считать ориентированной так, чтобы при движении вдоль нее в достаточно малой окрестности этой линии положительная (отрицательная) область оставалась слева (справа). Следовательно, уравнение $-f(x, y) = 0$ является уравнением кривой, совпадающей по расположению с кривой $f(x, y) = 0$, но имеющей противоположную по отношению к ней ориентацию. Соответственно меняются положительные и отрицательные области линии. Для параметрической модели линии положительным направлением считается направление в сторону увеличения параметра t . Кроме ориентации важнейшими характери-



Рис. 4.43. Ориентированная кривая линия, касательная V и нормаль N

стиками при моделировании линии являются касательная прямая к линии и вектор нормали в некоторой точке линии. Ориентированная кривая линия, касательная и нормаль к ней в некоторой точке $P_i = P(t_i) = (x_i, y_i)$ показаны на рис. 4.43.

Уравнение касательной $V = (V_x, V_y)$ в точке $P_i = P(t_i) = (x_i, y_i)$ для неявного уравнения линии имеет вид

$$N_x(x - x_i) + N_y(y - y_i) = 0,$$

где $N_x = \frac{\partial f(x, y)}{\partial x}$ и $N_y = \frac{\partial f(x, y)}{\partial y}$ — значения соответствующих производных в точке $P_i = P(t_i) = (x_i, y_i)$.

Уравнение касательной $V = (V_x, V_y)$ в точке $P_i = P(t_i) = (x_i, y_i)$ для параметрического уравнения линии имеет следующий вид:

$$x(t) = x_i + V_x t, \quad y(t) = y_i + V_y t,$$

где $V_x = \frac{dx(t)}{dt}$ и $V_y = \frac{dy(t)}{dt}$ — значения соответствующих производных в точке t_i .

Вектор нормали $N = (N_x, N_y)$ ортогонален к линии и направлен в сторону положительной области линии. При замене уравнения линии $f(x, y) = 0$ на $-f(x, y) = 0$ направление нормали меняется на противоположное.

Направляющий вектор касательной к линии $V = (V_x, V_y)$ начинается в точке $P_i = P(t_i) = (x_i, y_i)$ и направлен по касательной в сторону увеличения t , т. е. в сторону направления ориентации линии.

Связь между вектором нормали и направляющим вектором, ввиду их ортогональности, описывается следующими выражениями:

$$N = (V_y, -V_x); \quad V = (-N_y, N_x).$$

Рассмотрим геометрические модели прямой линии на плоскости. Неявное уравнение прямой задается тремя коэффициентами A, B, D :

$$f(x, y) = Ax + By + D = 0.$$

При этом ненулевым должно быть хотя бы одно из чисел A или B .

Уравнение $By + D = 0$ описывает горизонтальную прямую, а уравнение $Ax + D = 0$ описывает вертикальную прямую.

Вектор нормали $N = (A, B)$, а направляющий вектор $V = (-B, A)$.

Неявная форма уравнения прямой, проходящей через две точки $P_1 = (x_1, y_1)$ и $P_2 = (x_2, y_2)$, имеет следующий вид:

$$\begin{vmatrix} y_2 & 1 \\ y_1 & 1 \end{vmatrix} x + \begin{vmatrix} 1 & x_2 \\ 1 & x_1 \end{vmatrix} y + \begin{vmatrix} x_2 & y_2 \\ x_1 & y_1 \end{vmatrix} = 0.$$

Неявная форма прямой линии, проходящей через точку $P_i = P(t_i) = (x_i, y_i)$ в направлении вектора $V = (V_x, V_y)$, имеет вид

$$|V_x| x + |V_y| y + (-V_y x_i + V_x y_i) = 0.$$

Параметрическими уравнениями прямой линии будут любые линейные функции $x(t)$ и $y(t)$, в частности, прямая линия $f(x, y) = Ax + By + D = 0$ может быть описана параметрическими уравнениями

$$x(t) = \frac{-AC}{A^2 + B^2} + Bt; \quad y(t) = \frac{-BC}{A^2 + B^2} + At.$$

Прямая линия, проходящая через две точки $P_1 = (x_1, y_1)$ и $P_2 = (x_2, y_2)$, определяется путем решения двух уравнений

$$x(t) = x_1 + t(x_2 - x_1); \quad y(t) = y_1 + t(y_2 - y_1).$$

Параметрические уравнения прямой линии, проходящей через точку $P_i = P(t_i) = (x_i, y_i)$ в направлении вектора $V = (V_x, V_y)$ определяются из уравнений

$$x(t) = x_i + V_x t; \quad y(t) = y_i + V_y t.$$

При работе с точками и прямыми линиями часто встречаются две задачи:

• минимальное расстояние d между точкой $P_i = P(t_i) = (x_i, y_i)$ и прямой $Ax + By + D = 0$ рассчитывается по формуле

$$d = \sqrt{\frac{(Ax_i + By_i + D)^2}{A^2 + B^2}};$$

• точка пересечения $P_i = P(t_i) = (x_i, y_i)$ двух прямых $A_1x + B_1y + D_1 = 0$ и $A_2x + B_2y + D_2 = 0$ определяется путем решения системы из двух уравнений:

$$x_i = \frac{B_1C_2 - B_2C_1}{A_1B_2 - A_2B_1} \quad \text{и} \quad y_i = \frac{C_1A_2 - C_2A_1}{A_1B_2 - A_2B_1}.$$

Точка пересечения двух прямых, заданных в параметрической форме, определяется путем решения системы из двух соответствующих линейных уравнений относительно параметра t .

Угол φ , образованный двумя прямыми $A_1x + B_1y + D_1 = 0$ и $A_2x + B_2y + D_2 = 0$, определяется по следующей формуле:

$$\cos \varphi = \frac{A_1A_2 + B_1B_2}{\sqrt{(A_1^2 + B_1^2)(A_2^2 + B_2^2)}}.$$

Две прямые $A_1x + B_1y + D_1 = 0$ и $A_2x + B_2y + D_2 = 0$ параллельны, если $A_1B_2 = A_2B_1$, и перпендикулярны, если $A_1A_2 + B_1B_2 = 0$.

Уравнения перпендикуляра, опущенного из точки $P_0 = P(t_0) = (x_0, y_0)$ на прямую, заданную неявным уравнением $|V_x|x + |V_y|y + (-V_yx_i + V_xy_i) = 0$, имеет вид

$$-V_x(x - x_0) - V_y(y - y_0) = 0.$$

Уравнения перпендикуляра, опущенного из точки $P_0 = P(t_0) = (x_0, y_0)$ на прямую, заданную параметрическим уравнением $x(t) = x_i + V_x t$, $y(t) = y_i + V_y t$, имеют следующий вид:

$$x(t) = x_0 + V_y t; \quad y(t) = y_0 - V_x t.$$

Проверка положения точки $P_0 = P(t_0) = (x_0, y_0)$ относительно прямой $f(x, y)$ заключается в вычислении значения $f_0 = f(x_0, y_0)$. Все точки справа от прямой, т. е. со стороны нормали, дадут значение $f_0 > 0$, а слева от прямой — $f_0 < 0$. Если $f_0 = 0$, то это означает, что точка лежит на прямой.

Важным компонентом векторных данных на плоскости являются конические сечения — двумерные кривые, которые описываются неявным уравнением второй степени:

$$ax^2 + 2bxy + cy^2 + 2dx + 2ey + f = 0,$$

где a, b, c, d, e и f — коэффициенты.

Определяя коэффициенты a, b, c, d, e, f , можно получить разные конические сечения — параболы, гиперболы, эллипсы, окружности. Такие кривые часто используют в прикладной машинной графике, например для разработки деталей или в чертежных системах.

Особенно важно определить положение конического сечения и его пересечения и/или точек касания с другим коническим сечением или прямой. Выбор правильного решения не всегда очевиден, что приводит к нелинейным зависимостям. Однако можно разработать алгоритмы, исключаящие нелинейную математику. Основная идея заключается в том, чтобы методами геометрических преобразований, которые будут рассмотрены ниже, в частности плоским поворотом и переносом, перевести расчеты в первый квадрант в стандартной конфигурации. Если фигура включает коническое сечение, его центр (для параболы и гиперболы — вершина) располагается в начале координат. В общем случае неизвестные центры, вершины, точки касания и пересечения вычисляются с помощью непараметрических уравнений, а параметрическое представление используется для вывода полученных результатов. Разделение расчета и вывода полученных результатов позволяет максимально использовать преимущества обоих представлений. Рассмотрим пример построения окружности по трем точкам. На рис. 4.44 заданные точки обозначены соответственно $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2)$ и $P_3 = (x_3, y_3)$.

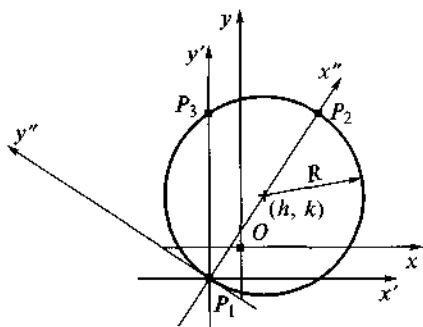


Рис. 4.44. Окружность, проходящая через три точки

Необходимо определить центр и радиус окружности. Для этого нужно решить систему из трех нелинейных уравнений с координатами центра $C(h, k)$ и радиусом R :

$$\begin{cases} (x_1 - h)^2 + (y_1 - k)^2 = R^2; \\ (x_2 - h)^2 + (y_2 - k)^2 = R^2; \\ (x_3 - h)^2 + (y_3 - k)^2 = R^2. \end{cases}$$

Эти уравнения для h и k можно упростить, перенеся начало координат в точку P_1 с последующим поворотом осей до пересечения точки P_2 с осью Ox (см. рис. 4.44):

1) перенести начало координат в точку P_1 — начало промежуточной системы координат $x'P_1y'$;

2) повернуть оси координат, чтобы одна из оставшихся точек лежала на положительной полуоси x ;

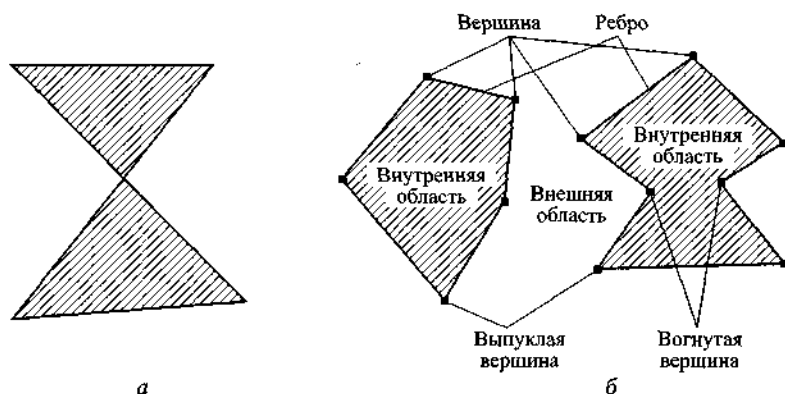


Рис. 4.45. Полигон с самопересечением (а) и простые полигоны (б)

- 3) проверить коллинеарность точек;
- 4) найти центр и радиус окружности в перенесенной и повернутой системе координат, решив линейные уравнения;
- 5) повернуть обратно к ориентации промежуточной системы координат;
- 6) перенести начало координат в первоначальное положение и найти центр окружности в исходной системе координат xOy .

Замкнутые области на плоскости, как правило, задаются полигонами (многоугольниками). Полигон — замкнутая полилиния на плоскости, образуемая отрезками прямых линий (рис. 4.45). Отрезки называют ребрами, или сторонами, полигона, а концевые точки отрезков, совпадающие для двух соседних ребер, — вершинами. Число вершин (или ребер), содержащихся в полигоне, определяет его размер. Полигон является простым, если он не пересекает самого себя. Полигоны с самопересечением всегда можно разделить на несколько простых полигонов (см. рис. 4.45).

Простой полигон охватывает непрерывную область плоскости, которая называется внутренней областью полигона. Область, окружающая простой полигон, называется внешней областью. В понятие «полигон» обычно включают заполненную внутреннюю область полигона, т. е. объединения границы и внутренней части простого полигона. Например, если говорят, что точка принадлежит полигону, то это означает принадлежность точки либо границе простого полигона, либо его внутренней области.

Вершины упорядочены циклически вдоль границы полигона. Две вершины, являющиеся концевыми точками одного и того же ребра, называются смежными. Вершина, соседняя при обходе по часовой стрелке, называется последователем, а соседняя при обходе против часовой стрелки — предшественником.

Вершины полигона подразделяют на выпуклые и вогнутые. Вершина называется выпуклой, если внутренний угол при этой вершине (находящийся в пределах внутренней области) меньше или равен 180° ; в противном случае вершина называется вогнутой (внутренний угол более 180°). Отрезок прямой линии меж-

ду двумя не соседними вершинами называется диагональю полигона. Диагональ называется хордой, или внутренней диагональю, если она лежит внутри полигона, не захватывая внешнюю область полигона.

Точку или отрезок прямой линии иногда удобно рассматривать как вырожденный полигон. Полигон-точка состоит из единственной вершины и имеет единственное ребро нулевой длины, соединяющее вершину саму с собой. Полигон-линия содержит две вершины и два совпадающих ребра, соединяющих две вершины. Кроме других преимуществ, использование вырожденных полигонов часто упрощает формирование полигона: начиная с полигона-точки, вносят вторую вершину для образования полигона-линии. Добавление последующих вершин приводит к формированию обычных полигонов с количеством вершин более двух. Если считать точки и отрезки прямых линий полигонами, то начальные этапы формирования полигона ничем не отличаются от последующих, поскольку каждый этап заключается в модификации полигонов.

Очень важны алгоритмы работы с выпуклыми простыми полигонами. Область на плоскости считается выпуклой, если для любых двух точек внутри области отрезок прямой линии между этими двумя точками полностью лежит внутри области. С выпуклыми полигонами работать проще, чем с произвольными полигонами. Например, любая диагональ в выпуклом полигоне является хордой. Кроме того, любая вершина в выпуклом полигоне будет выпуклой (в невыпуклом полигоне, по крайней мере, одна вершина вогнутая). Из этого следует, что при обходе границы выпуклого полигона по часовой стрелке в каждой вершине каждое следующее ребро либо продолжается по прямой линии, либо поворачивает вправо. Другое полезное свойство выпуклого полигона заключается в том, что пересечение любых двух выпуклых областей A и B является выпуклым. Из этого следует, что пересечение двух выпуклых полигонов является выпуклым и фактически образует выпуклый полигон (возможно, вырожденный). Более того, поскольку прямая линия является выпуклой областью, пересечение прямой линии и полигона должно быть выпуклой областью — в виде отрезка прямой линии или одинокой точки, если прямая линия только касается полигона в некоторой вершине. Выпуклый полигон обычно представляется в виде кольца вершин, хранящегося в кольцевом списке с двойными связями. Каждый узел соответствует вершине и связан со своими двумя соседями. Следуя связям, можно обойти границу полигона в любом направлении, а включая или удаляя узлы (редактируя связи соответствующим образом), можно создавать полигоны и динамически их модифицировать.

Все программное обеспечение геометрического моделирования на плоскости базируется на этих основных понятиях.

Двухмерные геометрические преобразования. Основой геометрических преобразований является использование однородных координат и реализация преобразований с помощью умножения матриц. Наибольшее внимание при разработке соответствующих алгоритмов нужно уделять формулировке задач и интерпретации результатов. Рассмотрим следующее преобразование координат точки x, y :

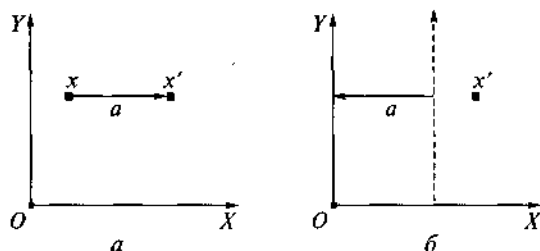


Рис. 4.46. Перемещение точки (а) и осей координат (б) на расстояние a вдоль оси X

$$\begin{cases} x' = x + a; \\ y' = y, \end{cases}$$

где x' , y' — новые координаты. Это преобразование можно интерпретировать двояко:

- все точки на плоскости XY перемещаются вправо на расстояние a (рис. 4.46, а);

- координатные оси x и y перемещаются влево на расстояние a (рис. 4.46, б).

Этот простой пример иллюстрирует принцип, применимый и в более сложных ситуациях. Как правило, преобразования, записываемые в виде произведений матриц, интерпретируются как преобразования всех точек в фиксированной системе координат. Однако те же самые преобразования можно интерпретировать и как изменение системы координат.

Аналогично в случае преобразования поворота необходимо определить:

- поворот объекта или поворот системы координат;
- способ определения положительного и отрицательного поворота;
- способ хранения координат — в виде строки или столбца матрицы;
- линию или ось, вокруг которой осуществляется поворот.

Обычно объект поворачивается в неподвижной координатной системе, положительный поворот определяется против часовой стрелки вокруг начала координат, координатные векторы представляются в виде вектора-строки. В случае однородных координат для положительного поворота вершин объекта x' на угол α вокруг начала координат используют умножение вектора-строки на матрицу поворота:

$$[X'] = [X][R],$$

$$[x' \ y' \ 1] = [x \ y \ 1] \begin{bmatrix} \cos \alpha & \sin \alpha & 0 \\ -\sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Если представить координатные векторы, заданные в однородных координатах в виде вектора-столбца, то поворот можно выполнить следующим образом:

$$[X'] = [R]^{-1} [X],$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos \alpha & -\sin \alpha & 0 \\ \sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}.$$

Для поворота системы координат без изменения координатных векторов необходимо заменить α на $-\alpha$. Эти примеры показывают, насколько аккуратно необходимо выполнять матричные преобразования.

Одной из проблем применения матриц поворота является быстрый расчет тригонометрических функций угла. Актуальность этой проблемы объясняется присутствием преобразования поворота практически в каждом сложном преобразовании и высокой частотой выполнения этой операции, в том числе и во вложенных циклах. Функции \cos и \sin обычно вычисляются суммированием рядов Тейлора до достижения необходимой точности, которая в данном случае не нужна. Для обратных тригонометрических функций, из которых во встроенной библиотеке обычно имеется только функция \arctg , сходимость рядов Тейлора до необходимой точности иногда может быть очень медленной. В этом случае наиболее экономичным будет табличный метод вычисления тригонометрических функций с кусочно-линейной интерполяцией между узлами сетки, суть которого состоит в том, что один раз с высокой точностью вычисляется $N + 1$ узловое значение наиболее часто встречающейся при выполнении геометрических преобразований монотонно убывающей функции $\cos(x)$ в интервале $x \in \left[0, \frac{\pi}{2}\right]$ с

шагом $\Delta x = \frac{\pi}{2N}$. Таким образом, в оперативной памяти компьютера создается матрица (таблица) узлов сетки:

$$x_i = i\Delta x; \quad c_i = \cos(x_i), \quad i = 0, \dots, N.$$

При вычислении тригонометрической функции $\cos(x)$ произвольный аргумент x приводится в интервал $\left[0, \frac{\pi}{2}\right]$. Вычисляя значение $c(x)$ по аргументу $x \in \left[0, \frac{\pi}{2}\right]$, определяется индекс i (как целая часть от $\frac{x}{\Delta x}$), при котором $x_i \leq x \leq x_{i+1}$. Тогда приближенное линейно-интерполированное значение функции будет равно:

$$c(x) = c_i + (c_{i+1} - c_i) \left(\frac{x}{\Delta x} - i \right).$$

Аналогичным способом с помощью линейно-интерполяционных функций вычисляют $\sin(x)$ и обратные тригонометрические функции $\arccos(x)$ и $\arcsin(x)$.

Отсечение двумерных геометрических объектов. Отсечение некоторой части двумерных графических и геометрических объектов прямоугольным или непрямоугольным окном играет важную роль в задачах КГ. Эти алгоритмы можно реализовать аппаратно или программно. Алгоритмы отсечения, реализованные программно, оказываются недостаточно быстродействующими для приложений, ориентированных на процессы, протекающие в реальном времени. Поэтому многие двумерные алгоритмы отсечения реализуются аппаратными или микропрограммными средствами.

Фундаментальной задачей в этом классе алгоритмов является задача отсечения отрезков прямых линий. Отсечение более сложных объектов, например многоугольников, сводится к последовательному отсечению отрезков. Конические кривые при отсечении могут быть аппроксимированы последовательностью отрезков, а параметрические кривые представляются в памяти своими характеристическими многоугольниками, и в этом случае также можно применять алгоритмы отсечения отрезков прямых. На рис. 4.47 показаны примеры отсечения отрезков прямых.

Рассмотрим простую задачу — отсечение отдельных точек с координатами x, y . Обозначим границы прямоугольника отсечения по оси x — x_{\min} и x_{\max} , а по оси y — y_{\min} и y_{\max} , тогда необходимо проверить четыре неравенства:

$$x_{\min} \leq x \leq x_{\max}; \quad y_{\min} \leq y \leq y_{\max}.$$

Если какие-нибудь из этих четырех неравенств не выполняются, то эта точка лежит вне прямоугольника отсечения. Если все четыре неравенства выполняются, то точка лежит внутри прямоугольника отсечения.

Для отсечения линии достаточно рассмотреть только ее конечные точки. Если обе конечные точки линии лежат в прямоугольнике отсечения (например, линия AB на рис. 4.47), то вся линия находится в прямоугольнике отсечения и может быть тривиально принятой. Если одна конечная точка находится внутри и одна снаружи (например, линия CD на рис. 4.47), то линия пересекает прямоугольник отсечения, и необходимо вычислить точку пересечения линии с пря-

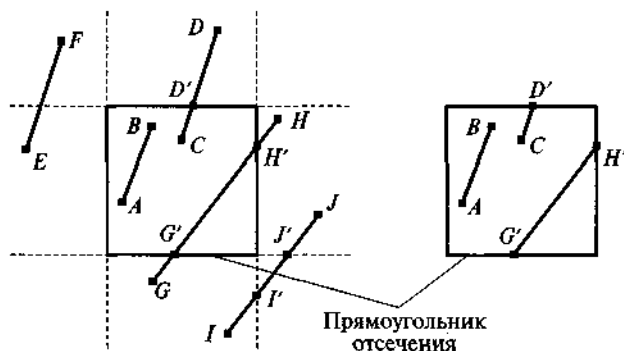


Рис. 4.47. Примеры отсечения отрезков прямых прямоугольным окном

моугольником. Если обе конечных точки являются внешними по отношению к прямоугольнику отсечения, то линия может (или не может) пересекать прямоугольник отсечения (линии GH и IJ на рис. 4.47), и нужно выполнять дальнейшие вычисления, чтобы определить, есть ли какие-либо пересечения, и если есть, то в каких точках это происходит.

Простой метод решения для отсечения линии, которая не может быть тривиально принятой, состоит в том, чтобы пересечь эту линию с каждым из четырех ребер прямоугольника отсечения и определить, лежат ли какие-либо точки пересечения на этих ребрах, и если так, то, значит, линия пересекает прямоугольник отсечения и частично находится внутри него. Для каждого отрезка прямой линии и ребра прямоугольника отсечения определяют две математически бесконечных линии, которые являются их носителями. Затем проверяют, является ли эта точка пересечения внутренней, т. е. находится ли она и на ребре прямоугольника отсечения, и на отрезке линии, и если так, то, значит, есть пересечение с прямоугольником отсечения. На рис. 4.47 точки пересечения G' и H' являются внутренними, а I' и J' не являются.

Для этого метода, если использовать формулы аналитической геометрии для описания бесконечных линий, необходимо решить систему из двух линейных уравнений для каждой пары <ребро, линия>. Параметрические формулы для сегментов линий с вычислительной точки зрения предпочтительнее:

$$x = x_0 + t(x_1 - x_0); \quad y = y_0 + t(y_1 - y_0).$$

Параметрические уравнения описывают координаты (x, y) на ориентированных сегментах линий от точки $[x_0, y_0]$ до точки $[x_1, y_1]$ для параметра t в диапазоне $[0, 1]$. Последовательно решаются две системы линейных уравнений для параметров t_e для ребра и t_l для отрезка линии. Эти значения проверяются на принадлежность диапазону $[0, 1]$; если они лежат в этом диапазоне, то точка пересечения находится в пределах обоих сегментов и является точкой пересечения прямоугольника отсечения. Сначала надо рассмотреть специальный случай линии, параллельной какому-либо ребру прямоугольника отсечения. В целом этот простой метод неэффективен и непригоден для многократного решения задачи отсечения линий.

Для повышения эффективности методов отсекаемые отрезки разделяют на три класса — целиком видимые, целиком невидимые и пересекающие окно отсечения. По способу выбора решения об отбрасывании невидимого отрезка целиком или принятия его существует два основных типа алгоритмов отсечения — алгоритмы, использующие кодирование концов отрезка или всего отрезка, и алгоритмы, использующие параметрическое представление отсекаемых отрезков и окна отсечения. Классический алгоритм первого типа — алгоритм Кохена—Сазерленда (Cohen—Sutherland). К алгоритмам второго типа относятся алгоритм Кируса—Бека (Cyrus—Beck) и алгоритм Лианга—Барски (Liang—Barsky). Алгоритмы с кодированием применимы для прямоугольного окна, стороны которо-

го параллельны осям координат, в то время как алгоритмы с параметрическим представлением применимы для произвольного окна.

Алгоритм Кохена—Сазерленда. По алгоритму Кохена—Сазерленда сначала выполняются быстрые проверки линий, для которых не надо выполнять вычисления пересечения. Затем пары конечных точек отрезков проверяются для тривиального принятия. Если линия не может быть тривиально принята, то выполняются проверки области отрезка. Например, два простых сравнения по оси X показывают, что обе конечные точки отрезка EF , представленного на рис. 4.47, имеют координату x меньше, чем x_{\min} , и таким образом отрезок лежит в области слева от прямоугольника отсечения (т. е. во внешней полуплоскости, определенной левым ребром прямоугольника отсечения), поэтому отрезок линии EF может быть тривиально отклонен. Точно так же можно тривиально отклонить отрезки с обеими конечными точками в областях справа от x_{\max} , ниже y_{\min} и выше y_{\max} .

Если отрезок линии не может быть тривиально принят или отклонен, то он разделяется на два отрезка в ребре отсечения так, чтобы один отрезок можно было тривиально отклонить. Таким образом, отрезок итерационно отсекается проверкой на тривиальное принятие или отклонение. Алгоритм особенно эффективен в двух случаях:

- очень большой прямоугольник отсечения, включающий все или большинство примитивов, которые можно тривиально принять;
- очень маленький прямоугольник отсечения, когда почти все примитивы можно тривиально отклонить. Этот случай встречается в стандартном методе выполнения выбора графических элементов, при котором маленький прямоугольник, окружающий курсор, используется для отсечения примитивов, чтобы определить, какие примитивы лежат в пределах окна выбора курсора.

Чтобы выполнить проверки тривиального принятия или отклонения, продлевают ребра прямоугольника отсечения, чтобы разделить плоскость прямоугольника отсечения на девять областей (рис. 4.48). Каждой области назначается 4-битовый код, определяемый расположением соответствующей области относительно внешних полуплоскостей ребер прямоугольника отсечения. Каждый бит в двоичном коде устанавливается в 1 или 0.

4 бита (слева направо) в коде соответствуют следующим условиям:

- первый бит — $y > y_{\max}$;
- второй бит — $y < y_{\min}$;
- третий бит — $x > x_{\max}$;
- четвертый бит — $x < x_{\min}$.

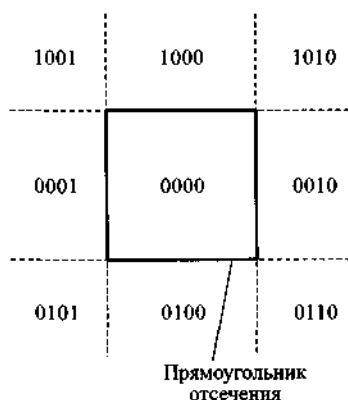


Рис. 4.48. Двоичные коды областей прямоугольника отсечения

Например, поскольку область, лежащая выше и слева от прямоугольника отсечения, находится во

внешней полуплоскости верхнего и левого ребер, то ей назначается код 1001. Особенно способ вычисления двоичного кода эффективен, когда бит 1 — знаковый разряд для $(y_{\max} - y)$, бит 2 — знаковый разряд для $(y - y_{\min})$, бит 3 — знаковый разряд для $(x_{\max} - x)$ и бит 4 — знаковый разряд для $(x - x_{\min})$. Каждая конечная точка отрезка линии получает код области, в которой она находится. Теперь можно использовать эти коды для определения, находится ли отрезок линии полностью в прямоугольнике отсечения или во внешней полуплоскости ребра. Если оба кода конечных точек являются нулевыми, то отрезок расположен полностью в прямоугольнике отсечения. Если обе конечных точки лежат во внешней полуплоскости некоторого ребра, например отрезок EF на рис. 4.47, код для каждой конечной точки имеет биты, показывающие, что эта точка находится во внешней полуплоскости этого ребра. Для отрезка EF коды равны 0001 и 1001 соответственно, откуда следует, что отрезок находится во внешней полуплоскости левого ребра. Поэтому, если логическое И кодов конечных точек не является нулевым, то линия может быть тривиально отклонена.

Если отрезок не может быть тривиально принят или отклонен, то необходимо разделить его на два подотрезка так, чтобы один или оба новых отрезка могли бы быть отвергнуты. Это разделение выполняют, используя ребро, которое разрезает линию на два отрезка. При этом отрезок, лежащий во внешней полуплоскости ребра, сразу отбрасывается. Можно выбрать любой порядок проверки ребер, но выбранный порядок проверок необходимо использовать на всех этапах алгоритма. Примем следующий порядок просмотра кодов: сверху вниз и справа налево. Основное свойство двоичного кода состоит в том, что биты, которые установлены в этом коде в единицу, соответствуют определенным ребрам. Если одна конечная точка находится во внешней полуплоскости какого-то ребра и отрезок линии не проходит проверки тривиального отклонения, то другая точка должна лежать во внутренней полуплоскости этого ребра и отрезок линии должен пересечь его. Таким образом, алгоритм всегда выбирает точку, которая находится вне области отсечения и затем использует соответствующий бит двоичного кода для определения ребра отсечения. Выбранное ребро будет первым в установленном порядке выбора (сверху вниз и справа налево), т. е. ему соответствует крайний левый бит, который установлен в единицу.

При такой реализации алгоритм будет работать следующим образом. Вычисляются двоичные коды для обеих конечных точек и проверяются для тривиального принятия или отклонения. Если обе проверки не выполняются, то находят конечную точку, которая расположена вне областей отсечения (по крайней мере, одна такая точка будет существовать). Затем проверяют двоичный код, чтобы найти ребро, которое пересекает линию. Далее определяют соответствующую точку пересечения. Можно отбросить отрезок линии от внешней конечной точки до этой точки пересечения, заменяя внешнюю конечную точку на точку пересечения и вычислить двоичный код этой новой конечной точки, чтобы подготовиться к следующей итерации.

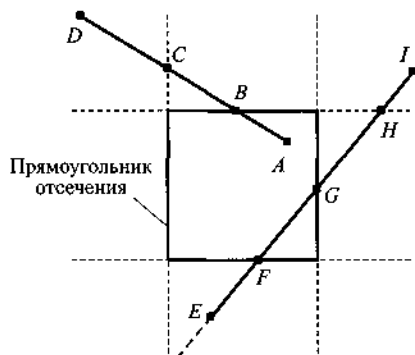


Рис. 4.49. Отсечение отрезков по алгоритму Кохена—Сазерленда

Рассмотрим отрезок линии AD , изображенный на рис. 4.49. Точка A имеет двоичный код 0000, а точка D — 1001. Этот отрезок не может быть тривиально принят или отклонен. Поэтому алгоритм за внешнюю точку выбирает точку D , двоичный код которой показывает, что линия пересекает верхнее и левое ребра. В соответствии с принятым порядком проверок сначала используется верхнее ребро, чтобы отсечь AD до AB и вычислить двоичный код для точки B как 0000. На следующей итерации выполняются тривиальные проверки для принятия или отклонения для отрезка AB и он тривиально принимается.

Линия EI (см. рис. 4.49) требует большего количества итераций. Первая конечная точка E имеет выходной код 0100, поэтому алгоритм выбирает ее как внешнюю точку и проверяет двоичный код точки, чтобы найти, какое первое ребро отсекает линию — это будет нижнее ребро; оно отсекает EI до отрезка FI . На второй итерации FI не может быть тривиально принят или отклонен. Двоичный код первой конечной точки F вычисляется как 0000, поэтому алгоритм выбирает внешнюю точку I , имеющую двоичный код 1010. Первым ребром отсечения для этой точки является верхнее ребро, что приводит к отрезку FH . Двоичный код для точки H вычисляется как 0010, поэтому третья итерация приводит к отсечению правым ребром до отрезка FG . Этот отрезок тривиально принимается на четвертой заключительной итерации. Если бы в качестве первой точки была выбрана точка I , то получилась бы другая последовательность отсечений. На основе ее двоичного кода сначала отсекался бы отрезок верхним ребром, затем правым ребром и, наконец, нижним ребром.

Рассмотренная реализация алгоритма не является самой эффективной. Поскольку проверки и отсечения делаются в установленном порядке, то алгоритм иногда выполняет бесполезные отсечения. Такое происходит, когда пересечение с ребром прямоугольника является внешним пересечением и оно не лежит на границе прямоугольника отсечения (например, точка H на линии EI на рис. 4.49). Для устранения этих недостатков разработаны различные более эффективные алгоритмы.

Алгоритм Кируса—Бека. Алгоритм Кохена—Сазерленда — наиболее известный алгоритм отсечения отрезков. Вместе с тем Кирус и Бек разработали алгоритм, который существенно отличается от рассмотренного выше алгоритма и использует более эффективный метод отсечения отрезков. Алгоритм Кируса—Бека можно использовать для двухмерного отсечения отрезков как прямоугольником, так и произвольным выпуклым многоугольником. Позже Лианг и Барски разработали более надежный алгоритм, отсекающий отрезки параметри-

чески, а также более эффективные проверки тривиальных отклонений, которые справедливы для любых областей отсечения. Рассмотрим основы алгоритма Кируса—Бека. Напомним, что алгоритм Кохена—Сазерленда для линий, которые не могут быть тривиально приняты или отклонены, вычисляет точки пересечения отрезка линии с ребром отсечения, подставляя известное значение координат x или y для вертикального или горизонтального ребра отсечения соответственно. Параметрический алгоритм отсечения отрезка определяет значение параметра t в параметрическом представлении отрезка линии для точки, в которой бесконечная линия этого отрезка пересекает бесконечную линию, на которой находится ребро отсечения. Поскольку все ребра отсечения в общем случае пересекаются такой линией, надо рассчитать четыре значения t . При этом для определения тех из четырех значений t , которые соответствуют фактическим пересечениям, используется ряд простых сравнений. Только после этого определяются значения (x, y) для одного или двух фактических пересечений. В целом этот подход более экономичный, чем алгоритм вычисления пересечения Кохена—Сазерленда, поскольку он не содержит цикла для отсечений несколькими ребрами прямоугольника отсечения. Кроме того, вычисления относительно одного параметра более простые, чем решение уравнений относительно координат (x, y) . Лианг и Барски улучшили алгоритм Кируса—Бека, исследуя каждое значение t для того, чтобы отклонить некоторый отрезок линии, прежде чем все четыре значения t будут вычислены.

Алгоритм Кируса—Бека основан на параметрической формулировке пересечения между двумя линиями. На рис. 4.50 изображено единственное ребро E_i прямоугольника отсечения и нормаль N_i этого ребра, направленную вне области отсечения (т. е. направленную наружу относительно прямоугольника отсечения), а также отрезок линии PQ , который должен быть отсечен этим ребром. Ребро и отрезок линии необходимо представить бесконечной прямой линией, чтобы найти точку пересечения. Параметрически такая линия описывается следующим уравнением:

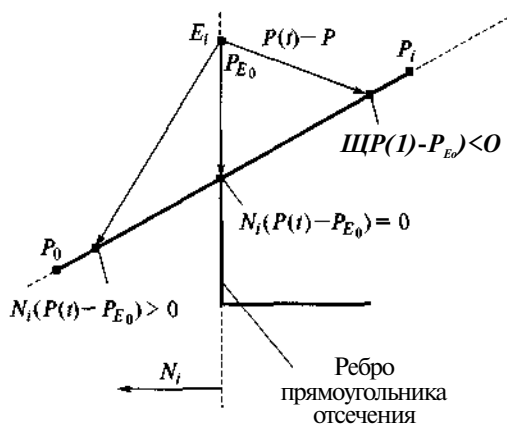


Рис. 4.50. Параметрические оценки для трех точек — внутри, вне и на границе ребра отсечения

$$P(t) = P_0 + (P_1 - P_0)t,$$

где $t = 0$ в точке P_0 и $t = 1$ в точке P_1 .

Выберем произвольную точку P_{E_i} на ребре E_i и рассмотрим три вектора $P(t) - P_{E_i}$ от P_{E_i} до трех определяемых точек $P(t)$ на отрезке линии от P_0 до P_1 точка пересечения, которая будет определена; точки линии во внутренней полуплоскости ребра и точки линии во внешней полуплоскости ребра. Анализируя значение скалярного произведения, можно различить, в какой области находится точка. Это значение отрицательно для точки во внутренней полуплоскости, равно нулю для точки на линии, содержащей ребро, и положительно для точки, которая находится во внешней полуплоскости. Определение внутренних и внешних полуплоскостей ребра соответствует просмотру областей отсечения ребер против часовой стрелки. Таким образом можно найти значение t для точки пересечения вектора PQ с ребром:

$$N_i(P(t) - P_{E_i}) = 0.$$

Заменим $P(t)$:

$$N_i(P_0 + (P_1 - P_0)t - P_{E_i}) = 0,$$

затем сгруппируем члены уравнения

$$N_i(P_0 - P_{E_i}) + N_i(P_1 - P_0)t = 0.$$

Введем вектор $D = (P_1 - P_0)$ и определим V .

$$\frac{N_i(P_0 - P_{E_i})}{-N_i D} \quad (4.2)$$

Знаменатель выражения (4.2) должен быть отличным от нуля, поэтому алгоритм проверяет:

- $N_i \neq 0$ (т. е. нормаль не должна быть равна нулю; это могло быть только ошибкой);
- $D \neq 0$ (т. е. $P_1 \neq P_0$);
- $N_i D \neq 0$ (т. е. ребро E_i и отрезок P_0P_1 не параллельны).

Уравнение для вычисления t можно использовать для нахождения пересечения между отрезком P_0P_1 и каждым ребром прямоугольника отсечения. Для этого определяют нормаль и произвольную точку P_{E_i} , например конечную точку ребра для каждого ребра отсечения. Далее используют эти значения для всех отрезков линий. Определив четыре значения t для некоторого отрезка линии, на следующем шаге находим, какое (или любое) из значений соответствует внутренним пересечениям отрезка линии с ребрами прямоугольника отсечения. Сначала любое значение t вне интервала $[0, 1]$ может быть отвергнуто, так как оно

находится вне отрезка P_0P_1 . Затем необходимо определить, находится ли пересечение на ребре отсекающего. Можно попробовать просто сортировать оставшиеся значения t , выбрав промежуточные значения t для точек пересечения, как показано на рис. 4.51 для линии 1.

Как отличить линию 1 от линии 2, в которой никакие части отрезка линии не находятся в прямоугольнике отсекающего, а промежуточные значения t соответствуют точкам не на ребрах отсекающего? Также, какие из четырех пересечений линии 3 лежат на ребрах отсекающего?

Точки пересечения, представленные на рис. 4.51, можно разделить на две группы: потенциально входящие в прямоугольник отсекающего (PE) или потенциально выходящие из прямоугольника отсекающего (PL) следующим образом: если при перемещении от P_0 до P_1 пересекаем специфическое ребро, чтобы войти во внутреннюю полуплоскость какого-либо ребра, то это пересечение — PE; если остаемся во внутренней полуплоскости ребра, то это пересечение PL. Две внутренние точки пересечения отрезка, пересекающие прямоугольник отсекающего, имеют разные обозначения.

Формально пересечения могут быть классифицированы как PE или PL на основе угла между отрезком P_0P_1 и нормалью N_i : если этот угол меньше 90° , то точка пересечения PL; если больше 90° , то точка пересечения PE. Эта информация содержится в знаке скалярного произведения N_i и P_0P_1 : $N_i D < 0 \Rightarrow PE$, $N_i D > 0 \Rightarrow PL$.

Скалярное произведение $N_i D$ представляет собой знаменатель выражения (4.1), т. е. в процессе вычисления значения t точку пересечения можно классифицировать сразу. С учетом этой классификации алгоритм для линии 3, изображенной на рис. 4.51, предполагает выполнение заключительного шага — выбор пары (PE, PL), которая определит отсекаемый отрезок. Часть бесконечной линии для отрезка P_0P_1 , которая находится в пределах области отсекающего, ограничена точкой пересечения PE с наибольшим значением t , которое назовем t_E , и точкой пересечения PL с наименьшим значением t , которое назовем t_L . Отсекаемый отрезок линии определяется диапазоном $[t_E, t_L]$. Но поскольку нас интересует пересечение отрезка P_0P_1 , а не бесконечной линии, то определение диапазона должно быть изменено так, чтобы $t = 0$ было нижней границей для t_E , а $t = 1$ — верхней границей для t_L . Если $t_E > t_L$, то это точно соответствует линии 2. Это означает, что P_0P_1 не находится в пределах прямоугольника отсекающего и весь

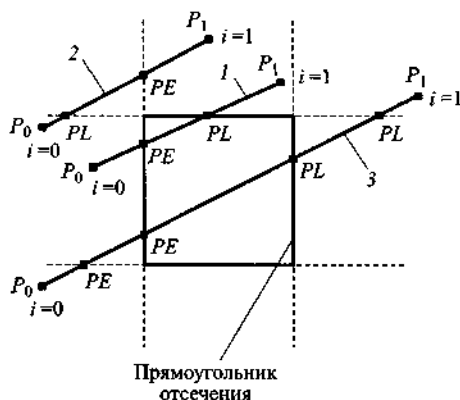


Рис. 4.51. Случаи пересечения ребер прямоугольника отсекающего с отрезками:

1–3 — линии

отрезок отклоняется. Значения t_E и t_L , соответствующие фактическим пересечениям, используются для вычисления координат x и y отсеченных отрезков.

В алгоритме Лианга—Барски для определения типов точек PE или PL используется знак знаменателя выражения (4.1), вычисляется значение параметра пересечения и выясняются тривиальные отклонения отрезков. Этот знак также выясняет тривиальные отклонения, если отрезок параллелен ребру или находится вне области прямоугольника отсечения. Основная процедура алгоритма выполняет фактическое отсечение только для отрезков внутри всех четырех ребер.

В целом алгоритм Кохена—Сазерленда эффективен, когда двоичный код может быть получен, например, выполнением поразрядных операций на ассемблере и вопрос о тривиальном принятии или отклонении применим к большинству отрезков линий. Параметрическое отсечение линии лучше, когда много отрезков линий должны быть отсечены, так как фактическое вычисление координат точек пересечения отложено, пока все проверки не будут сделаны с использованием параметрического выражения для отрезков линий. Алгоритм Лианга—Барски более эффективен, чем алгоритм Кируса—Бека, так как в нем выполняется проверка на тривиальное отклонение отрезков.

Алгоритмы отсечения многоугольников. В алгоритме, который отсекает какой-либо многоугольник, как показано на рис. 4.52, следует предусмотреть много различных случаев.

Рис. 4.52, *а* и *в* примечательны тем, что выпуклый и невыпуклый многоугольники отсекаются на два отдельных многоугольника. Каждое ребро многоугольника необходимо проверить для каждого ребра прямоугольника отсечения;

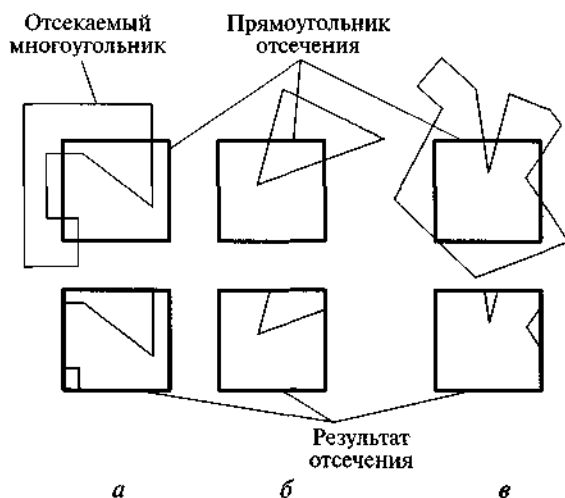


Рис. 4.52. Примеры отсечения многоугольников:

а — выпуклый многоугольник — отсекаются два; *б* — выпуклый многоугольник — отсекается один; *в* — невыпуклый многоугольник — отсекаются два

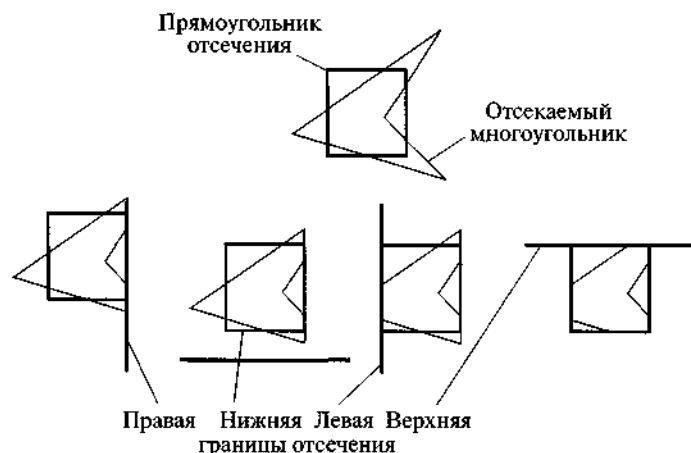


Рис. 4.53. Последовательное отсечение многоугольника

новые ребра должны быть добавлены, а существующие ребра — отвергнуты, сохранены или разделены. Много новых многоугольников можно получить из отсечения единственного многоугольника.

Известный алгоритм Сазерленда—Ходгмана использует фундаментальную стратегию многих алгоритмов КГ — разделий и властвуй. Основная идея этого алгоритма состоит в том, что отсечь многоугольник одной прямой очень легко. В этом алгоритме исходный и каждый из промежуточных многоугольников отсекается последовательно относительно одной бесконечной прямой. Работа алгоритма для прямоугольного окна изображена на рис. 4.53. Исходный многоугольник задается списком его вершин P_1, \dots, P_n , который порождает список его ребер $P_1P_2, P_2P_3, \dots, P_nP_1$. На рис. 4.53 показано, что многоугольник сначала отсекается правой стороной окна, в результате чего получается промежуточная фигура. Затем алгоритм вновь отсекает эту фигуру нижней стороной окна. Получается вторая промежуточная фигура. Далее процесс отсечения продолжается с оставшимися сторонами окна. Этим алгоритмом можно отсекать любой многоугольник, выпуклый или невыпуклый, плоский или неплоский, относительно любого окна, являющегося выпуклым многоугольником. Порядок отсечения многоугольника разными сторонами окна не имеет значения.

Результатом работы алгоритма Сазерленда—Ходгмана является список вершин многоугольника, у которого все вершины лежат внутри прямоугольника отсечения. Поскольку каждая сторона многоугольника отсекается независимо от других, то достаточно рассмотреть только возможные ситуации расположения одного отрезка относительно одной отсекающей линии. Будем рассматривать каждую точку P из списка вершин многоугольника, за исключением первой, как конечную точку ребра, стартовой точкой S которого является вершина, предшествующая P в этом списке. Тогда возможны только четыре ситуации взаимного расположения ребра и отсекающей линии (рис. 4.54).

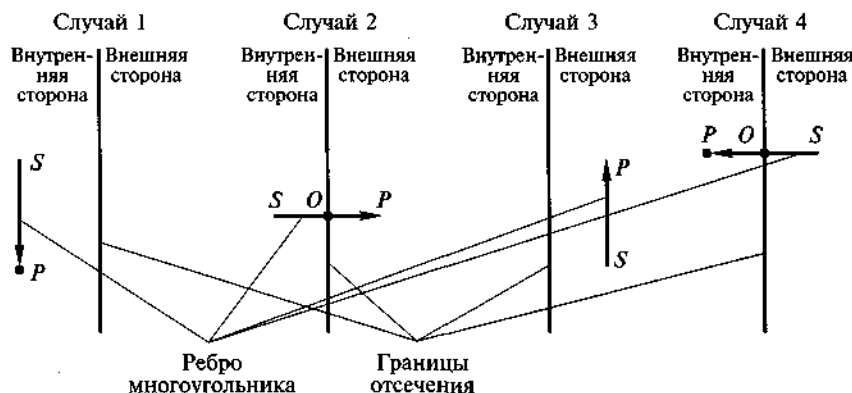


Рис. 4.54. Четыре случая отсечения ребра многоугольника:

■ — новая вершина многоугольника после отсечения

Результатом каждого сопоставления ребра многоугольника с отсекающей линией будет занесение в список вершин результирующего усеченного многоугольника: ни одной вершины (случай 3), одной (случай 1 и случай 2) или двух вершин (случай 4). Новые выходные вершины на рис. 4.54 обозначены точками O .

Если рассматриваемое ребро находится полностью внутри области отсечения, то результатом будет вершина P (случай 1). Заносить в результат начальную вершину S в этом случае не надо, так как если вершины рассматриваются поочередно, то S уже была конечной точкой предыдущего ребра и поэтому уже попала в результат. Если же ребро расположено полностью вне области отсечения, то результат не изменяется (случай 3). Если ребро многоугольника находится частично внутри области отсечения (случай 2 и 4), то оно может входить или выходить из области отсечения. Если ребро выходит из области отсечения, то надо определить и занести в результат точку O пересечения ребра и отсекающей линии. Если же ребро входит в область отсечения, то следует поступить точно так же. Поскольку в последнем случае конечная вершина P ребра видима, то она также должна попасть в выходной список вершин нового многоугольника после отсечения. Для первой вершины многоугольника необходимо определить только факт попадания ее в область отсечения. Если вершина находится внутри области отсечения, то она попадает в результат и становится начальной точкой S . Если же вершина не находится внутри области отсечения, то она тоже становится начальной точкой, но в результат не попадает. Последнее ребро P_nP_1 следует рассмотреть особо. Его обработка реализуется путем запоминания первой вершины многоугольника во временной вершине F . Тогда последним ребром становится P_nF , и его можно обрабатывать точно так же, как и любое другое ребро.

Для использования вышеизложенного алгоритма отсечения многоугольников требуются прямоугольные окна или выпуклые отсекающие области. Однако во многих приложениях необходимо уметь отсекал невыпуклыми отсекающи-

ми многоугольниками. Этой потребности отвечает более мощный, но и более сложный алгоритм Вейлера—Азертонна. Данный алгоритм позволяет производить отсечение невыпуклого многоугольника с внутренними пустыми областями по другому невыпуклому многоугольнику, который также может иметь внутренние пустые области.

Будем называть многоугольник, который отсекается, обрабатываемым многоугольником, а многоугольник, по которому производится отсечение, — отсекающим многоугольником (отсекателем). Новые границы, образуемые в результате отсечения обрабатываемого многоугольника отсекающим, совпадают с участками границ отсекаателя. Никаких иных новых границ не возникает. Следовательно, число многоугольников в результате минимально. Как обрабатываемый, так и отсекающий многоугольники описываются в алгоритме циклическими списками их вершин. Внешняя граница каждого из этих многоугольников обходится по часовой стрелке, а внутренние границы или отверстия — против часовой стрелки. Это условие означает, что при обходе вершин многоугольника в порядке их следования в соответствующем списке внутренняя его область будет расположена справа от границы. Границы обрабатываемого и отсекающего многоугольников могут пересекаться или не пересекаться между собой. Если они пересекаются, то точки пересечения образуют пары. Одно пересечение из пары возникает, когда ребро обрабатываемого многоугольника входит внутрь отсекающего многоугольника, а другое — когда оно выходит оттуда. Основная идея заключается в том, что алгоритм начинает свою работу с точки пересечения входного типа, затем он прослеживает внешнюю границу по часовой стрелке до тех пор, пока не обнаруживается еще одно ее пересечение с отсекающим многоугольником. В точке последнего пересечения производится поворот направо и далее прослеживается внешняя граница отсекаателя по часовой стрелке до тех пор, пока не обнаруживается ее пересечение с обрабатываемым многоугольником. И вновь в точке последнего пересечения производится поворот направо и далее прослеживается граница обрабатываемого многоугольника. Этот процесс продолжается до тех пор, пока не встретится начальная вершина. Внутренние границы обрабатываемого многоугольника обходятся против часовой стрелки.

В случае пересечения границ отсечения и отсекаемого многоугольника образуются точки двух типов:

- входные точки, когда ориентированное ребро отсекаемого многоугольника входит в область отсечения;
- выходные точки, когда ребро отсекаемого многоугольника идет с внутренней на внешнюю сторону области отсечения.

Общая схема алгоритма Вейлера—Азертонна для определения части отсекаемого многоугольника, попавшей в окно, следующая:

- 1) строятся списки вершин отсекаемого многоугольника и отсекаателя;
- 2) отыскиваются все точки пересечения. При этом расчете касания не считаются пересечением, т. е. когда вершина или ребро отсекаемого многоугольника инцидентна или совпадает со стороной отсекаателя;

3) списки координат вершин отсекаемого многоугольника и отсекаателя дополняются новыми вершинами — координатами точек пересечения. Причем если точка пересечения P_i находится на ребре, соединяющем вершины V_k, V_m , то последовательность точек V_k, V_m превращается в последовательность V_k, P_i, V_m . При этом устанавливаются двухсторонние связи между одноименными точками пересечения в списках вершин отсекаемого многоугольника и окна. Входные и выходные точки пересечения образуют отдельные подписки входных и выходных точек в списках вершин;

4) определение части обрабатываемого многоугольника, попавшей в окно, выполняется следующим образом:

- если не исчерпан список входных точек пересечения, то выбираем очередную входную точку;

- движемся по вершинам отсекаемого многоугольника, пока не обнаружится следующая точка пересечения; все пройденные точки, не включая прервавшую просмотр, заносим в результат; используя двухстороннюю связь точек пересечения, переключаемся на просмотр списка вершин окна;

- движемся по вершинам окна до обнаружения следующей точки пересечения; все пройденные точки, не включая последнюю, прервавшую просмотр, заносим в результат, используя двухстороннюю связь точек пересечения; переключаемся на список вершин обрабатываемого многоугольника;

- эти действия повторяем, пока не будет достигнута исходная вершина — очередная часть отсекаемого многоугольника, попавшая в окно, замкнулась. Переходим на выбор следующей входной точки в списке отсекаемого многоугольника.

Решение метрических задач на плоскости. К метрическим задачам на плоскости относятся задачи вычисления длины кривых линий, а также площадей, геометрических центров тяжести, статических моментов относительно осей координат, осевых и центробежных моментов инерции относительно осей, проходящих через центр тяжести и параллельных соответствующим осям координат для замкнутых областей. Общий подход к решению этих задач заключается в кусочно-линейной аппроксимации кривых линий отрезками прямых и в решении указанных задач для полученных ломаных и полигонов.

На рис. 4.55 показана аппроксимация кривой отрезками прямых линий при установленном допуске на максимальное отклонение d истинной кривой от хорды, соединяющей две последовательные точки, измеряемое по нормали к хорде и при заданной максимальной длине хорды L .

Обозначим вектор, соединяющий начальную и конечную точку очередной хорды, буквой C . Условием максимального отклонения точек кривой от хорды является условие параллельности касательной к кривой и данной хорды, т. е. скалярное произведение нормали N к кривой и вектора C должно быть равно нулю:

$$NC = 0.$$

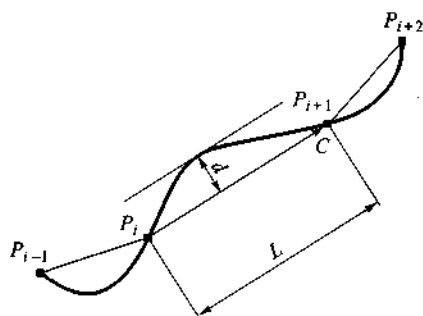


Рис. 4.55. Кусочно-линейная аппроксимация кривой

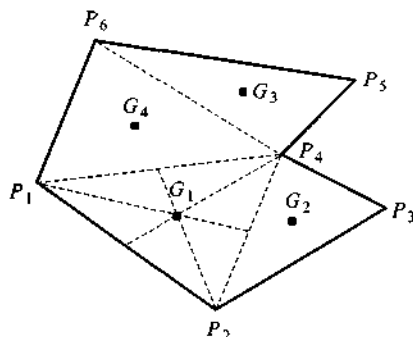


Рис. 4.56. Триангуляция полигона

Используя это уравнение, при помощи надлежащего итерационного процесса можно вычислить последовательность прямолинейных сегментов максимальной длины, лежащих в пределах предписанного допуска. Длину кривой легко определить, просуммировав длины соответствующих хорд.

Рассмотрим методы вычисления площади S и геометрического центра тяжести для полигона, аппроксимирующего некоторую криволинейную область. Площадь полигона и его геометрический центр можно вычислить, выполнив его триангуляцию — разрезание на непересекающиеся треугольники.

На рис. 4.56 приведен пример триангуляции некоторого полигона на четыре треугольника. Сначала вычисляются площадь и геометрический центр каждого треугольника. Рассмотрим в качестве примера первый треугольник $P_1 P_2 P_4$, изображенный на рис. 4.56. Геометрический центр G_1 этого треугольника находится на пересечении его медиан и делит каждую из них в отношении 2 : 1. Отсюда получаем

$$x_{G_1} = \frac{x_{P_1} + x_{P_2} + x_{P_4}}{3}; \quad y_{G_1} = \frac{y_{P_1} + y_{P_2} + y_{P_4}}{3}.$$

Площадь S_1 этого треугольника можно определить разными способами, наиболее экономичной является формула, использующая норму векторного произведения векторов любых двух сторон треугольника, например $P_1 P_4$ и $P_1 P_2$:

$$S_1 = 0,5 |P_1 P_4 \times P_1 P_2|.$$

Площадь S и геометрический центр G полигона вычисляются по следующим формулам:

$$S = \sum_i S_i; \quad G = \frac{\sum_i S_i G_i}{S},$$

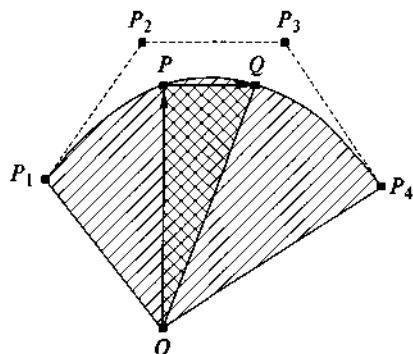


Рис. 4.57. Площадь области, ограниченной сегментом кривой Безье

Площадь элементарного треугольника OPQ (см. рис. 4.57) равна $0,5|OP \times PQ|$. Учитывая, что при уменьшении вектора PQ , он в пределе стремится к вектору касательной V , в точке P площадь области S , ограниченной сегментом кривой и прямыми, соединяющими начало координат с конечными точками сегмента, определяется формулой

$$S = 0,5 \int_{t_i}^{t_{i+1}} P(t) \times V(t) dt.$$

В случае кубической кривой Безье, показанной на рис. 4.57, вычисление данного интеграла позволяет определить площадь через радиусы-векторы полюсов OP_1 , OP_2 , OP_3 и OP_4 характеристической ломаной сегмента кривой:

$$S = \frac{3}{10}(OP_2 - OP_1) \times (OP_3 - OP_2) + \\ + \frac{3}{10}(OP_3 - OP_2) \times (OP_4 - OP_3) + \frac{9}{20}(OP_4 - OP_3) \times (OP_2 - OP_1).$$

Другие метрические задачи на плоскости решаются с помощью методов, аналогичных рассмотренным.

Параметрические и вариационные геометрические модели. Разработка сложной геометрической модели — непростая задача, требующая много времени, поэтому необходимо иметь возможность многократного использования разрабатываемых моделей. Это также важно для стандартизации моделей и создания семейства геометрических моделей. Для решения этой задачи были разработаны системы параметрического или вариационного моделирования.

Термины «параметрические» и «вариационные» геометрические модели почти эквивалентны и, с точки зрения конечного использования таких моделей, они практически не различаются. Процесс моделирования, поддерживающий и параметрические, и вариационные модели, одинаков:

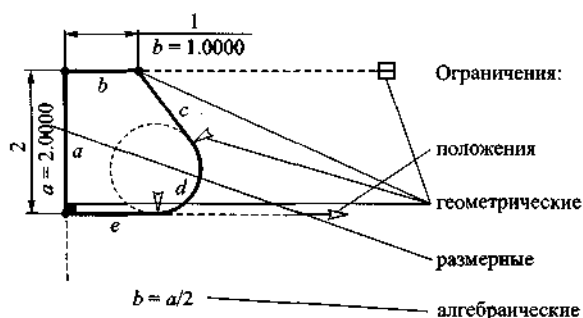


Рис. 4.58. Плоская параметрическая геометрическая модель

1) пользователь создает первоначальную топологию геометрической модели посредством обычного геометрического моделирования. Полученная в результате модель содержит нужные геометрические элементы, начальные значения их параметров и связи между элементами;

2) описываются требуемые связи между элементами модели в терминах ограничений. Ограничения определяют желаемые математические отношения между элементами модели. Типичными ограничениями для плоских моделей являются ограничения положения, геометрические, размерные и алгебраические, показанные на рис. 4.58;

3) система параметрического/вариационного моделирования выполняет общую процедуру решения ограничений, приводящую к параметризованной модели, в которой, если это возможно, выполняются объявленные ограничения. Если система ограничений не может быть выполнена, то система параметрического моделирования выдает предупреждающее сообщение пользователю;

4) пользователь системы параметрического/вариационного моделирования может создавать варианты модели, изменяя значения параметров элементов и ограничений. После каждого изменения новый вариант модели создается повторным запуском процедуры решения ограничений. В ряде случаев пользователь может также добавлять новые ограничения или удалять существующие ограничения для получения других вариантов геометрических моделей.

Геометрическими элементами параметрической модели могут быть точки, бесконечные прямые, эллипсы или кривые линии, которые позволяют ограничениям локализовать место расположения, ограничить или упорядочить другие элементы модели. Например, прямая линия как элемент параметрической модели может быть осевой линией для какого-либо симметричного контура. Параметрическая модель называется определенной, если она имеет набор геометрических элементов модели, которые полностью определяются ограничениями или являются постоянными и не имеют никаких дополнительных ограничений. При этом, возможно, существуют два и более реализуемых решений для набора элементов модели, которые полностью определены, но желаемый выбор обычно

ясен. Параметрическая модель называется недоопределенной, если она имеет набор геометрических элементов модели, который полностью не определен ограничениями и не является постоянным. Геометрический элемент модели, который недоопределен, имеет много возможных решений и неоднозначен, например окружность может иметь изменяемый радиус.

Избыточное ограничение — ограничение, применяемое к набору элементов модели, который уже является полностью определенным. Избыточное ограничение может входить, а может и не входить в противоречие с другими ограничениями. Степенью свободы (DOF — Degree of freedom) параметрической модели называется число, которое показывает степень неоднозначности (недоопределенности) параметрической модели. Обновлением ограничений называется решение алгебраических уравнений, описывающих модель для заданного набора ограничений и показ степеней свободы модели.

Пример параметрической модели, изображенной на рис. 4.58, состоит из пяти геометрических элементов моделирования: a , b , c , d и e . Объявленные ограничения между элементами на рисунке обозначены символически и подразумевают следующее:

- отрезок e горизонтален и касается дуги e ;
- отрезок a перпендикулярен e и имеет длину $a = 2,0$;
- отрезок b параллелен отрезку e и имеет длину b ;
- отрезок c связан с отрезком b общей точкой пересечения и касается дуги e ;
- линия d — круговая дуга, касательная к отрезкам c и e ;
- задано алгебраическое ограничение $b = a/2$;
- заданы точки пересечения отрезков e и a , а также a и b ;
- эта модель недоопределена и число степеней свободы равно двум, т. е. необходимо добавить еще два ограничения.

Большинство систем параметрического/вариационного геометрического моделирования используют оба метода одновременно. Такие модели называют параметрическими.

Зачатки параметризации, которую можно назвать жесткой, процедурной параметризацией, были реализованы еще в обычных чертежных системах типа AutoCAD. Эти системы сохраняют последовательность построения геометрических моделей в форме процедуры, закодированной в некотором внутреннем представлении. Фактически это ведет к представлению геометрической модели чертежа как последовательности определений геометрических элементов, где каждое определение в последовательности вычисляет значение параметров текущего элемента как функции параметров уже вычисленных элементов и некоторых входных параметров. В некоторых системах такую процедуру можно непосредственно отредактировать или же написать заново, используя ее текстовое описание, а затем повторно ее интерпретировать. В таких процедурных параметрических системах можно осуществлять только небольшой диапазон изменений геометрических моделей. Жесткая последовательность определений па-

раметров модели означает, что добавление новых типов зависимостей между элементами модели весьма затруднительно. Следовательно, многократное использование существующих моделей как основы получения новых подобных моделей в такой реализации практически невозможно.

В связи с этим были разработаны гибкие параметрические/вариационные системы геометрического моделирования, в которых жесткая последовательность определений параметров элементов заменена некоторой общей процедурой, которая не привязана к первоначальной последовательности операций геометрического моделирования, используемых при первоначальном построении геометрической модели. Эти системы используют некоторый тип декларативного представления ограничений вместо процедурного и используют графы ограничений и логику предикатов первого уровня. Для изменения модели в таких системах сначала редактируются ограничения, а затем выполняется общий алгоритм решения ограничений, чтобы разрешить отредактированные отношения между параметрами геометрических моделей элементов. Различие между параметрическими и вариационными методами заключается только в типе алгоритма решения ограничений, используемого этими системами.

Параметрические системы геометрического моделирования решают ограничения, выполняя последовательно назначения переменных модели, при этом каждое новое значение параметров геометрической модели вычисляется как функция предварительно назначенных значений параметров. В отличие от процедурных систем порядок назначений является гибким и определяется алгоритмом решения ограничений.

Вариационные системы геометрического моделирования выполняют ограничения путем формирования системы уравнений, представляющих ограничения и численного решения соответствующей системы нелинейных алгебраических уравнений.

Различие между параметрическими и вариационными системами можно продемонстрировать на следующих простых примерах определения переменной x :

- явное уравнение $x = \frac{-b \pm \sqrt{(b^2 - 4ac)}}{2a}$;
- неявное уравнение $ax^2 + bx + c = 0$.

С помощью явного уравнения можно непосредственно вычислять значение переменной x при заданных параметрах a , b и c и знаке квадратного корня. Параметрическая система работает, просматривая ограничения и выполняя предопределенные явные методы решения.

Напротив, вариационная система использует неявное уравнение, которое должно быть решено для x , при этом возможны несколько решений. Оба метода имеют преимущества и недостатки. Основанные на явном последовательном решении ограничений параметрические модели работают быстро. Однако они не

могут учесть взаимно связанные ограничения. С другой стороны, вариационные модели, используя неявные методы решения ограничений, могут обрабатывать связанные ограничения, но они менее быстрые и более ограничены при обработке не полностью определенных или противоречивых моделей.

4.2. Методы и алгоритмы трехмерной графики и геометрии

Алгоритмы создания трехмерных моделей рассмотрены в гл. 3. В литературе по КГ основное внимание уделяется задачам реалистического отображения трехмерных моделей, поэтому рассмотрим алгоритмические основы трехмерной графики в такой последовательности:

- алгоритмы визуализации трехмерных моделей;
- алгоритмы закрасивания видимых поверхностей;
- детальное отображение поверхностей;
- алгоритмы анимации трехмерных моделей.

4.2.1. Алгоритмы визуализации трехмерных моделей

После создания трехмерных геометрических моделей необходимо получить реалистическое отображение этих моделей, т. е. визуализировать их для проверки правильности полученных моделей. При этом необходимо определить, как разместить эти модели на сцене и выбрать место обзора с которого будет просматриваться эта сцена, т. е. должна быть выбрана определенная точка наблюдения. Также надо выбрать модель освещения — типы и количество источников света. При этом необходимо помнить, что основным назначением КГ является создание двухмерного изображения трехмерных объектов (оно должно быть двухмерным, так как выводится графически на плоский экран), но, принимая решения насчет объектов, которые будут нарисованы на экране, необходимо мыслить в трехмерных координатах. Будем предполагать, что все необходимые пространственные преобразования для визуализации сцены (поворот, параллельный перенос, масштабирование, зеркальное отражение, ортогональное или перспективное проецирование) выполнены. Поскольку визуализация сцены производится в прямоугольное окно, объекты (или части объектов), лежащие за пределами этого окна, должны отсекаться. В трехмерной КГ отсечение выполняется с помощью отбрасывания объектов, находящихся по одну сторону плоскости отсечения. Алгоритмы трехмерного отсечения (Сазерленда—Ходжмена, Вейлера—Айзертон и др.) во многом схожи с рассмотренными выше алгоритмами двухмерного отсечения.

Для реалистичного отображения простейших каркасных моделей необходимо решить задачу удаления невидимых линий, а для поверхностных и твердотельных моделей — задачу удаления невидимых поверхностей.

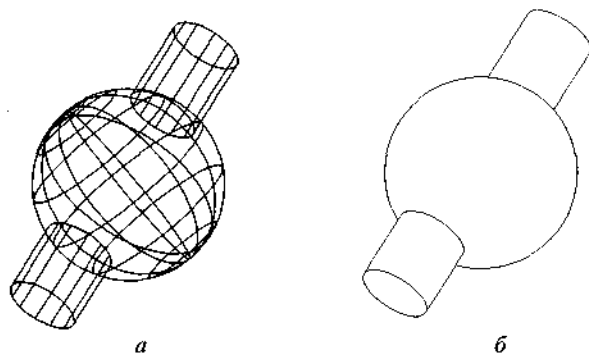


Рис. 4.59. Каркасная модель без удаления невидимых линий (а) и с удалением невидимых линий (б)

Алгоритмы удаления невидимых линий и поверхностей. Задача удаления невидимых линий и поверхностей является одной из наиболее сложных в машинной графике. Алгоритмы удаления невидимых линий и поверхностей служат для определения линий ребер, поверхностей или объемов, которые видимы или невидимы для наблюдателя, находящегося в заданной точке пространства. Необходимость удаления невидимых линий проиллюстрирована рис. 4.59.

На рис. 4.59, а приведена каркасная модель без удаления невидимых линий. Этот рисунок можно интерпретировать по-разному. Для этого достаточно перефокусировать глаза. Удаление тех линий или поверхностей, которые невидимы с соответствующей точки зрения, позволяет избавиться от неоднозначности. Результаты удаления невидимых линий показаны на рис. 4.59, б.

Все алгоритмы удаления невидимых линий (поверхностей) включают в себя сортировку. Порядок, в котором производится сортировка координат объектов, не влияет на эффективность этих алгоритмов. Главная сортировка ведется по геометрическому расстоянию от тела, поверхности, ребра или точки до точки наблюдения. Основная идея, положенная в основу сортировки по расстоянию, заключается в том, что чем дальше расположен объект от точки наблюдения, тем больше вероятность, что он будет полностью или частично заслонен одним из объектов, более близких к точке наблюдения. После определения расстояний или приоритетов по глубине остается провести сортировку по горизонтали и по вертикали, чтобы выяснить, будет ли рассматриваемый объект действительно заслонен объектом, расположенным ближе к точке наблюдения. Эффективность любого алгоритма удаления невидимых линий или поверхностей в большой мере зависит от эффективности алгоритма сортировки.

Алгоритмы удаления невидимых линий или поверхностей можно классифицировать по способу выбора системы координат или пространства. Алгоритмы, работающие в пространстве моделей, работают в системе координат, в которой описаны эти модели. Полученные изображения можно свободно масштабировать. Алгоритмы, работающие в пространстве изображения, имеют дело с сис-

темой координат того экрана, на котором визуализируются объекты моделирования. При этом точность вычислений ограничена разрешающей способностью экрана. Результаты, полученные в пространстве изображения, нельзя масштабировать вследствие искажений объекта моделирования.

Для любого алгоритма, работающего в пространстве моделей и сравнивающего каждый объект сцены со всеми остальными объектами этой сцены, объем вычислений растет пропорционально квадрату числа объектов. Аналогично для любого алгоритма, работающего в пространстве изображения и сравнивающего каждый объект сцены с позициями всех пикселей в системе координат экрана, объем вычислений растет как произведение числа объектов (тел, плоскостей или ребер) в сцене на число пикселей.

Рассмотрим алгоритмы, иллюстрирующие основополагающие идеи теории алгоритмов удаления невидимых линий и поверхностей.

Алгоритм плавающего горизонта. Алгоритм плавающего горизонта чаще всего используется для удаления невидимых линий трехмерного представления функций, описывающих поверхность в виде

$$f(x, y, z) = 0.$$

Этот алгоритм обычно работает в пространстве изображения. Главная идея данного метода заключается в сведении трехмерной задачи к двумерной путем пересечения исходной поверхности последовательностью параллельных секущих плоскостей, имеющих постоянные значения координаты z .

На рис. 4.60, *а* приведен пример, где указанные параллельные плоскости определяются постоянными значениями z — Функция $f(x, y, z) = 0$ сводится к последовательности кривых, лежащих в каждой из этих параллельных плоскостей, т. е. $y = f(x, z_i)$, где $z_i = \text{const}$ на каждой из заданных параллельных плоскостей. Поверхность теперь складывается из последовательности кривых, лежащих в каждой из этих плоскостей (см. рис. 4.60, *а*). Здесь предполагается, что полученные кривые являются однозначными функциями независимых переменных. Если спроецировать полученные кривые на плоскость $z = 0$, как показано на рис. 4.60, *б*, то становится

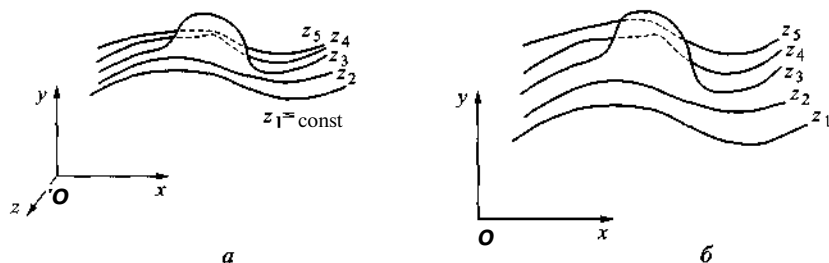


Рис. 4.60. Кривые отображаемой поверхности в секущих плоскостях с постоянной координатой z ;

а — в пространстве; *б* — в проекции на плоскость xy

ясна идея алгоритма удаления невидимых участков отображаемой поверхности. Алгоритм сначала упорядочивает плоскости $z = \text{const}$ по возрастанию расстояния до них от точки наблюдения. Затем для каждой плоскости, начиная с ближайшей к точке наблюдения, строится кривая, лежащая на ней, т. е. для каждого значения координаты x в пространстве изображения определяется соответствующее значение y .

Процесс удаления невидимой линии заключается в следующем. Если на текущей плоскости при некотором заданном значении x соответствующее значение y на кривой больше значения y для всех предыдущих кривых при этом значении x , то текущая кривая видима в этой точке; иначе она невидима.

Невидимые участки на рис. 4.60, б показаны пунктиром. Для хранения максимальных значений y при каждом значении x используется массив, длина которого равна числу различных точек (разрешению) по оси x в пространстве изображения. Значения, хранящиеся в этом массиве, представляют собой текущие значения «горизонта». Поэтому по мере рисования каждой очередной кривой этот «горизонт» как бы «всплывает». Фактически данный алгоритм удаления невидимых линий работает каждый раз с одной линией.

Рассматриваемый алгоритм работает очень хорошо до тех пор, пока какая-нибудь очередная кривая не окажется ниже самой первой из кривых, как показано на рис. 4.61, а. Эти кривые видимы и представляют собой нижнюю сторону исходной поверхности, однако алгоритм будет считать их невидимыми. Нижняя сторона поверхности становится видимой, если модифицировать этот алгоритм, включив в него «нижний горизонт», который опускается вниз по ходу работы алгоритма. Это реализуется при помощи другого массива, длина которого равна числу различных точек по оси x в пространстве изображения. Этот массив содержит наименьшие значения y для каждого значения x . Процесс удаления невидимой линии имеет такой вид. Если на текущей плоскости при некотором заданном значении x соответствующее значение y на кривой больше максимума или меньше минимума по y для всех предыдущих кривых и при этом значении x ,

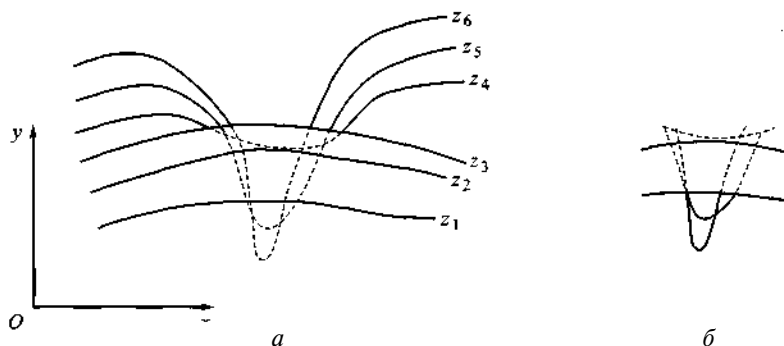


Рис. 4.61. Обработка нижней стороны поверхности:

а — исходная поверхность; б — нижняя сторона исходной поверхности

то текущая кривая видима. В противном случае она невидима. Полученный результат показан на рис. 4.61, б.

В алгоритме «плавающего горизонта» предполагается, что значение функции, т. е. y , известно для каждого значения x в пространстве изображения. Однако если для некоторого значения нет соответствующего ему значения y , то невозможно поддерживать массивы верхнего и нижнего «плавающих горизонтов». В таком случае используется линейная интерполяция значений y между известными значениями для того, чтобы заполнить массивы верхнего и нижнего «плавающих горизонтов».

Алгоритм Робертса. Алгоритм Робертса — наиболее известный алгоритм, работающий в пространстве объекта. Алгоритм Робертса прежде всего удаляет из каждого тела те ребра или грани, которые экранируются самим телом (нелицевые грани).

При использовании алгоритма Робертса требуется, чтобы все изображаемые тела или объекты были выпуклыми. Невыпуклые тела должны быть разбиты на выпуклые части. В этом алгоритме выпуклое многогранное тело с плоскими гранями должно представляться набором пересекающихся плоскостей. Неплоские грани аппроксимируются плоскими. Уравнение произвольной плоскости в трехмерном пространстве имеет следующий вид:

$$Ax + By + Cz + D = 0.$$

В матричной форме этот результат имеет вид

$$[xyz1]/[P]^T = 0,$$

где $[P]^T = [A \ B \ C \ D]$ — плоскость. Поэтому любое выпуклое твердое тело можно выразить матрицей тела V , состоящей из коэффициентов уравнений плоскостей, т. е.

$$V = \begin{bmatrix} A_1 & A_2 & \dots & A_n \\ B_1 & B_2 & \dots & B_n \\ C_1 & C_2 & \dots & C_n \\ D_1 & D_2 & \dots & D_n \end{bmatrix},$$

где каждый столбец содержит коэффициенты одной плоскости.

Любая точка пространства представима в однородных координатах вектором $S = [x \ y \ z \ 1]$. Если точка S лежит на плоскости P , то скалярное произведение $[S] \cdot [P] = 0$. Если же эта точка не лежит на плоскости, то знак этого скалярного произведения показывает, по какую сторону от плоскости расположена точка. В алгоритме Робертса предполагается, что точки, лежащие внутри тела, дают положительное скалярное произведение.

Тот факт, что плоскости имеют бесконечную протяженность и что скалярное произведение точки на матрицу тела отрицательно, если точка лежит вне этого тела, позволяет предложить способ определения граней, которые экранируются самим этим телом с помощью матрицы тела. Отрицательное скалярное произведение дает только такая плоскость (столбец) в матрице тела, относительно которой точка лежит снаружи. Если точка наблюдения находится в бесконечности на положительной полуоси z и взгляд направлен в сторону отрицательной полуоси z , то в однородных координатах вектор такого направления равен $E = [0 \ 0 \ -1 \ 0]$, который служит, кроме того, образом точки, лежащей в бесконечности на отрицательной полуоси z . Фактически E представляет собой любую точку, лежащую на плоскости $z = -\infty$, т. е. любую точку типа $(x, y, -\infty)$. Поэтому если скалярное произведение E на столбец, соответствующий какой-нибудь плоскости в матрице тела, отрицательно, то E лежит по отрицательную сторону этой плоскости. Следовательно, эти плоскости невидимы из любой точки наблюдения, лежащей в плоскости $z = \infty$, а пробная точка на $z = -\infty$ экранируется самим телом. Такие плоскости называются нелицевыми, а соответствующие им грани — задними. Следовательно, условие $S \cdot P < 0$ является условием того, что плоскости — нелицевые, а их грани — задние.

Алгоритм Робертса — простейший алгоритм удаления невидимых граней для тел, представляющих собой одиночные выпуклые многогранники. Он также используется для удаления нелицевых, или задних, граней из сцены перед применением более сложных алгоритмов удаления невидимых линий. Его часто называют отбрасыванием задних плоскостей. Для выпуклых многогранников число граней при этом сокращается примерно вдвое. Этот алгоритм эквивалентен вычислению нормали к поверхности для каждого отдельного многоугольника. Отрицательность нормали к поверхности показывает, что нормаль направлена в сторону от наблюдателя и, следовательно, данный многоугольник не виден. После удаления нелицевых граней, экранированных самими телами, все видимые ребра каждого тела сравниваются с ребрами оставшихся тел для определения того, какая его часть или части, если таковые есть, экранируются этими телами. Поэтому вычислительная трудоемкость алгоритма Робертса пропорциональна квадрату числа объектов. Известны реализации этого алгоритма, использующие предварительную приоритетную сортировку вдоль оси z и простые габаритные, или мини-максные, тесты, демонстрирующие почти линейную зависимость от числа объектов.

Алгоритм, использующий z-буфер. Это один из простейших алгоритмов удаления невидимых поверхностей. Работает этот алгоритм в пространстве изображения. Идея z-буфера является простым обобщением идеи о буфере кадра. Буфер кадра служит для запоминания интенсивности каждого пиксела в пространстве изображения, z-буфер — отдельный буфер глубины, используемый для запоминания координаты z или глубины каждого видимого пиксела в пространстве изображения. В процессе работы глубина или значение z каждого но-

вого пиксела, который нужно занести в буфер кадра, сравнивается с глубиной того пиксела, который уже занесен в z-буфер. Если это сравнение показывает, что новый пиксел расположен впереди пиксела, находящегося в буфере кадра, то новый пиксел заносится в этот буфер и, кроме того, производится корректировка z-буфера новым значением z . Если же сравнение дает противоположный результат, то никаких действий не производится. По сути, алгоритм является поиском по x и y наибольшего значения функции $z(x, y)$.

Главным преимуществом алгоритма является его простота. Кроме того, этот алгоритм решает задачу удаления невидимых поверхностей и делает тривиальной визуализацию пересечений сложных поверхностей. Сцены могут быть любой сложности. Поскольку размеры пространства изображения фиксированы, вычислительная трудоемкость алгоритма будет линейной. Так как элементы сцены или картинки можно заносить в буфер кадра или в z-буфер в произвольном порядке, их не нужно предварительно сортировать по приоритету глубины, следовательно, экономится вычислительное время, затрачиваемое на сортировку по глубине.

Основной недостаток алгоритма — большой объем требуемой памяти. Однако снижение цен на память делает экономически оправданным создание специализированных запоминающих устройств для z-буфера и связанной с ним аппаратуры.

Другой недостаток алгоритма z-буфера состоит в трудоемкости и высокой стоимости устранения лестничного эффекта, а также реализации эффектов прозрачности. Поскольку алгоритм заносит пикселы в буфер кадра в произвольном порядке, то нелегко получить информацию, необходимую для методов устранения лестничного эффекта, основывающихся на предварительной фильтрации. При реализации эффектов прозрачности пикселы могут заноситься в буфер кадра в некорректном порядке, что ведет к локальным ошибкам.

Формальное описание алгоритма z-буфера имеет следующий вид:

- 1) заполнить буфер кадра фоновым значением интенсивности или цвета;
- 2) заполнить z-буфер минимальным значением z ;
- 3) преобразовать каждый многоугольник в растровую форму в произвольном порядке;
- 4) для каждого значения (x, y) пиксела в многоугольнике вычислить его глубину $z(x, y)$. Сравнить глубину $z(x, y)$ со значением z , хранящимся в z-буфере в этой же позиции;
- 5) если $z(x, y) > z$ буфера, то записать атрибуты этого многоугольника (интенсивность, цвет и т. п.) в буфер кадра и заменить z-буфер в этой позиции на $z(x, y)$. В противном случае никаких действий не производить.

В качестве предварительного шага там, где это целесообразно, применяется удаление нелицевых граней, рассмотренное выше.

Алгоритм художника. При реализации всех обсуждавшихся алгоритмов удаления невидимых линий и поверхностей устанавливались приоритеты, т. е.

глубины объектов сцены или их расстояния от точки наблюдения. Алгоритмы, использующие список приоритетов, применяют предварительную сортировку по глубине или приоритету. Цель такой сортировки состоит в том, чтобы получить окончательный список элементов сцены, упорядоченных по приоритету глубины, основанному на расстоянии от точки наблюдения. Если такой список окончателен, то никакие два элемента не будут взаимно перекрывать друг друга, и можно записать все элементы в буфер кадра поочередно, начиная с элемента, наиболее удаленного от точки наблюдения. Более близкие к наблюдателю элементы будут заменять информацию о более далеких элементах в буфере кадра. Поэтому задача об удалении невидимых поверхностей решается тривиально.

Для простых элементов сцены, например для многоугольников, этот метод иногда называют алгоритмом художника, поскольку он аналогичен тому способу, которым художник создает картину. Сначала художник рисует фон, затем предметы, лежащие на среднем расстоянии, и, наконец, передний план. Тем самым художник решает задачу об удалении невидимых поверхностей, или задачу видимости, путем построения картины в порядке обратного приоритета.

Алгоритмы, использующие список приоритетов, работают как в пространстве объекта, так и в пространстве изображения. В частности, формирование списка приоритетов ведется в пространстве объекта, а результат заносится в буфер кадра в терминах пространства изображения. Поскольку подобно алгоритму z-буфера в алгоритмах, строящих список приоритетов, многоугольники обрабатываются в произвольном порядке, применение методов устранения лестничного эффекта к результирующему изображению затруднено. Однако для этой цели здесь применим метод постфильтрации, используемый также в алгоритмах z-буфера.

Алгоритмы, строящие список приоритетов и использующие z-буфер, могут также применяться и для удаления невидимых линий. При этом ребра каждого многоугольника заносятся в буфер кадра с одним атрибутом, а внутренняя область каждого многоугольника заносится в буфер кадра с атрибутом фона. При таком подходе многоугольники, которые находятся ближе к точке наблюдения, заслоняют ребра многоугольников, которые расположены дальше от нее.

Алгоритмы построчного сканирования. Алгоритмы z-буфера и алгоритмы, строящие список приоритетов, обрабатывают элементы сцены или многоугольники в порядке, который не связан с процессом визуализации. В то же время алгоритмы построчного сканирования обрабатывают сцену в порядке прохождения сканирующей прямой. Эти алгоритмы оперируют в пространстве изображения.

Растровая развертка отдельных многоугольников рассматривалась выше. Алгоритмы удаления невидимых линий и поверхностей с построчным сканированием являются обобщением изложенных методов. Эти алгоритмы сводят трехмерную задачу удаления невидимых линий и поверхностей к двухмерной. Сканирующая плоскость определяется точкой наблюдения, расположенной в бесконечности на положительной полуоси z , и сканирующей прямой, как показано на рис. 4.62.

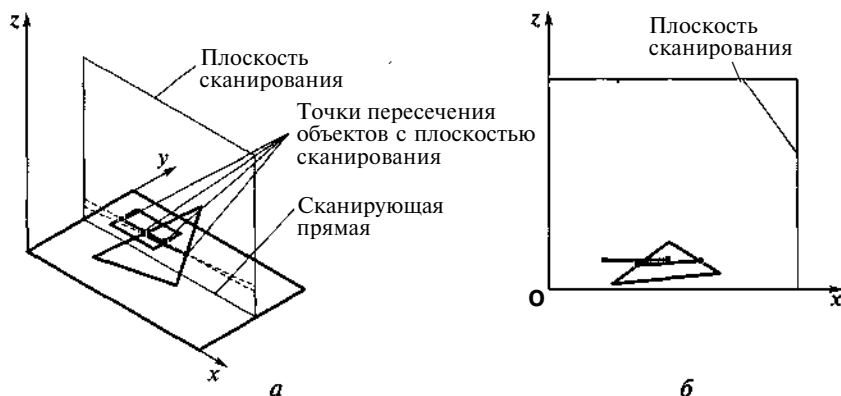


Рис. 4.62. Сканирующая плоскость $y = \text{const}$ (а) и пересечение сканирующей плоскости с многоугольниками (б)

Пересечение сканирующей плоскости и трехмерной сцены определяет окно размером в одну сканирующую строку. Задача удаления невидимых поверхностей решается в пределах этого окна, образованного сканирующей плоскостью. На рис. 4.62, б представлено пересечение сканирующей плоскости с многоугольниками (прямоугольником и треугольником). Этот рисунок показывает, что задача удаления невидимых поверхностей сводится к задаче о том, какой отрезок видим для каждой точки сканирующей строки.

Одним из простейших алгоритмов построения сканирования, который решает задачу удаления невидимых поверхностей, является разновидность алгоритма z-буфера, рассмотренного выше. Это алгоритм построения сканирования с z-буфером. Используемое в нем окно визуализации имеет высоту в одну сканирующую строку и ширину во весь экран. Для каждой сканирующей строки буфер кадра инициализируется с фоновым значением интенсивности, а z-буфер — с минимальным значением z . Затем определяется пересечение сканирующей строки с двухмерной проекцией каждого многоугольника сцены, если они существуют. Эти пересечения образуют пары точек пересечения. При рассмотрении каждого пиксела на сканирующей строке в интервале между концами пар его глубина сравнивается с глубиной, содержащейся в соответствующем элементе z-буфера. Если глубина этого пиксела больше глубины из z-буфера, то рассматриваемый отрезок будет текущим видимым отрезком. И следовательно, атрибуты многоугольника, соответствующего данному отрезку, заносятся в буфер кадра в позиции данного пиксела; соответственно корректируется и z-буфер в этой позиции. После обработки всех многоугольников сцены буфер кадра размером в одну сканирующую строку содержит решение задачи удаления невидимых поверхностей для данной сканирующей строки. Он выводится на экран дисплея в порядке, определяемом растровым сканированием, т. е. слева направо. В этом алгоритме можно использовать методы устранения ступенчатости, основывающиеся как на пре-, так и на постфильтрации.

Однако практически сравнение каждого многоугольника с каждой сканирующей строкой оказывается неэффективным. Поэтому используется некоторая разновидность списка упорядоченных ребер, которая обсуждалась выше. В частности, для повышения эффективности этого алгоритма применяются групповая сортировка по оси u , список активных многоугольников и список активных ребер.

Алгоритм трассировки лучей. Трассировка лучей — рейтресинг (ray tracing), также известная как выбрасывание лучей — рейкастинг (ray casting), определяет видимость поверхностей, прослеживая воображаемые лучи света от глаза наблюдателя до объектов в сцене. Главная идея, лежащая в основе этого алгоритма, заключается в том, что наблюдатель видит любой объект посредством испускаемого неким источником света, который падает на этот объект и затем каким-то путем доходит до наблюдателя. Свет может достичь наблюдателя, отразившись от поверхности, преломившись или пройдя через нее. Если проследить за лучами света, испускаемыми источником света, то можно убедиться, что весьма немногие из них дойдут до наблюдателя, следовательно, этот процесс вычислительно неэффективен. Однако можно отслеживать (трассировать) лучи в обратном направлении, т. е. от наблюдателя к объекту. Выбираются центр проецирования (глаз наблюдателя) и окно вида на произвольной плоскости. Окно вида можно представить в виде правильной сетки, элементы которой соответствуют пикселям в соответствующей разрешающей способности. Тогда для каждого пикселя в окне луч света запускается из центра проецирования через центр пикселя в сцену, как показано на рис. 4.63.

Полный алгоритм, учитывающий эффекты отражения одного объекта от поверхности другого, преломления, прозрачности и затенения, обсуждается далее. Рассмотрим алгоритм трассировки лучей для определения видимых или скрытых поверхностей. Рис. 4.63 является иллюстрацией алгоритма трассировки лучей с перспективной проекцией. В простейшем алгоритме предполагается, что точка зрения, или наблюдатель, находится в бесконечности на положительной полуоси z , поэтому все световые лучи параллельны оси z . Каждый луч, исходящий от наблюдателя, проходит через центр пикселя до сцены. Траектория каж-

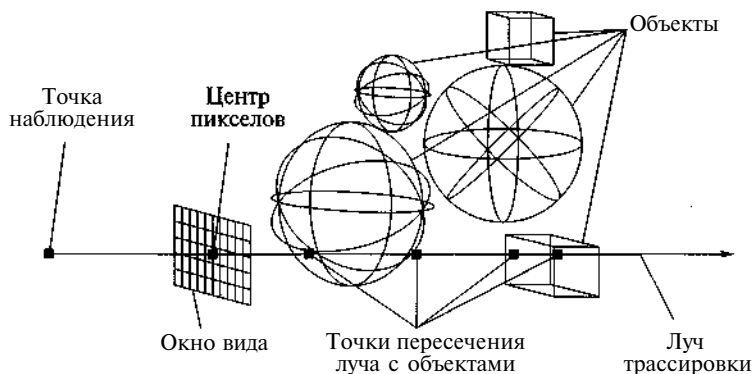


Рис. 4.63. Трассировка луча из точки наблюдения

дого луча отслеживается, чтобы определить, какие именно объекты сцены, если таковые существуют, пересекаются с данным лучом. Необходимо проверить пересечение каждого объекта сцены с каждым лучом. Если луч пересекает объект, то определяются все возможные точки пересечения луча и объекта. Можно получить большое количество пересечений, если рассматривать много объектов. Эти пересечения упорядочиваются по глубине. Пересечение с максимальным значением z представляет видимую поверхность для данного пиксела. Атрибуты этого объекта используются для определения характеристик пиксела.

Наиболее важным элементом алгоритма определения видимых поверхностей путем трассировки лучей является процедура определения пересечений. В состав сцены можно включать любой объект, для которого можно создать процедуру построения пересечений. Объекты сцены могут состоять из набора плоских многоугольников, многогранников или тел, ограниченных или определяемых квадратичными или В-сплайнными параметрическими поверхностями. Вычислительная стоимость определения пересечений произвольной пространственной прямой (лучом) с одним выделенным объектом может оказаться высокой. Чтобы избавиться от ненужного поиска пересечений, выполняется проверка пересечения луча с объемной оболочкой рассматриваемого объекта. И если луч не пересекает оболочки, то не нужно больше искать пересечений этого объекта с лучом. В качестве оболочки можно использовать прямоугольный параллелепипед или сферу, показанные на рис. 4.64.

Факт пересечения трехмерного луча со сферой определяется просто. В частности, если расстояние от центра сферической оболочки до луча превосходит радиус этой сферы, то луч не пересекает оболочки. Следовательно, он не может пересечься и с объектом. Поэтому тест со сферической оболочкой сводится к определению расстояния от точки до трехмерной прямой, т. е. луча. Будем использовать параметрическое представление прямой, проходящей через точки $P_1(x_1, y_1, z_1)$ и $P_2(x_2, y_2, z_2)$ в виде функции $P(t) = P_1 + (P_2 - P_1)t$ с компонентами

$$x = x_1 + (x_2 - x_1)t = x_1 + at;$$

$$y = y_1 + (y_2 - y_1)t = y_1 + bt;$$

$$z = z_1 + (z_2 - z_1)t = z_1 + ct.$$

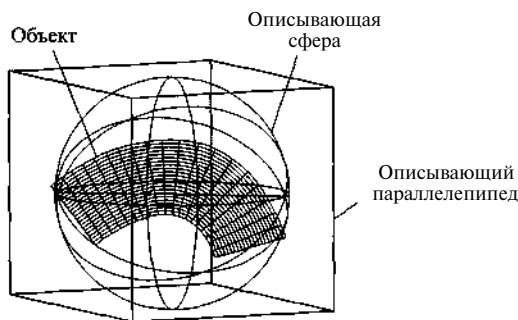


Рис. 4.64. Сферическая и прямоугольная оболочки

Тогда параметр t , определяющий ближайшую точку $P(t)$, равен

$$t = -\frac{a(x_1 - x_0) + b(y_1 - y_0) + c(z_1 - z_0)}{a^2 + b^2 + c^2}.$$

Если расстояние d от прямой до этой точки, определяемое из соотношения

$$d^2 = (x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2,$$

больше радиуса сферической оболочки, то луч не может пересечься с объектом.

Выполнение габаритного теста с прямоугольной оболочкой в трехмерном пространстве требует большого объема вычислений. При этом следует проверить пересечение луча по меньшей мере с тремя бесконечными плоскостями, ограничивающими прямоугольную оболочку. Поскольку точки пересечения могут оказаться вне граней этого параллелепипеда, то для каждой из них следует, кроме того, провести проверку на охват или попадание внутрь параллелепипеда. Следовательно, тест с прямоугольной оболочкой оказывается более медленным, чем тест со сферической оболочкой.

Вычисления пересечений для параметрических поверхностей более сложны. Анализ проводится для элементов поверхности. Если луч пересекает сферическую оболочку элемента поверхности, то этот кусок разбивается на подэлементы. Затем луч проверяется на пересечение со сферическими оболочками подэлементов. Если луч пересекается со сферической оболочкой какого-нибудь подэлемента, то последний разбивается также на подэлементы. Процесс завершается, если ни одна из сферических оболочек не пересечена или если достигнут заранее определенный их минимальный размер. Эти сферические оболочки минимального размера и являются искомыми пересечениями луча и элемента поверхности.

4.2.2. Алгоритмы закрашивания видимых поверхностей

Рассмотрим сначала модели освещения поверхностей. Световая энергия, падающая на поверхность, может быть поглощена, отражена или пропущена. Частично она поглощается и превращается в тепловую энергию, а частично отражается или пропускается. Объект можно увидеть, только если он отражает или пропускает свет; если же объект поглощает весь падающий свет, то он будет невидим, и в этом случае его называют абсолютно черным телом. Количество поглощенной, отраженной или пропущенной энергии зависит от длины волны света. При освещении белым светом, в котором интенсивность всех длин волн снижена примерно одинаково, объект выглядит серым. Если поглощается почти весь свет, то объект кажется черным, а если только небольшая его часть — белым. Если поглощаются лишь определенные длины волн, то у света, исходящего от объекта, изменяется распределение энергии и объект выглядит цветным. Цвет объекта определяется поглощаемыми длинами волн. Свойства отраженного све-

та зависят от строения, направления и формы источника света, от ориентации и свойств поверхности. Это учитывается в различных моделях освещения.

Самая простая модель освещения отображает рассеянный свет: каждый объект визуализируется, испуская свет равномерной интенсивности со своей поверхности, т. е. без внешнего источника освещения, — это довольно нереалистичный мир самосветящихся объектов. Каждый объект появляется как некий монохроматический силуэт, если только его индивидуальным частям (типа многоугольников поверхности) не задать различных оттенков при создании этого объекта. Модель освещения может быть выражена уравнением освещения в переменных, связанных с точкой на закрашиваемом объекте:

$$I = I_a + k_i I_s,$$

где I — итоговая интенсивность света; k_i — коэффициент встроенной интенсивности объекта.

Поскольку это уравнение для интенсивности освещения не содержит членов, которые зависят от позиции закрашиваемой точки, то можно оценить ее сразу для каждого объекта.

Предположим, что есть рассеянный, ненаправленный источник света — продукт множественных отражений света от многих поверхностей, существующих в среде. Такой свет называется рассеянным светом. Если принять, что рассеянный свет освещает одинаково все поверхности во всех направлениях, то уравнение для интенсивности освещения принимает следующий вид:

$$I = I_a + k_a I_s,$$

где I_a — интенсивность рассеянного света, постоянная для всех объектов; k_a — коэффициент рассеяния (количество рассеянного света, отраженного от поверхности объекта).

Коэффициент рассеяния — свойство поверхности материала, он не соответствует непосредственно никакому физическому свойству реальных материалов и служит только для целей визуализации объектов, которые иначе нельзя показать.

Рассмотрим освещение объекта точечным источником света, лучи которого идут однородно во всех направлениях из единственной точки. Яркость объекта при этом изменяется от одной части объекта до другой в зависимости от направления и расстояния от источника света.

Свойства отраженного света зависят от строения, направления и формы источника света, от ориентации и свойств поверхности. Отраженный от объекта свет может быть диффузным или зеркальным. Диффузное отражение света происходит, когда свет как бы проникает под поверхность объекта, поглощается, а затем вновь испускается. При этом положение наблюдателя не имеет значения, так как диффузно отраженный свет рассеивается равномерно по всем направлениям. Зеркальное отражение происходит от внешней поверхности объекта.

Свет точечного источника диффузно отражается от идеальной поверхности по закону косинусов Ламберта: интенсивность отраженного света пропорциональна косинусу угла между направлением света и нормалью к поверхности, т. е.

$$I = I_0 \cos(\theta); \quad 0 \leq \theta \leq \pi/2,$$

где I — интенсивность отраженного света; I_0 — интенсивность точечного источника в направлении L ; k_d — коэффициент диффузного отражения; θ — угол между направлением света L и нормалью к поверхности n (рис. 4.65), $0 \leq \theta \leq \pi/2$. Если $\theta > \pi/2$, то источник света расположен за объектом. Коэффициент диффузного отражения k_d зависит от материала и длины волны света, но в простых моделях освещения обычно считается постоянным.

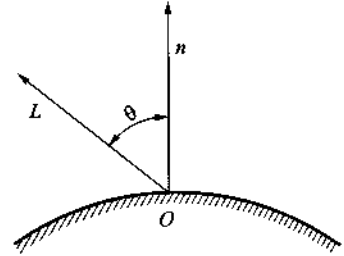


Рис. 4.65. Диффузное отражение света

Поверхность предметов, изображенных при помощи простой модели освещения с ламбертовым диффузным отражением, выглядит блеклой и матовой. Поскольку источник света точечный, объекты, на которые не падает прямой свет, кажутся черными. Однако на объекты реальных сцен падает еще и рассеянный свет, рассмотренный выше, поэтому обычно используется простая модель с интенсивностью освещения

$$I = I_0 k_d + I_a k_d \cos(\theta); \quad 0 \leq \theta \leq \pi/2.$$

Пусть даны два объекта, одинаково ориентированные относительно источника света, но расположенные на разном расстоянии от него. Если найти их интенсивность по данной формуле, то она окажется одинаковой. Это значит, что когда предметы перекрываются, их невозможно различить, хотя интенсивность света обратно пропорциональна квадрату расстояния от источника света, и объект, лежащий дальше от него, должен быть темнее. Если предположить, что источник света находится в бесконечности, то диффузный член модели освещения обратится в нуль. В случае перспективного преобразования сцены в качестве коэффициента пропорциональности для диффузного члена можно взять расстояние d от центра проекции до объекта. Но если центр проекции лежит близко к объекту, то $1/d^2$ изменяется очень быстро, т. е. у объектов, лежащих примерно на одинаковом расстоянии от источника света, разница интенсивностей чрезмерно велика. Большей реалистичности можно добиться при линейном затухании. В этом случае модель освещения выглядит следующим образом:

$$I = I_0 k_d + \frac{I_a k_d \cos(\theta)}{d + K}; \quad 0 \leq \theta \leq \pi/2,$$

где K — произвольная константа.

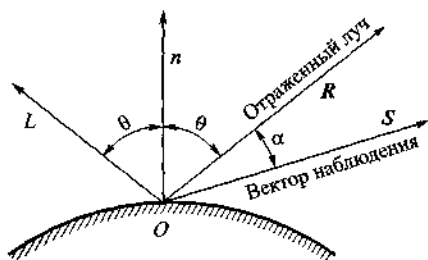


Рис. 4.66. Зеркальное отражение

Если предположить, что точка наблюдения находится в бесконечности, то d определяется положением объекта, ближайшего к точке наблюдения. Это означает, что ближайший объект освещается с полной интенсивностью источника света, а более далекие — с уменьшенной. Для цветных поверхностей данная модель освещения применяется к каждому из трех основных цветов.

Интенсивность зеркально отраженного света зависит от угла падения, длины волны падающего света и свойств вещества. Свойства зеркального отражения описываются уравнением Френеля из геометрической оптики.

Зеркальное отражение света является направленным. Угол отражения θ от идеальной отражающей поверхности (зеркала) равен углу падения θ , в любом другом положении наблюдатель не видит зеркально отраженный свет. Это означает, что вектор наблюдения S (рис. 4.66) совпадает с вектором отражения R и угол α равен нулю. Если поверхность не идеальна, то количество света, достигающее наблюдателя, зависит от пространственного распределения зеркального отраженного света. У гладких поверхностей распределение узкое или сфокусированное, у шероховатых — более широкое.

Благодаря зеркальному отражению на блестящих предметах появляются световые блики. Вследствие того, что зеркально отраженный свет сфокусирован вдоль вектора отражения, блики при движении наблюдателя тоже перемещаются. Более того, так как свет отражается от внешней поверхности (за исключением металлов и некоторых твердых красителей), то отраженный луч сохраняет свойства падающего. Например, при освещении блестящей синей поверхности белым светом возникают белые, а не синие блики.

В простых моделях освещения обычно используют эмпирическую модель Буи-Туонга Фонга, так как реальные физические свойства зеркального отражения очень сложны. Эта модель имеет следующий вид:

$$I_s = I_0 \cos^n(\alpha),$$

где I_s — интенсивность зеркально отраженного света; $\cos^n(\alpha)$ — кривая отражения, представляющая отношение зеркально отраженного света к падающему как функцию угла падения α и длины волны λ ; n — степень, аппроксимирующая пространственное распределение зеркально отраженного света.

На рис. 4.67 показана функция $\cos^n(\alpha)$ при $0 < \alpha < \pi/2$ для различных n : большие значения n дают сфокусированные пространственные распределения характеристик металлов и других блестящих поверхностей, а малые — более широкие распределения для неметаллических поверхностей, например бумаги.

Коэффициент зеркального отражения зависит от угла падения, однако даже при перпендикулярном падении зеркально отражается только часть света, а остальное либо поглощается, либо отражается диффузно. Эти соотношения определяются свойствами вещества и длиной волны. Коэффициент отражения для некоторых неметаллов может составлять всего 4 %, в то время как для металлических материалов — более 80 %. Функция $\rho(0, X)$ довольно

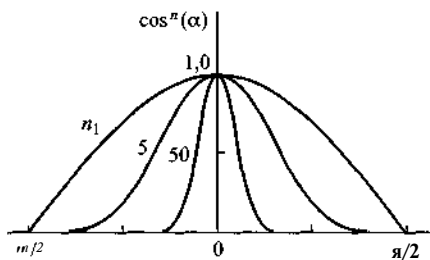


Рис. 4.67. Пространственное распределение зеркального отражения

сложная, поэтому ее обычно заменяют константой k_s , которая определяется экспериментально. Таким образом, модель интенсивности освещения можно представить в следующем виде:

$$I = I_s \cdot \frac{I_l}{d + K} \left[k_d \cos(\theta) + k_s \cos^n(\alpha) \right].$$

В машинной графике эта модель называется функцией закраски и применяется для расчета интенсивности или тона точек объекта, или пикселей изображения. Чтобы получить цветное изображение, нужно найти функции закраски для каждого из трех основных цветов. Константа k_s обычно одинакова для всех трех основных цветов, поскольку цвет зеркально отраженного света определяется цветом падающего.

Если имеется несколько источников света, то их эффекты суммируются. В этом случае модель интенсивности освещения имеет вид

$$I = I_a k_a + \sum_{j=1}^m \frac{I_{l_j}}{d + K} \left[k_d \cos(\theta_j) + k_s \cos^n(\alpha_j) \right],$$

где m — число источников света.

Рассмотрим модели закрашивания для многоугольников поверхностей. Ясно, что можно закрасить любую поверхность, вычисляя нормаль поверхности в каждой видимой точке и применяя желаемую модель освещения в этой точке. К сожалению, эта модель закрашивания требует больших затрат машинного времени.

Закраска с постоянной интенсивностью. Самая простая модель закрашивания для многоугольника — закрашка с постоянной интенсивностью, известная также как фасетное закрашивание, или равномерное затенение. Этот подход рассчитывает модель освещения один раз для любой точки каждого многоугольника так, что рассчитанное значение интенсивности используется для закрашивания всего многоугольника. Данный подход справедлив при следующих допущениях:

- источник освещения находится в бесконечности, так что угол между нормалью к поверхности N и направлением источника света L является постоянным во всех точках грани многоугольника;
- наблюдатель находится в бесконечности, так что угол между нормалью к поверхности N и направлением взгляда V является постоянным во всех точках грани многоугольника;
- многоугольник представляет фактическую поверхность, а не ее приближение.

Если любое из первых двух предположений не выполняется, то нужно определить единственное значение для каждого из значений L и V . Например, эти значения могут быть рассчитаны для центра многоугольника или для первой вершины многоугольника. Конечно, закрашка с постоянной интенсивностью в этом случае будет не совсем правильная.

Интерполированное закрашивание. Как альтернативу оценке освещенности в каждой точке многоугольника можно использовать интерполированное закрашивание, при котором информация закрашивания многоугольника линейно интерполируется относительно значений, определенных для его вершин. Это выполнять особенно просто для алгоритма построчного сканирования изображения, который уже интерполирует значение z конечных точек промежутка сканирования.

Данный метод хорошо подходит для многогранников, но многие из поверхностей на практике криволинейны, поэтому он дает не совсем правильные результаты.

Предположим, что мы моделируем криволинейную поверхность сетью многоугольников. Если каждая грань многоугольника в сети закрашивается индивидуально, то она отличается от соседних, ориентация которых различна, производя фасетное изображение. Это справедливо как для закрашивания с постоянной интенсивностью, так и для интерполированного закрашивания потому, что два смежных многоугольника различной ориентации имеют разные интенсивности по их границам. Простое решение использования более мелкой сети оказывается крайне неэффективным, поскольку различие в закрашивании между смежными фасетами еще более подчеркивается эффектом полос Маха, который преувеличивает изменение интенсивности в любой границе, где есть неоднородность по величине интенсивности или направлению интенсивности. На границе между двумя фасетами темная фасета выглядит более темной, а светлая более светлой. Эффект полос Маха еще более заметен для цветных изображений.

Модели закрашивания многоугольников, описанные выше, определяют оттенок каждого многоугольника индивидуально. Две основные модели закрашивания для сетей многоугольников используют информацию смежных многоугольников, чтобы моделировать гладкость криволинейных поверхностей. В порядке увеличивающейся сложности (и реалистичности) они известны как закрашивание по методу Гуро и закрашивание по методу Фонга, по именам ученых, которые их разработали. Оба эти метода сейчас поддерживаются на аппаратном уровне систем машинной графики.

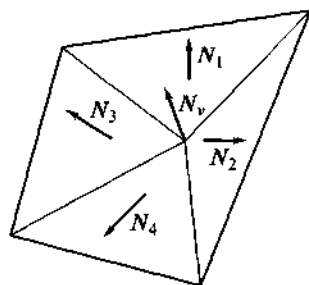


Рис. 4.68. Усреднение нормалей по методу Гуро

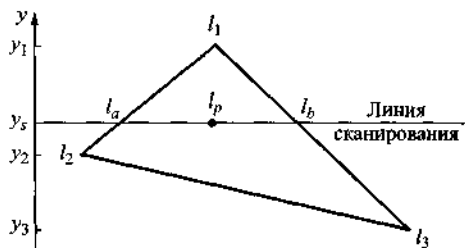


Рис. 4.69. Интерполяция интенсивности света относительно ребер многоугольника и вдоль линии сканирования:

I_a, I_p, I_b — интенсивности освещения вдоль линии сканирования

Закрашивание по методу Гуро. Закрашивание по методу Гуро обеспечивает непрерывность интенсивности света через границы многоугольников. Процесс закрашивания по методу Гуро требует, чтобы нормали были известны для каждой вершины сети многоугольников. Гуро вычислял эти нормали вершин непосредственно из аналитического описания поверхности. Если нормали вершин не сохраняются и не могут быть определены непосредственно из фактической поверхности, то можно определять их приблизительно как среднее значение нормалей всех фасет многоугольников, совместно использующих каждую вершину (рис. 4.68).

Следующий шаг в закрашивании по методу Гуро определяет интенсивности света вершин, используя нормали вершин с любой желаемой моделью освещения. Наконец, каждый многоугольник закрашивается линейной интерполяцией интенсивностей света вершин для каждого ребра и затем между гранями по каждой строке сканирования вышеописанным способом. На рис. 4.69 представлена интерполяция интенсивности света относительно ребер многоугольника и вдоль линии сканирования. Здесь $I_a = I_1 - (I_1 - I_2) \frac{y_1 - y_s}{y_1 - y_2}$; $I_b = I_1 - (I_1 - I_3) \frac{y_1 - y_s}{y_1 - y_3}$;

$$I_p = I_b - (I_b - I_a) \frac{y_b - y_p}{y_b - y_a}.$$

Закрашивание по методу Фонга. Закрашивание по методу Фонга интерполирует нормали поверхности (N_a, N_b, N_c), а не интенсивность света вершин. Интерполяция происходит на промежутке многоугольника вдоль строки сканирования между начальными и конечными нормальными для этого промежутка. Эти нормали сами интерполируются по граням многоугольника относительно нормалей вершин, которые вычисляются в случае необходимости, как и в

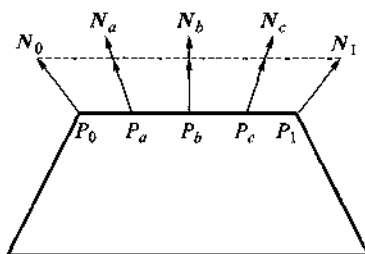


Рис. 4.70. Интерполяция векторов нормалей

закрашивании по методу Гуро. В каждом пикселе по строке сканирования интерполированная нормаль и новое значение интенсивности пиксела выполняется, используя любую желаемую модель освещения. На рис. 4.70 изображены две крайние нормали ребра многоугольника LM и IM и нормали, интерполированные по ним до и после нормализации.

4.2.3. Детальное отображение поверхностей

Описанные выше методы отображают гладкие, однородные поверхности. Рассмотрим методы, разработанные для детального моделирования поверхностей — текстурирование, взаимные отражения от поверхностей (рейтресинг и радиосити).

Наложение текстуры. Для отображения фактуры поверхностей явное моделирование многоугольниками или другими геометрическими примитивами неприменимо. Существует альтернативный подход, известный как отображение шаблона или наложение текстуры; изображение называют картой текстуры, а ее индивидуальные элементы — элементами текстуры. Прямоугольная карта текстуры постоянно хранится в ее собственном (s, t) координатном пространстве. Программируемая текстура может быть определена специальной подпрограммой.

Как показано на рис. 4.71, наложение текстуры может быть получено за два шага. Сначала на поверхность отображается четыре угла пиксела. Для бикубического куска поверхности это отображение, естественно, определяет набор точек на координатном пространстве поверхности $\{u, v\}$. Затем точки угла пиксела в координатном пространстве этой поверхности отображаются в координатное пространство текстуры (s, t) . Четыре (s, t) точки в карте текстуры определяют четырехугольник, приближающий более сложную форму, которую пиксел может фактически отобразить из-за искривления поверхности. Значение для пиксела вычисляют, учитывая все элементы текстуры, расположенные в пределах четырехугольника. При этом необходимо учитывать каждую долю элемента текстуры, лежащего в пределах четырехугольника.



Рис. 4.71. Отображение текстуры от пиксела до поверхности и до карты текстуры

Моделирование глобального освещения с трассировкой лучей. Модель освещения с трассировкой лучей (рейтресингом) предназначена для расчета интенсивности отраженного к наблюдателю света в каждой точке (пикселе) изображения. Она может быть локальной или глобальной. В первом случае учитывается только свет, падающий от источника света и ориентация поверхности, а во втором случае учитывается также свет, отраженный от других объектов сцены или пропущенный сквозь них. Глобальная модель освещения воспроизводит важные эффекты; некоторые из них показаны на рис. 4.72.

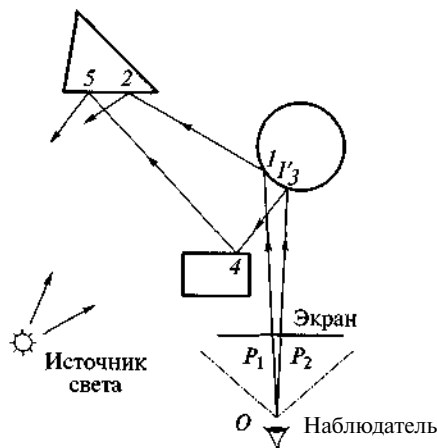


Рис. 4.72. Глобальная модель освещения:

7-5 — видимые точки объектов сцены;
 P_1, P_2 — лучи

Сфера, треугольная призма и параллелепипед, изображенные на рис. 4.72, непрозрачны, их поверхность зеркальна. Наблюдатель смотрит из точки O на точку 1 на сфере. При этом он видит не только сферу, но и точку 2 на призме. Призма, загороженная от наблюдателя параллелепипедом, становится видимой вследствие отражения света на сфере. Точка 5 на призме видима благодаря двум отражениям. Она отражается от обратной стороны параллелепипеда в точке 4 к точке 3 на сфере, а затем — к наблюдателю. Кроме того, наблюдатель видит точку 5 в точке $1'$ после одного отражения от сферы, поэтому на сфере находятся два изображения призмы. Изображение, расположенное вокруг точки 1 , перевернуто, так как получается после одного отражения; а вокруг точки 3 не перевернуто, так как получается после двух отражений. Интенсивность второго изображения меньше. Наконец, в сфере отражается обратная сторона параллелепипеда, т. е. наблюдатель видит ее, хотя прямой свет источника на нее не падает. Эта сторона освещается рассеянным светом и светом, отраженным от других объектов сцены.

Отсюда следует, что обычная операция отбрасывания задних граней, применяемая при удалении невидимых поверхностей, здесь не годится, как и предварительная сортировка по глубине. Поэтому из всех алгоритмов удаления невидимых граней, рассмотренных выше, остается только метод трассировки лучей. Таким образом, глобальная модель освещения является частью алгоритмов выделения видимых поверхностей путем трассировки лучей.

Выше рассмотрен алгоритм построения видимых непрозрачных поверхностей методом трассировки лучей, в котором луч трассируется до первого пересечения с объектом. В глобальной модели освещения трассировка луча на этом не кончается. Здесь предполагается, что падающий луч отражается от поверхности и пропускается сквозь эту поверхность по законам геометрической оптики. Это значит, что в точке пересечения луча с объектом образуются еще два луча,

для которых нужно найти все пересечения с объектами сцены. Этот процесс повторяется, пока не останется ни одного пересечения. Процесс трассировки заканчивается, когда луч покидает сцену. При пересечении луча с поверхностью направления отраженного и пропущенного лучей рассчитываются по законам геометрической оптики. В частности, отраженный луч и падающий луч лежат в одной плоскости, и угол падения равен углу отражения. Для того чтобы определить интенсивность в каждом пересечении луча с поверхностью, надо пройти трассу луча в обратном направлении. Интенсивность в узлах рассчитывается в соответствии с моделью освещения, причем для каждого следующего узла она ослабляется пропорционально расстоянию между точками пересечения. После прохода всей трассы луча определяется окончательная интенсивность пиксела.

Теоретически трассы лучей могут быть бесконечно разветвленными. Процесс построения трассы лучей заканчивается, когда все лучи уходят за пределы сцены, но его можно также прерывать, когда интенсивность света падает ниже определенного предела.

Моделирование глобального освещения с радиосити. Радиосити (Radiosity), согласно определению в литературе по физике, является общей падающей энергией на единицу площади поверхности. В контексте визуализации «энергия» — это энергия света.

Процедура решения радиосити — сложный метод вычисления света, отраженного между рассеивающими свет поверхностями. Его можно использовать, когда надо показать эффекты отражения цвета (когда одна цветная поверхность содержит оттенок цвета другой близлежащей поверхности) и дисперсии света (отражение косвенного света на другие поверхности в сцене). Отметим, что рейтресинг распознает только зеркальное отражение лучей света в сцене.

Процедура решения радиосити в отличие от рейтресинга — не собственно метод закрашивания, так как она просто генерирует решение освещения, к которому, в свою очередь, может быть применено закрашивание. Фактически процедура решения радиосити и возможности рейтресинга нужно применять совместно, чтобы воспроизвести растровое изображение реалистичного закрашивания, используя возможности обоих методов. Процедура решения радиосити используется как препроцессор для закрашивания, при этом вычисляется глобальное, видонезависимое (рассеянное) решение освещения. В методе трассировки лучей используется решение радиосити для представления видозависимого растрового изображения путем добавления зеркальных глянцевых отсветов и отражений. Только рейтресинг, который включает решение радиосити, создает естественную сцену, освещенную как направленным, так и отраженным светом с едва различимыми тенями, обычно присутствующими в реальном мире.

Поскольку решение радиосити является видонезависимым, то его можно многократно использовать для представления дополнительных растровых изображений объекта из различных видов. В процессе решения радиосити за достаточно короткий промежуток времени вычислений получаются полезные промежуточные результаты. Затем результаты автоматически улучшаются до

конечного решения. Решение радиосити позволяет отображать промежуточные результаты так, чтобы наблюдатель мог определить, когда решение будет удовлетворительным, и остановить вычисления. Кроме того, можно использовать в качестве критерия остановки вычислений или некоторое фиксированное число «передач энергии отраженного света», или некоторую долю общего глобального освещения.

4.2.4. Алгоритмы анимации трехмерных моделей

Методы КГ широко используются для анимации в различных сферах, включая индустрию развлечений (кино и мультфильмы), рекламу, научные и инженерные исследования, обучение и образование. Термин «компьютерная анимация» означает любую последовательность видимых изменений изображения. Помимо изменения положения объекта путем переноса или вращения компьютерная анимация может отображать изменение во времени размера объекта, его цвета, степени прозрачности и текстуры поверхности. В рекламе часто используется преобразование одного объекта в другой. Компьютерную анимацию можно получать также, меняя положение, ориентацию или фокусное расстояние камеры. Кроме того, для получения компьютерной анимации можно менять эффекты освещения или другие параметры, связанные с освещением и визуализацией.

Двумя базовыми методами построения последовательности изображений движения является *анимация реального времени* и *покадровая анимация*. В компьютерной анимации реального времени сцены последовательности изображаются по мере генерации. Таким образом, анимацию нужно генерировать со скоростью, совместимой с ограничениями, наложенными частотой обновления. При покадровой анимации каждый кадр сцены движения генерируется и записывается отдельно. Позже кадры можно записать в фильм или же последовательно отобразить на мониторе в режиме воспроизведения в реальном времени. Простые анимационные сцены обычно генерируются в реальном времени. Сложная анимация требует медленного покадрового построения. В то же время некоторым приложениям необходима анимация в реальном времени независимо от ее сложности. Например, анимация симулятора полетов выполняется в реальном времени, поскольку изображение на мониторе нужно сгенерировать немедленно после изменения настроек управляющих сигналов. В подобных случаях часто разрабатываются специальные аппаратные и программные системы, позволяющие быстро выводить на экран сложные последовательности.

В большинстве случаев простые анимационные последовательности можно создавать, используя методы реального времени. Однако в общем случае анимационная последовательность в системе с растровой разверткой создается по одному кадру за раз, так что каждый законченный кадр можно сохранить в файле для будущего просмотра. После этого для просмотра анимации последовательно

воспроизводится полная последовательность кадров или же кадры переводятся в фильм. Если требуется сгенерировать анимацию в реальном времени, кадры картины необходимо создавать достаточно быстро, чтобы непрерывно отображать последовательность движения. Для сложной сцены построение одного кадра анимации может занимать большую часть времени цикла обновления. В этом случае объекты, сгенерированные вначале, будут отображаться в течение почти всего времени цикла обновления, но объекты, сгенерированные к концу цикла обновления, исчезнут почти сразу же после того, как были отображены. Кроме того, для очень сложных сцен время построения кадра может быть больше времени обновления экрана, что приведет к неравномерному движению и ломаным изображениям кадров.

Можно анимировать объекты вдоль двухмерных траекторий движения, применяя преобразования с использованием таблицы цветов. В этом случае объект предопределяется в последовательных точках вдоль траектории движения, и в позиции таблицы цветов устанавливаются последовательные блоки значений пикселей. Пикселям в первом положении объекта задается цвет изображения, а пикселям в других положениях объекта присваивается цвет фона. Анимация заключается в изменении значений таблицы цветов так, чтобы цвет объекта в последовательных положениях вдоль траектории анимации становился цветом изображения, а предыдущее положение перекрашивалось в цвет фона.

Разработка и проектирование анимационных сюжетов. Производство традиционной анимации заслуживает некоторого обсуждения. Полезно иметь представление о том, как фрагмент анимации разбивается на части и как аниматоры подходят к производству законченного проекта. Большинство методов из традиционной анимации применимо и к компьютерной анимации.

Фрагмент анимации обычно рассматривается с использованием четырехуровневой иерархии. Вся анимация называется продукцией. Как правило, продукции разбиваются на главные части, называемые последовательностями. Последовательность — главный эпизод — обычно определяется областью сцены. Продукция может содержать до десятка последовательностей. Последовательность разбивается на один или несколько дублей. Дубль — непрерывный фрагмент, снятый камерой. Дубль разбивается на отдельные кадры фильма. Кадр — одно записанное изображение. В результате имеем иерархию, показанную на рис. 4.73.

Чтобы успешно спланировать и осуществить производство части анимации, требуется несколько шагов. Анимация является процессом проб и ошибок, кото-

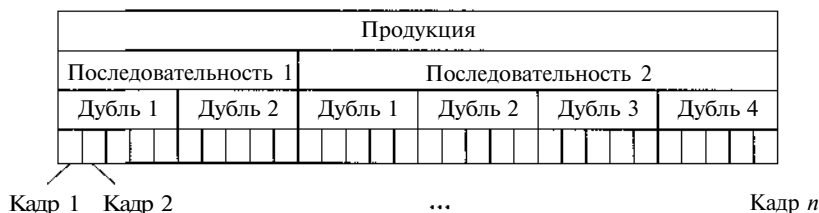


Рис. 4.73. Пример иерархии в традиционной анимации

рый включает в себя обратную связь от одного шага к предыдущим шагам и обычно требует нескольких итераций путем повтора многих предыдущих шагов в различные моменты времени разработки анимации. Несмотря на это, производство анимации обычно следует стандартному шаблону. Сначала определяется предварительная история, включая сценарий. Пишется раскадровка, определяющая действие посредством набросков отдельных кадров. Кадры часто сопровождаются текстом, который обрисовывает происходящее действие. Это используется для представления, обзора и рецензирования действия, а также для просмотра развития персонажа. Разрабатывается модельный лист, состоящий из набора рисунков для каждого персонажа в различных положениях и использующийся для того, чтобы убедиться, что все появления персонажа согласованы между собой, по мере того как этот персонаж рисуют во время анимационного процесса. Демонстрационный лист содержит информацию о каждом кадре: звуковое сопровождение, движение камеры и составляющие элементы. В направляющем листе собирается статистика по каждой сцене. Когда раскадровка определена создается детализированный сценарий, который более подробно определяет действие. Ключевые кадры определяются и делаются ведущими аниматорами для помощи в определении и развитии персонажа и качества изображения. Аниматоры отвечают за производство остальных кадров между ключевыми кадрами. Пробные съемки, короткие последовательности, снятые в полном цвете, используются для проверки качества изображения. Для того чтобы полностью проверить движение, может быть снят карандашный тест, который является полной отрисовкой движения (здесь используются низкокачественные изображения, например наброски карандашом). Проблемы, обнаруженные в пробных съемках и карандашных тестах, могут потребовать переработки ключевых кадров, детализированного сценария или даже раскадровки. Рисование — перенос карандашных рисунков на пленки. Ретуширование — процесс переноса цвета на пленки.

Вследствие высокого уровня точного отслеживания времени по сравнению с живой съемкой в анимации очень важен звук. В анимации звук бывает трех видов: музыка, специальные эффекты и голос. Что будет создано первым — звук или анимация — зависит от типа анимации и роли, которую играет звук. Для анимации с синхронизированным движением губ сначала создается звуковой трек и на его основе создается анимация. В большинстве других случаев сначала создается анимация, а потом на ее основе создается звук. При этом приблизительно звуковое сопровождение строится в то же самое время, что и раскадровка.

Разработка компьютерной анимации основана на большинстве идей из процесса производства традиционной анимации, включая использование раскадровки, тестовых съемок и карандашных тестов. Раскадровка непосредственно переносится на процесс компьютерной анимации, хотя она может быть выполнена непосредственно в процессе производства анимации. Раскадровка занимает ключевое место в процессе анимации и является важным компонентом в планировании анимации. Использование ключевых и промежуточных кадров также используется в некоторых компьютерных анимационных системах.

В то время как компьютерная анимация использует определенные подходы к производству из традиционной анимации, существуют значительные отличия, между тем как в компьютерной и традиционной анимациях создаются отдельные кадры анимации. В компьютерной анимации обычно существует четкое различие между: созданием моделей; расположением моделей, включая расположение камеры и освещение; заданием движения моделей, источников света и камер и процессом визуализации, примененным к этим моделям. Это позволяет повторно использовать модели и параметры освещения. В традиционной анимации все эти процессы происходят одновременно, по мере создания очередного рисунка, единственным исключением является возможное повторное использование фона, например в многослойном подходе.

Два основных инструмента традиционной анимации — тестовые съемки и карандашные тесты — имеют аналоги в компьютерной анимации. Баланс между скоростью и качеством может быть изменен на каждом из трех этапов создания кадра в компьютерной анимации: построение модели, управление движением и визуализация.

Компьютерная анимация обычно использует следующий план:

- создание раскадровки (storyboard) — разработка схемы действия. Она определяет последовательность движения как набор базовых событий, которые должны произойти. В зависимости от типа анимации раскадровка может состоять из набросков (эскизов) вместе с кратким описанием движения или быть просто списком основных идей, которые будут реализованы в действии. Изначально набор эскизов движения прикреплялся на большую доску, которая представляла общий вид анимационного проекта, откуда и пошло название (от англ. story — история, сюжет, board — доска);
- для каждого участника действия дается определение объекта. Объекты могут определяться как многогранники или тела с гладкими (сплайнными) поверхностями. Кроме того, часто дается описание движения, которое должен совершить каждый персонаж или объект сюжета;
- создание ключевых кадров (key frame) — подробное изображение сцены в определенный момент анимационной последовательности. В каждом ключевом кадре все объекты (или символы) размещаются согласно времени этого кадра. Одни ключевые кадры выбираются в крайних положениях действия; другие размещаются так, чтобы интервал между ключевыми кадрами не был слишком большим. Для сложных движений задается больше ключевых кадров, чем для простых, медленно меняющихся. За разработку ключевых кадров обычно отвечают старшие аниматоры и часто каждый персонаж анимационного фильма курирует отдельный аниматор.

Промежуточными называются кадры, расположенные между ключевыми. Общее число кадров, а следовательно, общее число промежуточных кадров, необходимых для создания анимации, определяется средой, в которой планируется отображать анимацию. Фильм требует 24 кадра в секунду, а графические терминалы обновляются с частотой не менее 60 или более кадров в секунду. Обычно

временные рамки движения задаются так, чтобы между каждой парой ключевых кадров располагалось 3-5 промежуточных кадров. В зависимости от скорости движения некоторые ключевые кадры могут дублироваться. В качестве примера укажем, что минутный фильм без дублирования требует 1440 кадров. Если между каждой парой вводится пять промежуточных кадров, потребуется разработать 288 ключевых кадров.

В зависимости от приложения может потребоваться решить еще несколько задач. К ним относятся проверка и редактирование движения, а также создание и синхронизация звуковой дорожки. Такие задачи, необходимые для создания полной анимации, также решаются на компьютере.

Базовые функции компьютерной анимации. Многие программные системы анимации разрабатывались либо для анимации в целом, либо для выполнения специализированных задач. Базовые функции анимации включают управление движением объекта, генерацию проекций объекта, создание движения камеры и генерацию промежуточных кадров. Одни программы анимации, например Wavefront, предлагают специальные функции как для проектирования анимации в целом, так и для обработки отдельных объектов. Другие являются специализированными программами, предназначенными для выполнения отдельных задач, например генерации промежуточных кадров или анимации рисунков.

Для разработки анимации в целом часто предлагается набор процедур для записи базы данных объекта и управления ею. Формы объектов и сопутствующие параметры записываются в базу данных и в ней же обновляются. К функциям работы с объектами относятся функции для генерации движения объекта и визуализации его поверхностей. Движение может генерироваться, согласно заданным условиям, с использованием двух- или трехмерных преобразований. Затем для определения видимых поверхностей можно применять стандартные функции и алгоритмы визуализации.

Дополнительные функции имитируют движение камеры — наезд, панормирование и наклон. Они также могут автоматически генерировать промежуточные кадры.

Системы ключевых кадров. Несколько заданных ключевых кадров с помощью системы ключевых кадров позволяют сгенерировать набор промежуточных кадров. Пути движения задаются априорно в виде кинематического описания (как набор сплайновых кривых) или их можно рассчитывать, основываясь на заданных силах, действующих на анимируемые объекты.

Для сложных сцен кадры разбиваются на отдельные компоненты или объекты, называемые келями (eel, celluloid transparency — диапозитив кинофильма). Данный термин заимствован из мультипликационных технологий, где фон и каждый персонаж на сцене размещают на различных диапозитивах. Диапозитивы выстраивают в порядке от фона до переднего плана, фотографируют и получают полный кадр. Затем, используя заданные пути анимации, получают следующий кель для каждого персонажа, положения которых интерполировались по информации с ключевых кадров.

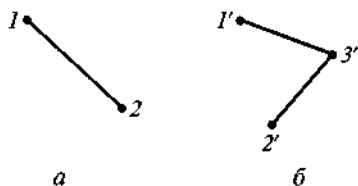


Рис. 4.74. Ключевые кадры k (а) и $k+1$ (б):
1, 2, $1'$, $2'$ — вершины

При преобразовании сложных объектов их формы могут меняться со временем. Примеры — одежда, детали лица, увеличенные детали, изменяющиеся формы и взрывающиеся или распадающиеся объекты. Для поверхностей, описанных многоугольной сеткой, данные изменения могут привести к значительным изменениям формы многоугольников, так что число углов может быть разным для соседних кадров. Эти преобразования включаются в разработку промежуточных кадров, для чего, согласно требованиям заданных ключевых кадров, добавляются или удаляются стороны многоугольников.

Преобразование форм объектов называется трансформацией (morphing, сокращенно от metamorphosis — метаморфоза). Для моделирования трансформации в соседних ключевых кадрах изменяют формы многоугольников, а в промежуточных кадрах это выглядит как переход одной формы в другую.

Для двух данных ключевых кадров с разным числом отрезков, задающих преобразование объекта, вначале можно так выровнять спецификацию объекта в одном кадре, чтобы число сторон многоугольника (или число его вершин) было одинаковым для двух кадров. Данный процесс предварительной обработки иллюстрируется рис. 4.74.

Отрезок в ключевом кадре k преобразуется в два отрезка в ключевом кадре $k+1$. Поскольку ключевой кадр $k+1$ имеет лишнюю вершину, в ключевом кадре k вводится дополнительная вершина между вершинами 1 и 2, чтобы уравнять число вершин (и сторон) в двух ключевых кадрах. Используя линейную интерполяцию для генерации промежуточных кадров, дополнительную вершину в ключевом кадре k переводят в вершину $3'$ по прямолинейной траектории, изображенной на рис. 4.75.

На рис. 4.76 приведен пример треугольника, линейно расширяющегося до четырехугольника.

Языки компьютерной анимации. При необходимости можно разработать процедуры создания и управления анимационными последовательностями на

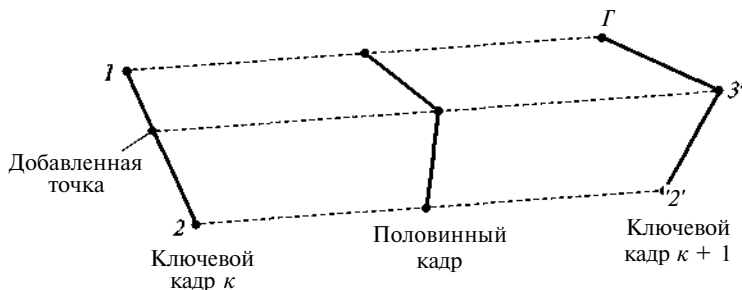


Рис. 4.75. Половинный кадр:
1, 2, $1'$, $2'$, $3'$ — вершины

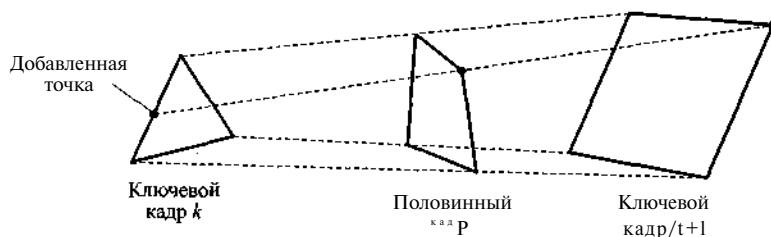


Рис. 4.76. Трансформация треугольника в четырехугольник

универсальном (C, C++, Lisp или Fortran) или специализированном языке программирования. Эти языки обычно включают графический редактор, генераторы ключевых и промежуточных кадров, стандартные графические процедуры. Графический редактор позволяет аниматору разрабатывать и модифицировать формы объектов, используя сплайновые поверхности, методы конструктивной твердотельной геометрии или другие схемы представления.

При спецификации анимации очень важно правильно описать сцену, т. е. разместить объекты и источники освещения, определить фотометрические параметры (интенсивности излучения источника и параметры освещенности поверхности) и установить параметры камеры (положение, ориентацию и характеристики линз), а также разработать другую стандартную функцию — спецификацию действия, включающую схему траекторий движения объектов и камеры. Кроме того, надо реализовать обычные графические процедуры: преобразование изображения путем изменения точки обзора и перспективное преобразование, геометрические преобразования, определяющие движение объекта как функцию ускорения или кинетических спецификаций траектории, определение видимых поверхностей и операции по визуализации поверхностей.

Системы ключевых кадров изначально представляли собой отдельный набор процедур анимаций, предназначенных для генерации промежуточных кадров по заданным пользователем ключевым кадрам. В настоящее время данные процедуры часто являются компонентом более общего пакета программ анимации. В простейшем случае каждый объект сцены определяется как набор объемных тел, соединенных в некоторых точках и имеющих ограниченное число степеней свободы.

Параметризованные системы позволяют, определяя объект, задавать характеристики его движения. Настраиваемые параметры контролируют такие характеристики объекта, как степени свободы, ограничения движения и допустимые изменения формы.

С помощью сценариев можно определить спецификации объекта и анимационные последовательности в форме сценариев, вводимых пользователем. По сценарию создается библиотека различных объектов и движений.

Анимационный язык — группа структурированных конструкций, которые можно использовать для задания информации, необходимой для получения

анимации. Язык бывает сценарным (script based) — инструкции записаны для последующего выполнения, или графическим — диаграммы типа блок-схем несут в себе отношения и процедуры.

Первые анимационные системы для получения последовательностей движений использовали языки программирования общего назначения. Однако при получении анимации возникали большие накладные расходы в задании графических примитивов, структур данных объектов, преобразований и при визуализации. Анимационный язык — любое текстовое описание получаемой анимации. Он может быть написан на специальном сценарном языке или алгоритмическом языке общего назначения упрощенно. Типичными возможностями являются встроенные операции ввода-вывода графических объектов, данных для представления и поддержки иерархической композиции объектов, параметров времени, функций интерполяции, функций для анимирования иерархий объектов, аффинных преобразований, параметров визуализации, параметров для задания атрибутов камер и пирамиды видимости и возможности для управления производством, просмотром и хранением одного или более кадров анимации. Программа, написанная на анимационном языке, часто называется сценарием (script).

Преимущества использования анимационного языка двояки. Во-первых, задание анимации, записанной в виде программы на данном языке, является жестко запрограммировано записью анимации, которая может быть использована для получения анимации в любой момент времени. Кроме того, использование языка позволяет итерационно улучшить анимацию, поскольку сценарий можно изменять и заново строить анимацию. Во-вторых, наличие программных конструкций позволяет применять алгоритмический подход к управлению движением. Анимационный язык, если он достаточно мощный, может интерполировать значения функций и вычислять сложные выражения, реализовывать правила поведения и т. п. Недостаток использования анимационного языка — он фактически является языком программирования, и, следовательно, пользователю (аниматору) нужно быть программистом. Аниматор должен быть обучен не только искусству, но и программированию. Разработаны языки с упрощенными программными конструкциями, но их возможности ограничены. По мере ознакомления пользователя с языком у него появляется желание поместить в анимационный язык все больше алгоритмических возможностей, присущих языкам общего назначения. В настоящее время разработаны анимационные системы, которые фактически являются пользовательским интерфейсом поверх сценарного языка. Пользователи могут применять систему через интерфейс, предоставленный системой, и писать свои собственные сценарии на внутреннем языке. Примером такой системы является язык MEL компании Alias/Wavefront. Язык MEL предоставляет переменные управляющие операторы, процедуры, выражения, доступ к атрибутам объектов и возможность настраивать пользовательский интерфейс. Стандартный пользовательский интерфейс помогает пользователям, не умеющим программировать или не знающим используемый язык, предостав-

ляя в то же время всю мощь сценарного языка тем пользователям, которые хотят его применять.

В помощь аниматорам, не имеющим подготовки в программировании, были разработаны простые анимационные языки с простым синтаксисом и более легкой для понимания семантикой (например, язык ANIMA II). В начале развития КГ практически все анимационные системы были основаны на подобных языках, поскольку было мало художников с технической подготовкой, которые могли программировать на полноценном языке программирования. Эти системы с графическими пользовательскими интерфейсами были доступны художникам, но имели ограниченные возможности. В этих простых анимационных языках типичные операторы относятся к именованным объектам и преобразованиям объектов. Эти языки обладали синтаксисом, легким для интерпретатора, компилятора и аниматора.

По мере того как аниматоры становились более опытными в написании анимационных сценариев на подобных языках, в них требовалось вносить большие возможности. Разработчики языков добавили поддержку конструкций для организации циклов, условных операторов, переменных, вызовов процедур, структур данных. Стало очевидным, что добавление поддержки графических объектов и операций к существующему языку, такому как C или LISP, более целесообразно, чем разработка полноценного анимационного языка с нуля.

Во многих языках можно связывать значения переменной с функцией времени, которая задается процедурно или разрабатывается интерактивно с использованием интерполяционных функций. Таким образом, когда сценарной или анимационной системе другого типа требуется получить значение переменной для своих вычислений, то она передает эту переменную времени артикуляционной функции, которая возвращает свое значение для этого момента времени. Термин «артикуляционная переменная» (articulation variable), часто сокращаемый до *avar*, распространен в различных системах для управления сочленениями объектов.

Графические представления, подобно использованным в коммерческой системе NOI, представляют анимацию как сетевую диаграмму (dataflow network). Для представления объектов, операций и отношений между ними используется ациклический граф. Узлы этого графа имеют входы и выходы, подключенные к другим узлам сетевой диаграммы. Данные переходят от узла к узлу по дугам, интерактивно задающимся пользователем. Узел представляет операцию, которую нужно выполнить над данными, переданными в этот узел, такими как описание объекта. Узел преобразования обработает переданное описание объекта и на выходе выдаст преобразованный объект. Входы в узел могут быть использованы для параметризации, преобразования или метода создания и могут задаваться интерактивно в соответствии с артикуляционной переменной или в соответствии с произвольной процедурой.

Анимационные языки, построенные на модели «актер» (actor), представляют собой объектно-ориентированный подход к анимации, в котором «актер» (ин-

капсулированная модель) является графическим объектом со своими данными и процедурами, включая геометрическое описание, атрибуты визуализации и управление движением.

Основная идея этого подхода к анимации заключается в том, что данные, связанные с графическим объектом, не только определяют его геометрию и визуальные характеристики, но также описывают его движение. Так, инкапсулированная модель автомобиля включает в себя информацию о том, как открываются двери, как опускаются окна, и, возможно, даже внутреннюю работу двигателя. Работа с моделью «актер» осуществляется при помощи посылки сообщений, анимация объекта — передачей запросов «актеру» на соответствующие движения. Текущее состояние «актера» может быть получено посредством посылки ему запроса на информацию и получения от него ответа.

Языки, построенные на модели «актер», предоставляют удобный способ передачи информации, зависящей от времени. Однако инкапсулированная природа «актера» с системой передачи сообщений может быть неэффективной, когда она используется имитационными системами, в которых все объекты потенциально могут влиять на все остальные.

4.3. Форматы графических и геометрических файлов

Знание файловых форматов графических и геометрических файлов для обмена соответствующей информацией, а также их возможностей являются ключевыми факторами в КГ. Каждый из используемых сегодня форматов прошел естественный отбор, доказал свою жизнеспособность и полезность. Все существующие форматы имеют свои характерные особенности и возможности, что делает их незаменимыми в работе.

По способу представления в компьютере все графические данные можно подразделить на два больших класса: растровые и векторные данные, которым соответствуют свои форматы файлов для обмена этими данными.

4.3.1. Форматы векторных данных

Векторы — математическое описание объектов относительно точки начала координат. Как показано выше, чтобы компьютер нарисовал прямую, необходимо задать координаты двух точек, которые связываются по кратчайшему расстоянию, для дуги задается радиус и две точки и т. д. Таким образом, векторное представление — это набор геометрических примитивов. Кроме того, большинство векторных форматов могут также содержать внедренные в файл растровые объекты или ссылку на соответствующий растровый файл (технология OPI). Векторные форматы часто называют форматами представления геометрических данных. В целом эти форматы можно подразделить на две основные группы:

- международные стандартные форматы ISO;
- промышленные стандартные форматы.

Международные стандартные форматы ISO. ISO — Международная организация по стандартизации, в которую входит более 80 стран. Системы национальных стандартов (ANSI, ГОСТ, DIN) во многом соответствуют системе стандартов ISO. В рамках ISO и Международной электротехнической комиссии (IEC) работает совместный комитет по КГ, который занимается подготовкой и принятием стандартов в этой области.

Первым стандартом по представлению графических геометрических данных был стандарт **GKS** (Graphical Kernel System) — стандарт двумерного графического интерфейса, который использует неиерархический, нередатируемый формат описания изображений. В этом стандарте изображения строятся путем непосредственного вызова функций графической системы. Каждому графическому примитиву соответствует своя функция. Несколько примитивов вывода могут быть объединены в сегмент, которому присваивается уникальный идентификатор. Сегменты размещаются в общей памяти для повторного использования. Изменять содержимое сегментов не допускается. Специальные функции позволяют осуществлять операции над сегментами: перенос, масштабирование, поворот. Геометрическая информация подвергается преобразованиям в системе мировых координат. Нормированные координаты (NDC) устройства поддерживают несколько классов логических устройств ввода-вывода. В качестве примера реализации этого стандарта можно привести библиотеку SunGKS. Дальнейшее развитие стандарт GKS получил в стандарте GKS-3D ISO IS 8805 (Graphical Kernel System for three dimensions). GKS-3D — стандарт трехмерного графического интерфейса, который использует все возможности стандарта GKS и содержит некоторые дополнительные функции для работы с трехмерной графикой. GKS-3D поддерживает трехмерные примитивы, трехмерный ввод, трехмерные сегменты и обеспечивает работу с невидимыми линиями и поверхностями. Этот стандарт обеспечивает преобразование трехмерных мировых координат в координаты двумерного устройства отображения. Цепочка преобразований связывает четыре системы координат: мировые координаты, нормированные координаты (NDC-3), нормированные координаты проекции (NPC) и координаты устройства.

Позднее появился стандарт **PHIGS** (ISO IS 9592 — Programmer's Hierarchical Interactive Graphic System) — стандарт трехмерного графического интерфейса для моделирования пространственной геометрии. PHIGS использует иерархический редактируемый формат описания изображений. Изображение определяется как иерархическая структура графических объектов. Каждый объект имеет свою локальную координатную систему, что позволяет изменять ориентацию и взаимное расположение объектов на экране. Последнее свойство необходимо для динамических эффектов и анимации. Графический объект состоит из элементов, которыми являются примитивы и атрибуты. В отличие от формата GKS-3D, механизм создания структур в PHIGS позволяет редактировать отображаемый объ-

ект в процессе выполнения прикладной программы. Все объекты в такой структуре имеют уникальные идентификаторы. При этом элементы объекта могут быть помечены метками. Помеченные элементы могут быть удалены или изменены в любой момент времени. Ориентация объекта является его атрибутом и может быть изменена для элемента. Координатная модель PHIGS включает пять уровней: координаты моделирования, мировые координаты, координаты просмотра, нормированные координаты проекции (NPC) и координаты устройства.

PHIGS+ — расширение стандарта PHIGS, которое содержит дополнительные функции для получения реалистичных изображений трехмерных объектов. Данные функции позволяют получать блики, тени и другие эффекты освещения предметов. На платформе Sun реализована библиотека SunPHIGS, которая включает в себя возможности стандартов PHIGS и PHIGS+. Курсы по КГ во многих зарубежных университетах при наличии соответствующего технического и программного обеспечения включают практические занятия по этим языкам.

Стандарт GKS предусматривает работу с метафайлом **GKSM** (GKS Metafile). Структура GKSM описана в приложении к стандарту GKS. Взаимодействие прикладной программы с метафайлом осуществляется при помощи функций GKS. В стандарте GKSM регистрируется последовательность примитивов, выводимых GKS в течение сеанса на монитор. При воспроизведении изображения из метафайла графическая система повторно выполняет всю последовательность функций вывода примитивов на экран. Аналогичные средства хранения информации есть в стандарте GKS-3D. Некоторые CAD-системы используют формат метафайла GKSM для передачи изображений на другие платформы. Во время работы с графической системой в стандарте PHIGS (PHIGS+) описание структуры изображения находится в памяти системы. Функции PHIGS позволяют сохранить структуру в виде архива для повторного использования во время следующего сеанса или в другой системе. Стандарт PHIGS несовместим со стандартом GKS, поскольку в них существуют различия в форме описания изображений.

CGI (ISOIS 9639 — Computer Graphics Interface) — стандарт двухмерного графического интерфейса, который был принят значительно позже GKS. Принципы описания изображений у GKS и CGI аналогичны. Стандарт CGI имеет более широкий набор графических примитивов по сравнению с GKS. В нем значительно упрощена координатная модель и отсутствуют сложные геометрические преобразования. Он имеет одноуровневую виртуальную систему. Данный стандарт часто рассматривают как промежуточный стандарт на аппаратно-независимый интерфейс между физическими устройствами отображения и графическими системами более высокого уровня (GKS, PHIGS и др.).

CGM (Computer Graphics Metafile) — стандарт на графический метафайл. Формат CGM — это формат метафайла для хранения и передачи статических изображений. Один метафайл может содержать описание нескольких рисунков. Рисунок хранится в виде совокупности примитивов, используемых для его построения.

Основным преимуществом стандартов ISO является независимость от операционной системы и физических устройств. Это позволяет осуществлять перенос программ без существенных переделок на уровне исходных текстов (при использовании стандартных версий языка программирования). Другое преимущество стандартов ISO обусловлено стандартизацией двух- и трехмерного графического ввода, включая ввод координат графического курсора.

Рассмотренным стандартам присущи и недостатки. Один из них связан с потенциальной неполной совместимостью различных реализаций одного стандарта. Так, например, стандарт GKS определяет набор всего из шести примитивов вывода, в то время как в реальных библиотеках их используется гораздо больше. Один из примитивов GKS является обобщенным примитивом вывода, который обеспечивает расширение набора графических элементов в конкретной реализации. На практике большинство примитивов графических библиотек (окружность, эллипс, дуга и др.) являются расширением стандартного набора. Аналогичные проблемы возникают и с другими стандартами ISO, имеющими обобщенный примитив. Однако главным недостатком этих стандартов является отсутствие достаточной поддержки в реальных системах. Графические интерфейсы ISO/ANSI получили очень ограниченное распространение на рабочих станциях и практически не имеют поддержки на персональных компьютерах. Исключение составляет лишь стандарт CGM, но он пригоден для обмена изображениями между программами, а не для организации диалога. Отчасти такое положение можно объяснить сложностью реализации и недостаточным быстродействием вследствие многоуровневых геометрических преобразований. Другим недостатком является некоторое отставание стандартов от новаций в машинной графике. Время рассмотрения стандарта в ISO составляет до 5 лет, что не позволяет учитывать новейших тенденций.

Важным этапом стандартизации, связанным с машиной графикой, была разработка стандартов описания продуктов и технологий, которые появились в середине 1980-х годов в форме проекта STEP (STandard for the Exchange of Product model data) — семейства стандартов для обеспечения универсального механизма обмена данными о продукции и технологии как между различными организациями, так и между разными этапами жизненного цикла продукции. Ядром STEP стал объектно-ориентированный язык информационного моделирования EXPRESS (ISO 10303, ч. 11). Не являясь языком программирования, действующая версия EXPRESS все же обеспечивает объектно-ориентированную идеологию для описания концептуальных моделей данных (множественное наследование данных и ограничений, выводимые атрибуты и др.). Важной особенностью, на которой основан EXPRESS, является модель «сущность-связь» (E-R). Графическая версия языка EXPRESS-G полностью вытеснила язык IDEF IX, который использовался на начальных этапах проекта STEP. В версии языка EXPRESS v2 уже предполагается полная объектная реализация с поддержкой моделирования процессов, событий, транзакций, а также единая формальная метамодель, гораздо более детализированная и семантически более строгая, чем части Generic Resources серии стандартов ISO 10303 (ч. 41—49).

Вся работа над проектом STEP велась под эгидой подкомитета 4, технического комитета 184 ISO (ISO TC184/SC4), к концу 1990-х годов в рамках которого появилось еще несколько разной степени завершенности серий стандартов, связанных с описанием уже не только продукции и технологии (ISO 13584, ISO 14959, ISO 15926), но и управления производством (Manufacturing Management — MANDATE — ISO 15531) и использующих в качестве своей основы язык EXPRESS.

За прошедшие годы на основе EXPRESS и STEP сформировалась уже целая отрасль, которая обеспечивает значительное уменьшение трудозатрат при внедрении новых технологий и новых видов продукции. Причем если серия ISO 10303 задумывалась прежде всего для обслуживания автомобильной и аэрокосмической отраслей, то сейчас она охватывает уже большинство видов производств, включая электротехническое, кораблестроительное, строительное, нефтехимическое и другие производства. Появились не только компании, специализирующиеся на инструментарии технологии STEP, но и организации общеметодологического плана, связанные с развитием технологии данных о продукции (Product Data Technology — PDT), например EuroSTEP, PDT Solutions, PDTAG, PDES и др.

Несмотря на внешние успехи, идеология, методология и технология STEP/EXPRESS требуют глубокого совершенствования. С одной стороны, нужна гармонизация и модуляризация стандартов внутри самого ISO TC184/SC4, с другой стороны, оказалось необходимым выйти за рамки описания продукции и технологии и включить более широкий круг вопросов бизнеса, с третьей стороны — все более возникает необходимость в согласовании аналогичных работ с другими организациями, занимающимися разработками в том же направлении и прежде всего с группами CSMF (Conceptual Schema Modelling Facilities) и CDIF (CASE Data Interchange Format) в рамках объединенного технического комитета ISO и Международной электротехнической комиссии ISO/IEC JTC1, с Консорциумом WWW, с рабочими группами OMG, с группой KIF (Knowledge Interchange Format), ANSI ASC X3T2, а также с группой Open Application Group. Внутри самого сообщества ISO TC184/SC4 существует неудовлетворенность состоянием дел. Проект все более разрастается и становится все менее управляемым, не приводя к созданию более общего и семантически строгого ядра, которое бы использовалось во всех специализированных описаниях не только продукции и технологии, но и бизнеса в широком смысле, включая управление производством и данными о продукции. Для этого в рамках рабочей группы ISO TC184/SC4/WG10 началась работа по созданию общей метамодели в рамках проекта IIDEAS (Integration of Industrial Data for Exchange Access and Sharing). В проекте предполагается охватить не только стандарты ISO TC184/SC4, но и установить связь с другими упомянутыми организациями по стандартизации. Под их эгидой проводятся специальные семинары по интеграции моделей данных и процессов в информационных системах. В качестве базы предполагается использовать основанный на XML язык разметки для

материалов **MatML**, так как база должна быть полностью структурированной и Web-ориентированной.

В настоящее время в мире идет развитие множества тематически специализированных языков на основе языка XML. Появилось и уже стандартизовано консорциумом World Wide Web Consortium (W3C) ряд тематических подмножеств языка XML. Так, для описания математических выражений и формул используется язык **MathML**. Для описания векторных графических данных для Web разработан ряд форматов:

- язык **VML** (Vector Markup Language), разработанный на основе языка XML;
- формат **VRML** (Virtual Reality Modeling Language) (ISO/IEC 14772), поддерживаемый организацией Web3D Consortium;
- формат **SVG** (Scalable Vector Graphics standard), поддерживаемый организацией W3C (только 2D).

Промышленные стандартные форматы. Значительно чаще на практике поддерживаются промышленные стандарты. Точного определения промышленного стандарта не существует. В качестве такового обычно признают спецификации графической системы, получившей распространение на разных платформах и являющиеся стандартом де-факто.

Особо следует выделить формат **STL**, или формат стереолитографии, который используется в автоматизированном производстве для представления трехмерных моделей объектов на стадии быстрого прототипирования. Этот формат является стандартным входным форматом для большинства систем быстрого прототипирования. Информация об объекте включает список треугольных граней, которые описывают поверхность его твердотельной модели с заданной точностью, и представляется в виде ASCII, или двоичного файла. Файл ASCII .stl должен начинаться ключевым словом solid и заканчиваться endsolid. Внутри этих ключевых слов приводится описание треугольников. Описание каждого треугольника включает описание единичного вектора нормали, направленного от его поверхности, за которым следует список трехмерных координат всех вершин. Все координаты представлены в декартовой системе координат и записаны в виде чисел с плавающей точкой.

Формат **WMF** (Windows Metafile) является графическим форматом. Он служит для передачи векторной графики через буфер обмена (Clipboard) и распознается практически всеми программами Windows, связанными с векторной графикой. Однако, несмотря на кажущуюся простоту и универсальность, использовать формат WMF нужно только для передачи векторов. Формат WMF искажает цвет, не сохраняет ряд параметров, которые могут быть присвоены объектам в различных векторных редакторах, не содержит растровые объекты.

Формат **DGN** (DesiGN file format) разработан фирмой Bentley на основе формата ISFF (Intergraph Standard File Formats) фирмы Intergraph для представления векторных данных в двоичном упакованном виде. DGN — это внутренний

формат представления данных в программе автоматизированного проектирования MicroStation и других продуктах фирмы Bentley.

Формат **DWG** (DraWinG Format) разработан фирмой Autodesk для представления векторных графических и геометрических данных в двоичном упакованном виде; это внутренний формат представления данных в программе создания чертежей для САПР — AutoCAD и других продуктах фирмы Autodesk.

Формат **DXF** (Drawing Exchange Format) разработан фирмой Autodesk для представления векторных графических и геометрических данных в текстовом виде с возможностью редактирования в любом текстовом редакторе.

Формат **DWF** (Drawing Web Format) разработан фирмой Autodesk для представления векторных данных в Web.

Поскольку последние три формата де-факто стали международными, рассмотрим их подробнее.

Формат **DWG** — формат файлов, создаваемых в программе AutoCAD. Фактически формат DWG стал стандартом для всех САПР в машиностроении, архитектуре и строительстве. В 1997 г. компания Autodesk утверждала, что в мире создано более 2 млрд DWG-файлов. В настоящее время используются следующие форматы: DWG 12, DWG 13, DWG 14, DWG 2000, DWG 2004 DWG 2005 и разрабатываются новые. В ответ на просьбы заказчиков компания Autodesk предлагает пользователям DWG бесплатные инструменты: DWG TrueView для просмотра файлов DWG и DWG TrueConvert для преобразования файлов DWG из старых версий в новые, и наоборот. Программы DWG TrueView и DWG TrueConvert доступны для бесплатной загрузки с сайта компании Autodesk.

Программы DWG True View и DWG TrueConvert являются последними достижениями Autodesk в области промышленных и технологических стандартов, которые помогают заказчикам максимально использовать свои конструкторские данные. Кроме того, недавно представлен инструмент разработки Autodesk RealDWG, помогающий сторонним разработчикам приложений создавать и продвигать на рынок продукты, способные считывать и записывать данные в формате DWG, не требуя установки программы AutoCAD.

Следует упомянуть об ассоциации программных разработчиков и пользователей Open Design Alliance для продвижения открытых форматов обмена данными автоматизированного проектирования.

Разрабатываемый ассоциацией формат OpenDWG основан на формате DWG. OpenDWG спроектирован для обеспечения максимальной совместимости с DWG. Программные библиотеки OpenDWG обновляются регулярно, предоставляя совместимость со всеми версиями DWG вплоть до 2005.

Разрабатываемый ассоциацией формат OpenDGN основан на формате DGN. Фирма Bentley сотрудничала с Open Design Alliance для предоставления пользователям и разработчикам программ высокоэффективных библиотек, поддерживающих чтение-запись родного формата DGN.

Ассоциация финансирует развитие программных библиотек, которые доступны всем ее членам. Таким образом, члены ассоциации могут сосредоточиться

на том, чтобы разрабатывать решения автоматизированного проектирования, а не на трудностях чтения и записи сложных файлов автоматизированного проектирования.

Формат **DXF** (Drawing Exchange Format) — формат обмена чертежами — разработан фирмой Autodesk для экспорта информации из программы AutoCAD в программные продукты третьих сторон. Файл DXF является ASCII-файлом. Формат DXF — открытый формат и полностью документирован фирмой Autodesk (вся необходимая информация есть на сайте фирмы).

В каждом чертеже выделяются три основных группы данных: блоки (block), словари (dictionary) и таблицы символов (symbol table). Блоки содержат наборы графически отображаемых объектов (entities), имеют имя, собственную систему координат, могут быть включены в другие блоки как графические объекты с собственными уникальными параметрами преобразования координат (поворотом, масштабированием и т. д.), а также могут существовать как отдельные чертежи (external references). Данный формат воспринимается всеми реализациями программы AutoCAD и большинством программ САПР, существует также возможность его преобразования как в двоичный формат DWG, так и наоборот.

DXF-файлы являются стандартными текстовыми файлами ASCII. Их можно просто перетранслировать в форматы другой системы проектирования или добавить к другим программам для решения специализированных задач. Процесс написания программы, которая реализует связь с программой AutoCAD посредством алгоритма обработки информации DXF, не является сложной задачей, хотя DXF-файл наполнен различного рода информацией, и при ее анализе вручную может показаться, что эта задача невыполнима. Однако DXF-файл построен таким образом, чтобы его обработка проводилась легко с помощью соответствующей написанной на любом языке программы, например на языке Basic. Формат преднамеренно построен таким образом, что ненужная информация без особого труда может быть опущена, а считывание необходимой информации осуществляется просто. Обрабатывать группы, информации можно в любом порядке, пропуская любую ненужную группу.

Написание программы, которая осуществляет построение DXF-файла, является более трудной задачей, поскольку требуется сохранить логическую последовательность в пределах конкретного файла, которая является обязательной для программы AutoCAD.

Формат DXF-файла представляет собой полное описание чертежа в текстовой форме кода ASCII, допускающей простую обработку с помощью других программ. В ряде случаев удобнее использовать гораздо более компактный файловый формат DXB (Drawing Exchange Binary) — двоичный обмен чертежами, который строится на основе DXF.

Формат **DWF** (Drawing Web Format) — это открытый формат файлов, разработанный Autodesk в целях организации эффективного обмена проектными данными между проектировщиками с использованием Web-технологий для их просмотра, печати и рецензирования. DWF-файлы имеют высокую степень сжатия,

поэтому по сравнению с файлами моделей их размер значительно меньше и обмен осуществляется гораздо быстрее. Сжатие сокращает издержки, связанные с пересылкой чертежей, созданных в САПР (а также с управлением внешними ссылками и зависимостями). DWF позволяет отображать определенные проектные данные и стили печати для просмотра заинтересованными лицами и выполнить публикацию многолистовых наборов чертежей из нескольких DWG-файлов в один DWF-файл. В DWF могут быть опубликованы 3D-модели практических для всех продуктов Autodesk.

DWF-файлы ни в коей мере не заменяют формат DWG-файлов. Редактирование и обновление проектных данных по-прежнему выполняется в DWG-файлах. Однако DWF-файлы позволяют конструкторам, инженерам, разработчикам и другим пользователям передавать техническую документацию и демонстрировать свои идеи всем заинтересованным лицам. Для просмотра и печати DWF-файлов применяется небольшая бесплатная программа Autodesk — DWF Viewer. Просмотр, печать, выполнение изменений, нанесение пометок и отслеживание изменений осуществляются в программе Autodesk DWF Composer.

Формат DWF необходим всем специалистам, использующим формат DWG. Начиная с разработки идеи и заканчивая воплощением этой идеи в жизнь и дальнейшей эксплуатацией, во все этапы жизненного цикла любого изделия вовлекается множество проектировщиков, инженеров, разработчиков, клиентов и подрядчиков. Членам коллективов необходимо иметь возможность просмотра, вывода на печать, рецензирования и комментирования моделей. Пользователям САПР необходимо быстрое, удобное и безопасное средство обмена электронными файлами проектов, поддерживающее целостность данных исходной модели.

DWF-файлы можно также создавать с помощью бесплатной программы Autodesk DWF Writer. DWF Writer позволяет безопасно и эффективно преобразовывать файлы чертежей и другие документы в DWF из многих графических приложений.

Распространение DWF-файлов не нарушает целостности оригинальных чертежей. DWF-файлы сходны с бумажными чертежами — по умолчанию они включают только ту информацию, которую проектировщик намерен предоставить. По желанию можно включать или отключать дополнительную атрибутивную информацию. DWF-файлы можно защищать паролем и шифровать.

По сравнению с чертежами, выполненными на бумаге, DWF-файлы обладают намного более высокой информативностью. В то же время они менее точны, чем исходные DWG-файлы, и несут в себе меньший объем проектных данных. По умолчанию публикация DWF-файлов выполняется с разрешением 400 dpi независимо от физического размера модели. Причиной потери точности является применение алгоритмов сжатия при публикации из DWG в DWF. Если точность по умолчанию не соответствует конкретной ситуации, ее повышают либо выбором меньшего формата листа, либо изменением разрешения. Следует иметь в виду, что размер DWF-файла увеличивается при повышении разрешающей способности процесса публикации.

Обмен оригинальными файлами моделей в форматах DWG может быть непрактичным по ряду причин. Одна из них — это защита интеллектуальной собственности. Любой, кто получит файл в собственном формате программ (AutoCAD, Autodesk Inventor и т. п.), может изменить данные в нем или несанкционированно использовать их в своей разработке. Кроме того, стоимость программ типа AutoCAD, Autodesk Inventor, плата за установку этих программ и обучение каждого сотрудника работе с такими сложными приложениями может быть очень высокой. Использование DWF-формата помогает избежать этих трудностей. DWF-файлы занимают меньший объем, их быстрее и безопаснее распространять, чем DWG-файлы. В то же время в них полностью представлена вся проектная графическая информация. Пользователи САПР могут выполнять публикацию чертежей, карт или других графических моделей в DWF-формат. Лица, просматривающие DWF-файлы, всегда четко видят предназначенную для них информацию. Кроме того, DWF-формат поддерживает публикацию многолистовых наборов чертежей и упрощает просмотр и печать таких наборов. Компания Autodesk предоставляет набор функций для чтения и записи файлов в DWF-формате. Например, функции из набора DWF 6 Toolkit позволяют работать с многолистовыми DWF-файлами.

4.3.2. Форматы растровых данных

Растровый формат представляет собой прямоугольную матрицу (bitmap), разделенную на пиксели. Различают два типа растровых файлов: предназначенные для вывода на экран и для печати.

Разрешение файлов таких форматов, как GIF, JPEG, BMP, зависит от видеосистемы компьютера. Растровые форматы, предназначенные исключительно для вывода на экран, имеют только экранное разрешение, т. е. один пиксел в файле соответствует одному экранному пикселу. На печать они выводятся с экранным разрешением.

Растровые файлы, предназначенные для допечатной подготовки изданий, имеют подобно большинству векторных форматов параметр Print Size — размер для вывода на печать. С ним связано понятие печатного разрешения, которое представляет собой соотношение количества пикселей на один квадратный дюйм страницы (ppi, pixels per inch или dpi — dots per inch). Печатное разрешение бывает от 130 (газеты) до 300 dpi (высококачественная печать).

Конкретные растровые форматы отличаются друг от друга способностью нести дополнительную информацию: различные цветовые модели, векторы, альфа-каналы или каналы платковых (вро^д-цветов, слои различных типов, анимацию, возможности сжатия и многое другое.

Формат **BMP** (Windows Device Independent Bitmap) — это формат Windows. Он поддерживается всеми графическими редакторами, работающими под управлением этой операционной системы. Применяется для хранения растровых изображений, предназначенных для использования в Windows и, по сути, боль-

ше ни на что не пригоден. Способен хранить как индексированный (до 256 цветов), так и RGB-цвет (16 700 000 оттенков). Формат BMP используется только в Windows.

Формат **GIF** (CompuServe Graphics Interchange Format) — не зависящий от аппаратного обеспечения способ представления растровых данных, разработанный в 1987 г. (GIF87a) фирмой CompuServe. В 1989 г. формат был модифицирован (GIF89a), в него были добавлены поддержка прозрачности и анимации. GIF использует LZW-компрессию, что позволяет хорошо сжимать файлы, в которых много однородных заливок (логотипы, надписи, схемы).

Метод сжатия LZW (Lempel—Ziv—Welch), разработанный израильянами Лемпелом и Зивом в 1978 г., позднее был доработан в США. Этот метод сжимает данные путем поиска одинаковых последовательностей (они называются фразы) во всем файле. Выявленные последовательности сохраняются в таблице, им присваиваются более короткие маркеры (ключи). Так, если в изображении имеются наборы из розового, оранжевого и зеленого пикселей, повторяющиеся 50 раз, LZW выявляет это, присваивает данному набору отдельное число (например, 7) и затем сохраняет эти данные 50 раз в виде числа 7. Метод сжатия LZW лучше действует на участках однородных, свободных от шума цветов, при сжатии произвольных графических данных, но процесс кодирования и распаковки происходит медленно.

Формат GIF позволяет записывать изображение через строчку (Interlaced), благодаря чему имея только часть файла, можно увидеть изображение целиком, но с меньшим разрешением. Это достигается за счет записи, а затем подгрузки, сначала 1, 5, 10 и т. д. строчек пикселей и растягивания данных между ними, вторым проходом следуют 2, 6, 11 строчки, разрешение изображения в интернетовском браузере увеличивается. Таким образом, задолго до окончания загрузки файла пользователь может понять, что внутри, и решить, стоит ли ждать, когда загрузится весь файл.

В формате GIF можно назначить один или более цветов прозрачными, они станут невидимыми в интернетовских браузерах и некоторых других программах. Прозрачность обеспечивается за счет дополнительного alpha-канала, сохраняемого вместе с файлом. Кроме того, файл GIF может содержать не одну, а несколько растровых картинок, которые браузеры могут подгружать одну за другой с указанной в файле частотой. Так достигается иллюзия движения (GIF-анимация).

Основное ограничение формата GIF состоит в том, что цветное изображение может быть записано только в режиме 256 цветов. В ряде случаев этого недостаточно.

Формат **JPEG** (Joint Photographic Experts Group) — один из самых распространенных растровых форматов. Строго говоря, JPEG называется не формат, а алгоритм сжатия, основанный не на поиске одинаковых элементов, как в LZW, а на разнице между пикселями. Кодирование данных происходит в несколько этапов. Сначала графические данные конвертируются в цветовое пространство

LAB, затем отбрасывается половина или три четверти информации о цвете (в зависимости от реализации алгоритма). Далее анализируются блоки 8×8 пикселей. Для каждого блока формируется набор чисел. Первые несколько чисел представляют цвет блока в целом, в то время как последующие числа отражают тонкие детали. Спектр деталей базируется на зрительном восприятии человека, поэтому крупные детали более заметны. На следующем этапе в зависимости от выбранного уровня качества отбрасывается определенная часть чисел, представляющих тонкие детали. На последнем этапе используется кодирование методом Хаффмана для более эффективного сжатия конечных данных. Восстановление данных происходит в обратном порядке.

Цветовое пространство LAB представляет цвет в трех каналах: один канал выделен для значений яркости (L — Lightnes) и два других для цветовой информации (A и B). Цветовые каналы соответствуют шкале, а не какому-нибудь одному цвету. Канал A представляет непрерывный спектр от зеленого до красного, канал B — от синего до желтого. Средние значения для каналов A и B соответствуют реальным оттенкам серого.

Метод сжатия Хаффмана (Huffman), разработанный в 1952 г., используется как составная часть в ряде других схем сжатия, таких как LZW, JPEG. В методе Хаффмана, чтобы определить частоту каждого символа, анализируется набор символов. Затем для наиболее часто встречающихся символов используется представление в виде минимально возможного количества битов. Например, буква «е» чаще всего встречается в английских текстах. Используя кодировку Хаффмана, можно представить букву «е» всего лишь двумя битами (1 и 0) вместо восьми битов, необходимых для представления буквы «е» в кодировке ASCII.

Таким образом, чем выше уровень компрессии, тем больше данных отбрасывается, тем ниже качество. Используя JPEG, можно получить файл в 500 раз меньше, чем BMP. Формат аппаратно независим, полностью поддерживается на PC и Macintosh, однако он относительно нов и не понимается старыми программами (до 1995 г.). JPEG не поддерживает индексированные палитры цветов. Первоначально в спецификациях формата не было и CMYK, фирма Adobe добавила поддержку цветоделения, однако поддержка CMYK JPEG во многих программах имеет проблемы. Лучшим решением является использование JPEG-сжатия в Photoshop EPS-файлах фирмы Adobe, которое описывается ниже.

Существуют подформаты JPEG. Например, JPEG Baseline Optimized разработан специально для Интернет, его поддерживают все основные браузеры.

Формат JPEG лучше сжимает растровые картинки фотографического качества, чем логотипы или схемы — в них больше полутоновых переходов, среди однотонных заливок же появляются нежелательные помехи. Лучше сжимаются и с меньшими потерями большие изображения для Web или изображения с высоким разрешением (200–300 и более dpi), чем с низким (72–150 dpi), поскольку в каждом квадрате 8×8 пикселей переходы получаются более мягкие за счет того, что их (квадратов) в таких файлах больше. Нежелательно сохранять с JPEG-сжатием изображения, в которых важны все особенности цветопередачи,

так как во время сжатия происходит отбрасывание части цветовой информации. В JPEG следует сохранять только конечный вариант работы, потому что каждое дополнительное сохранение приводит ко все новым потерям (отбрасыванию) цветовых данных.

Формат **PNG** (Portable Network Graphics) — относительно новый формат, призванный заменить устаревший формат GIF. Этот формат использует сжатие без потерь (Deflate), сходное с LZW. Сжатые индексированные PNG-файлы, как правило, меньше аналогичных GIF-файлов, RGB PNG-файлы меньше соответствующих файлов в формате TIFF, рассматриваемом далее.

Глубина цвета может быть любой до 48 бит. Используется двухмерная запись изображения через строчку (interlacing не только строк, но и столбцов), который так же, как и в формате GIF, увеличивает размер файла. В отличие от формата GIF, где прозрачность либо есть, либо нет, формат PNG поддерживает также полупрозрачные пиксели, т. е. в диапазоне прозрачности от 0 до 99 %, за счет альфа-канала с 256 градациями серого.

В файл формата PNG записывается информация о гамма-коррекции. Поскольку гамма-коррекция — это некое число, характеризующее зависимость яркости свечения экрана монитора от напряжения на электродах кинескопа, то это число, считанное из файла, позволяет ввести поправку яркости при отображении. Оно необходимо для того, чтобы изображение выглядело одинаково на разных графических аппаратных средствах. Таким образом, гамма-коррекция помогает реализации основной цели передачи изображений в Интернет — одинакового отображения информации независимо от аппаратуры пользователя.

Формат **TIFF** (Tagged Image File Format) — аппаратно независимый формат, на сегодняшний день является одним из самых распространенных и надежных, его поддерживают практически все программы, так или иначе связанные с графикой. TIFF является лучшим форматом для импорта растровой графики в векторные графические программы и издательские системы. Ему доступен весь диапазон цветовых моделей от монохромной до RGB, CMYK и дополнительных цветов Pantone. TIFF может сохранять обтравочные контуры, альфа-каналы, другие дополнительные данные. В формате TIFF может быть использовано LZW-сжатие.

4.3.3. Язык Adobe PostScript

Язык **PostScript** — средство описания страниц фирмы Adobe. Был создан в 1980-х годах для реализации принципа WYSIWYG (What You See is What You Get — что ты видишь, то ты и получаешь). Файлы этого формата фактически представляют из себя программу с командами для вывода на печать. Они имеют расширение .ps или, реже, .rpt и получаются с помощью функции Print to file графических программ при использовании драйвера PostScript-принтера. Такие файлы содержат в себе сам документ (только то, что располагалось на страницах), все связанные файлы (как растровые, так и векторные), использованные шрифты, а

также другую информацию: цветоделение, полутоновые растры, линиатуру растра и другие данные для устройства вывода на печать. Если файл закрыт правильно, то не имеет значения, на какой платформе он делался, какие были использованы шрифты и т. п.

Растровые данные, как правило, записываются в двоичной кодировке (Binary). Бинарный код занимает вдвое меньше места, чем ASCII. Это относится ко всем форматам, основанным на языке PostScript: EPS и PDF, которые описываются далее.

Формат **EPS** (Encapsulated PostScript) можно назвать самым надежным и универсальным способом хранения данных. Он использует упрощенную версию PostScript: не может содержать в одном файле более одной страницы, не сохраняет ряд установок для принтера. Как и в файлы печати PostScript, в EPS записывают конечный вариант работы, хотя такие программы, как Adobe Illustrator и Adobe Photoshop, могут использовать его как рабочий. EPS предназначен для передачи векторных и растровых данных в издательские системы и создается почти всеми программами, работающими с графикой. Использовать его имеет смысл только тогда, когда вывод осуществляется на PostScript-устройстве. EPS поддерживает все необходимые для качественной печати цветовые модели, среди них такая, как Duotone; этот формат содержит данные RGB, данные обратного контура, информацию о растрах, о внедренных шрифтах.

Вместе с файлом можно сохранить эскиз файла (image header, preview). Это копия низкого разрешения в формате PICT, TIFF, JPEG или WMF, которая сохраняется вместе с файлом EPS и позволяет увидеть общее содержимое файла. Такой эскиз выводится на печать на принтере, не поддерживающем PostScript.

Изначально EPS разрабатывался как векторный формат, позднее появилась его растровая разновидность — Photoshop EPS. Кроме типа эскиза (TIFF, PICT, JPEG) Photoshop EPS позволяет выбрать способ кодирования данных. Photoshop EPS позволяет сжимать растровые данные с помощью алгоритма JPEG. Adobe доработала этот способ сжатия. JPEG в исполнении Photoshop EPS поддерживает CMYK и сжимает лучше, чем JPEG, полностью соответствуя первоначальным спецификациям JPEG.

Формат EPS имеет много разновидностей, что зависит от программы-создателя. Самые надежные EPS-файлы создают программы производства фирмы Adobe Systems: Photoshop, Illustrator и др.

Формат **PDF** (Portable Document Format) предложен фирмой Adobe как независимый от платформы формат для создания электронной документации, презентаций, передачи оригинал-макетов и графики по сети Интернет.

PDF первоначально проектировался как компактный формат электронной документации. Поэтому все данные в нем могут сжиматься, причем к разного типа информации применяются разные, наиболее подходящие для них типы сжатия: JPEG, CCITT, ZIP.

Метод сжатия CCITT (International Telegraph and Telephone Committee) был разработан для факсимильной передачи и приема. Он является более узкой вер-

сией кодирования методом Хаффмана. CCITT Group 3 идентичен формату сообщений по факсу, CCITT Group 4 — это формат факсов, но без специальной управляющей информации.

Файл PDF может быть оптимизирован. Из него можно удалять повторяющиеся элементы, устанавливая постраничный порядок загрузки страниц через Интернет, с приоритетом передачи сначала для текста, потом графики и, наконец, шрифтов.

Поскольку в настоящее время наиболее актуальной становится передача данных через Интернет, то следует выделить формат DWF для передачи векторных графических данных, форматы GIF, JPEG, PNG для передачи растровых графических данных и формат PDF для передачи документации.

Вопросы для самоконтроля

1. Перечислите основные алгоритмы для обработки растровых изображений.
2. В чем заключается основная идея преобразования Хоха?
3. Представьте геометрическую иллюстрацию алгоритма Брезенхема.
4. Назовите основные типы ограничений при параметризации плоских контуров и приведите соответствующие геометрические иллюстрации.
5. В чем состоит основная идея алгоритмов, использующих г-буфер, и алгоритма постстрочного сканирования?
6. Дайте геометрическую иллюстрацию метода рейтресинга.
7. Какие эффекты фотореалистической визуализации трехмерных моделей нельзя получить по методу Фонга?
8. В любой доступной вам системе трехмерного геометрического моделирования (3DMax, Maya, Inventor, Solid Works, КОМПАС 3D, T-Flex 3D и т. п.) создайте сцену из плоскости, на которой разместите примитивные тела (параллелепипеды, сферы, конусы, цилиндры, эллипсоиды вращения и т. п.), задайте три разных источника освещения и разместите их так, чтобы в сцене присутствовали тени и взаимные отражения света от созданных объектов. Проверьте качество визуализации следующих эффектов: наложение разных материалов на поверхности тел, мягкие тени, прозрачность, зеркальные взаимные отражения света.
9. Что такое система ключевых кадров?
10. Перечислите основные форматы файлов для передачи растровых данных.

СПИСОКЛИТЕРАТУРЫ

1. Computer Graphics: principle and Practice Second Edition. J. Foley, A. van Dam, et al. Addison-Wesley, 1996.
2. www.ixbt.com.
3. www.spec.org.
4. www.top500.org.
5. *Абламейко С.В., Лагуновский Д.М.* Обработка изображений: технология, методы, применение. — Минск: Амалфея, 2000.
6. *Хери Я., Бейкер Х.* Компьютерная графика: Пер. с англ. — М: Вильямс, 2005.
7. *Гуревич М.М.* Цвет и его измерение. — М.-Л.: Мир, 1950.
8. *Ивэнс Р.М.* Введение в теорию цвета: Пер. с англ. — М.: Мир, 1964.
9. *Каган Б.М.* Электронные вычислительные машины и системы. — М.: Энергоатомиздат, 1991.
10. *Королев Л.Н.* Архитектура процессоров электронных вычислительных машин. — М.: Издательский отдел факультета Вычислительной Математики и Кибернетики МГУ им. М.В. Ломоносова, 2003.
11. *Линдли К.* Практическая обработка изображений на языке Си: Пер. с англ. — М.: Мир, 1996.
12. *Медведев А.* DX Current. Настоящее аппаратного ускорения графики. — www.ixbt.com.
13. *Медведев А.* Longhorn Microsoft улучшает графику на 55 %. — www.ixbt.com.
14. Методы компьютерной обработки изображений / Под ред. В.А. Сойфера. — М.: ФИЗМАТ ЛИТ, 2003.
15. *Мураховский В.И.* Железо ПК. Новые возможности. — СПб.: Питер, 2005.
16. *Никулин Е.А.* Компьютерная геометрия и алгоритмы машинной графики. — СПб: БХВ-Петербург, 2003.
17. *Поляков А.Ю.* Методы и алгоритмы компьютерной графики в примерах на Visual C++. — СПб.: БХВ-Петербург, 2002.
18. *Пэрент Р.* Компьютерная анимация: Пер. с англ. — М.: КУДИЦ-ОБРАЗ, 2004.
19. *Роджерс Д.* Алгоритмические основы машинной графики: Пер. с англ. — М.: Мир, 1989.
20. *Роджерс Д., Адаме Дж.* Математические основы машинной графики: Пер. с англ. — М.: Мир, 2001.
21. *Цветков В.Я.* Геоинформационные системы и технологии. — М.: Финансы и статистика, 1998.
22. *Шикин А.В., Боресков А.В.* Компьютерная графика. Полигональные модели. — М.: ДИАЛОГ-МИФИ, 2001.

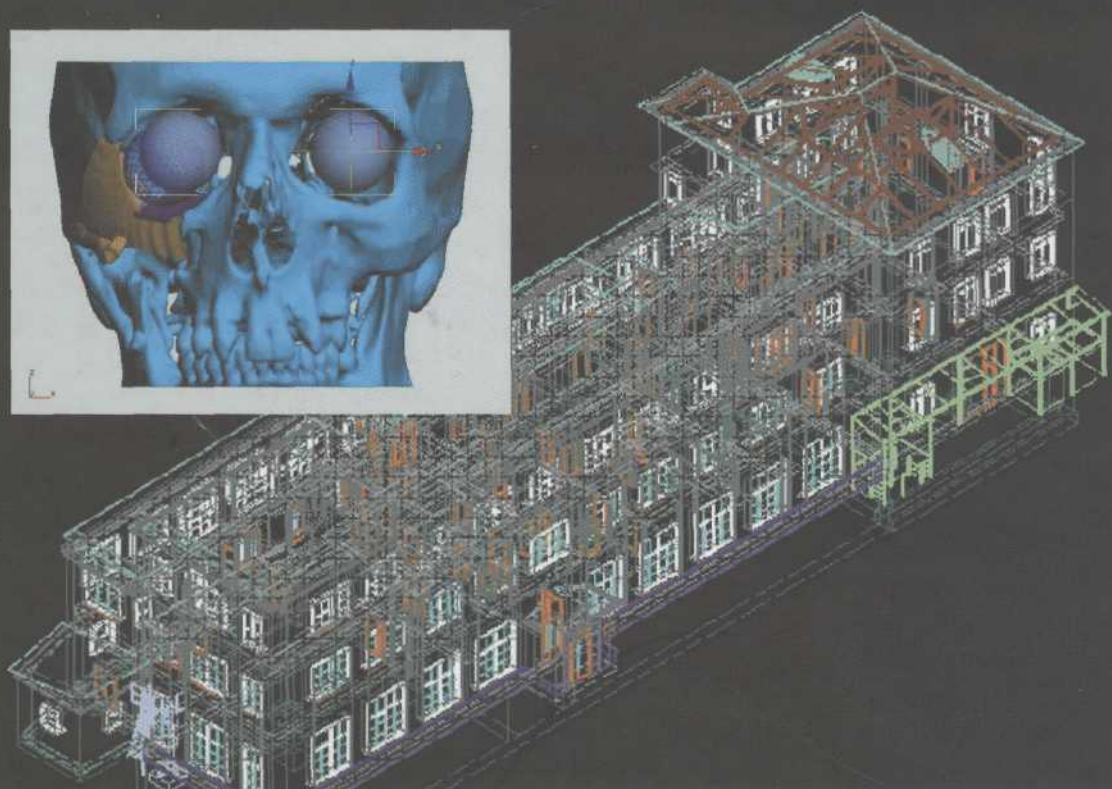
Информатика в техническом университете

А.Н. Божко, Д.М. Жук, В.Б. Маничев

Компьютерная графика



Издательство МГТУ
им. Н.Э. Баумана



ISBN 978-5-7038-3015-4



9 785703 830154

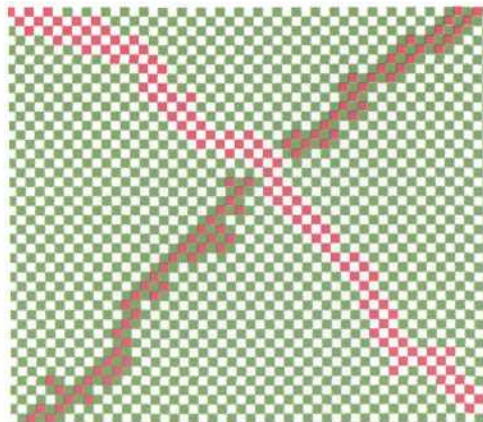


Рис. 1. Пример цветовой иллюзии



Рис. 2. Аддитивное (а) и субтрактивное (б) смешение цветов



Рис. 3. Формирование основных оттенков в субтрактивной модели

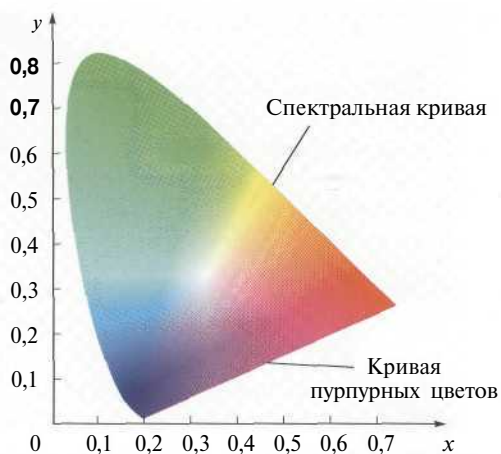


Рис. 4. Хроматическая диаграмма CIE

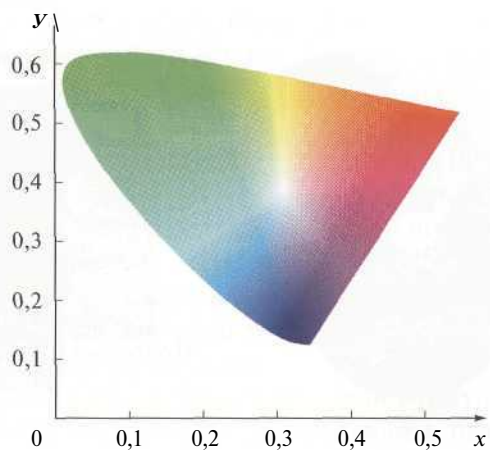


Рис. 5. Хроматическая диаграмма системы CIE Luv

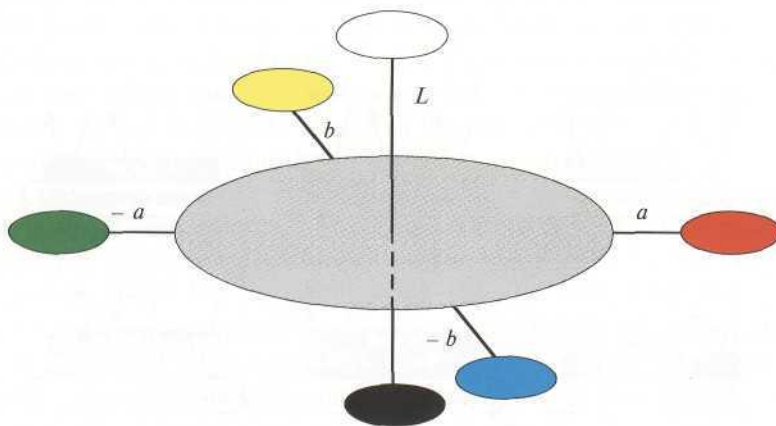


Рис. 6. Цветовая модель системы CIE Lab



Рис. 7. Представление системы HSB

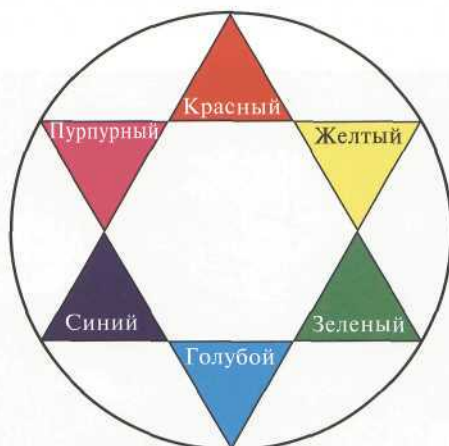


Рис. 8. Цветовой круг



Рис. 9. Изображение в натуральную величину и при пятикратном увеличении

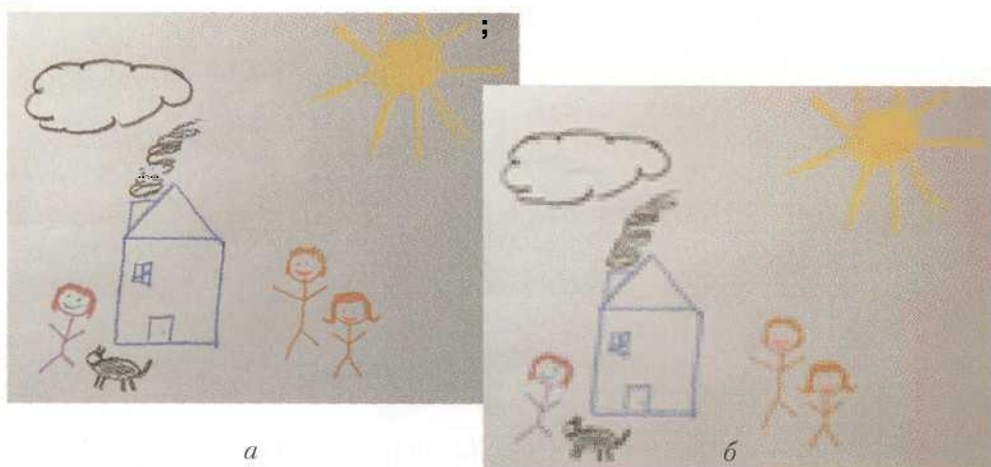


Рис. 10. Оригинал, оцифрованный с различным разрешением:
а — высокое разрешение; *б* — низкое



Рис. 11. Цветовой эталон

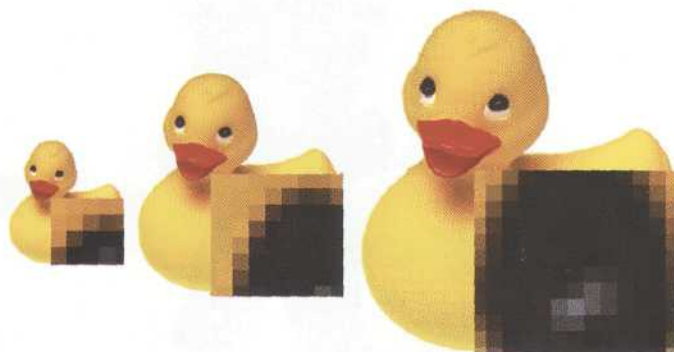


Рис. 12. Изменение экранных размеров одного оригинала при выборе различных значений разрешения

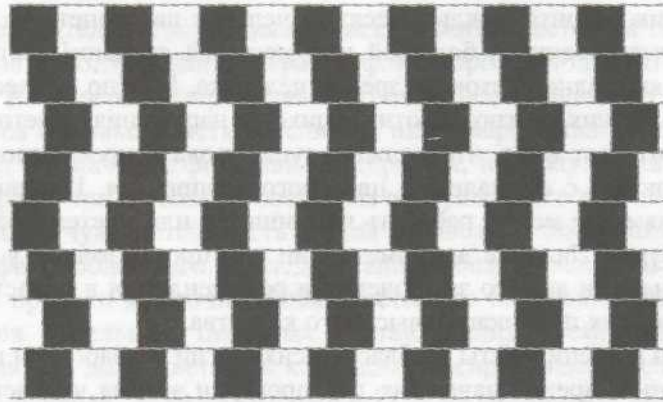


Рис. 1.5. Пример оптической иллюзии

На рис. 1.6 показана регулярная сетка, в узлах которой расположены круги небольшого размера. Это классический пример, демонстрирующий оптическую иллюзию в самой простой форме. Испытуемому предлагается подсчитать количество кружков разного цвета. Обычно уже после обработки первого ряда начинают путаться люди с самой устойчивой психикой и идеальным зрением.

На рис. 1 цветной вклейки приведен пример иллюзии цветового восприятия. Прямоугольное поле заполнено в шахматном порядке клетками светло-зеленого цвета, на диагоналях этой фигуры размещены прямоугольники отличного цвета. Требуется оценить сходство цветов клеточек, заполняющих диагонали. Кажется, что диагонали этой фигуры закрашены разным цветом, но инструментальная проверка или осмотр изображения при большом увеличении свидетельствуют о полной хроматической тождественности. Можно выиграть любое пари, что

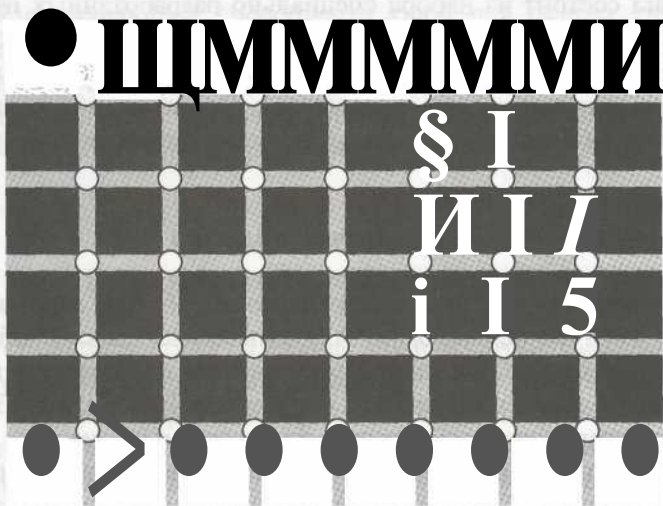


Рис. 1.6. Пример иллюзии восприятия тонов

дающие единый фундамент процедурам настройки и измерения цвета. Беспорядку со специализированными фирменными форматами был положен конец в 1995 г., когда фирма Apple объявила о создании встроенной в операционную среду системы управления цветом ColorSync 2. Фирма предложила новый стандарт записи профайлов и сделала его открытым. Формат оказался удачным и был стандартизован международным консорциумом по свету ICC (International Color Consortium). Разработка принята сообществом разработчиков программного и технического обеспечения и в настоящее время все системы управления цветом основываются на профилях ICC.

Профилирование и калибровка технических устройств оказываются неработоспособными без надлежащей системной организации. Программно-аппаратная среда, объединяющая средства управления цветом в КГ, называется системой управления цветом и ее часто обозначают аббревиатурой CMS (Color Management System). Существует несколько таких систем, среди которых можно выделить двух явных лидеров: на платформе Windows — Image Color Management (ICM); на платформе Macintosh — ColorSync.

Все системы CMS (рис. 1.11) включают в себя три основных составляющих:

- базовое цветовое пространство системы. Аппаратно-независимый способ описания цветов, свободный от ограничений и особенностей классов и типов технических устройств. Это своего рода общий знаменатель, к которому приводятся цветовые пространства отдельных технических устройств, входящих в технологическую цепочку подготовки цветных публикаций. В последних CMS эти функции выполняют CIE Lab или CIE XYZ. Базовое пространство — важная теоретическая составляющая любой системы управления цветом. Для рядового пользователя она не имеет прикладного значения, поскольку является полностью закрытой;



Рис. 1.11. Структура системы управления цветом

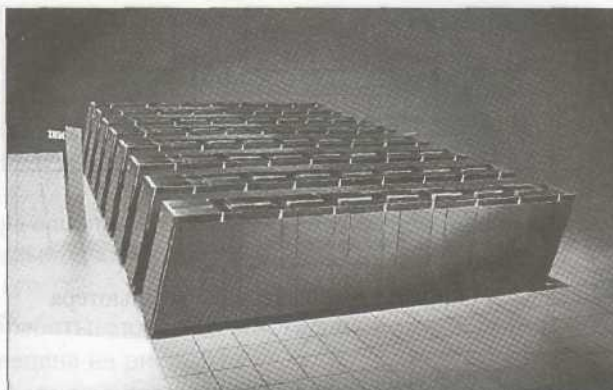


Рис. 2.5. Суперкомпьютер BlueGene/L фирмы IBM

числений, таких как моделирование атмосферных явлений, решение астрономических задач, прогнозирование погоды, разведка нефтяных и газовых месторождений и т. п. Как правило, суперкомпьютеры создаются под конкретные задачи или научные программы и изготавливаются в единичных экземплярах из серийных комплектующих. Суперкомпьютеры содержат сотни и тысячи процессоров, имеют большую оперативную память и очень высокое быстродействие. Они состоят из большого количества различных аппаратных средств, стоят десятки миллионов долларов, занимают большие помещения, а иногда и специально построенные здания (рис. 2.5).

Многие современные суперкомпьютеры созданы по кластерной технологии (Cluster). По этой технологии компьютер строится из нескольких десятков вычислительных машин, связанных между собой и функционирующих как единая система. Кластерные суперкомпьютеры легко масштабируются и позволяют получать высокое быстродействие и высокую готовность.

Быстродействие суперкомпьютеров обычно измеряется во ФЛОПСах (FLOPS — Floating Point Operations Per Second). ФЛОПС — количество арифметических операций с плавающей запятой, выполняемых в секунду. Производные единицы: 1 МегалоПС (МФЛОПС) = 1 млн арифметических операций в секунду; 1 ГигаФЛОПС (ГФЛОПС) = 1 млрд арифметических операций в секунду; 1 ТераФЛОПС (ТФЛОПС) = 1 трлн арифметических операций в секунду.

Организация TOP500 Supercomputer sites (www.top500.org) с 1993 г. дважды в год публикует статистику по 500 наиболее мощным суперкомпьютерам, определяя их производительность на тестовой программе High-Performance Linpack Benchmark (HPL) решения системы алгебраических уравнений. Характеристики пяти лучших компьютеров по данным на ноябрь 2006 г. приведены в табл. 2.1;

Мэйнфреймы — высокопроизводительные компьютеры с большими вычислительными ресурсами, способные решать сложные задачи, обрабатывать большие объемы данных и выполнять обработку нескольких тысяч запросов одновременно.

Таблица 2.1

Вычислительная система	Производитель	Заказчик	Число процессоров	Тип процессора	Максимальная производительность (TFLOPS)
BlueGene/L EServer Blue Gene	IBM	DOE/NNSA/LLNL	131072	Power PC 440	280,60
Red Storm	Cray Inc.	NNSA/Sandia National Laboratories	26544	Opteron 2,4 GHz dual core	101,4
BGW EServer Blue Gene/L	IBM	IBM Thomas J. Watson Research Center	40960	Power PC 440	91,29
ASCI Purple eServer pSeries" p575	IBM	DOE/NNSA/LLNL	12208	PSeries 575	75,76
MareNostrum BladeCenter JS21 Cluster	IBM	Barcelona Supercomputing Center	10240	PPC 970, 2,3 GHz	62,63

Конструктивно мэйнфреймы выполняются в едином корпусе в форме шкафа или тумбы (отсюда и их название), к которому могут подключаться многочисленные терминалы (рис. 2.6). Как правило, мэйнфреймы отличаются очень высокой надежностью.

Мэйнфреймы обычно используют для хранения и обработки больших баз данных, а также для создания крупных web-узлов с большим количеством клиентов.

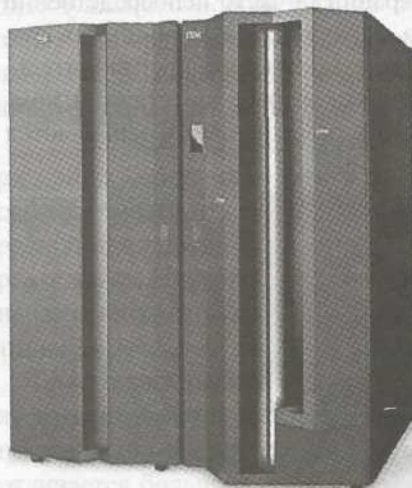


Рис 2.6. Мэйнфрейм IBM zSeries 990
2003 г.

Серверы — компьютеры, которые в вычислительных сетях являются центральными управляющими и информационными узлами. На серверах хранится большое количество информации, в том числе и *графической*, которую могут использовать все компьютеры, подключенные к сети, в зависимости от их статуса.

Сервер определяет работоспособность всей сети, сохранность баз данных и другой информации, поэтому серверы имеют систему хранения данных, отличающуюся большой емкостью и высокой надежностью, возможность замены неисправных блоков при непрерывной работе («горячая» замена)

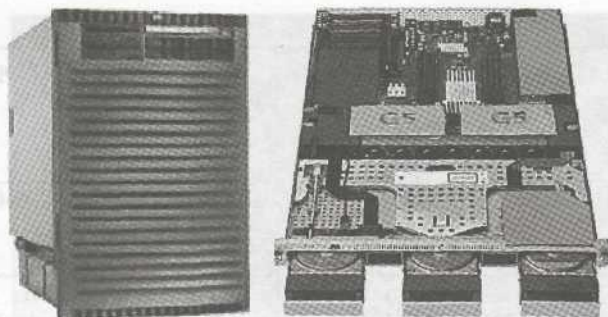


Рис. 2.7. Сервер Hewlett Packard Integrity rx8620-32 (2005 г.) и сервер Apple Xserve G5 (2005 г.)

и средства обеспечения длительной непрерывной работы в различных условиях. Серверы могут быть построены на базе различных семейств компьютеров и содержать от единиц до нескольких десятков процессоров. Конструктивное исполнение серверов многообразно — от настольного до шкафа (рис. 2.7).

Инженерные рабочие станции — относительно недорогие и достаточно мощные и надежные компьютеры на базе современных RISC-процессоров и ОС Unix, которые используются в качестве рабочих мест для профессиональной работы в различных областях, связанной со сложными и объемными вычислениями и с большими объемами данных. Обычно эти компьютеры используют для работы со сложной графической информацией и в CAD/CAE/CAM системах, поэтому их часто называют также графическими рабочими станциями (рис. 2.8).

Рабочие станции, как правило, специально конфигурируются для работы со сложной графической информацией, т. е. в их состав входит мощная графическая подсистема, реализующая многие графические операции (а часто непосредственно операторы языка OpenGL) аппаратно. Как правило, вычислительная мощность графических процессоров рабочих станций существенно превышает производительность центрального процессора. Кроме того, графические рабочие станции комплектуются мониторами с большим экраном и высоким разрешением и оснащаются



Рис. 2.8. Графическая рабочая станция

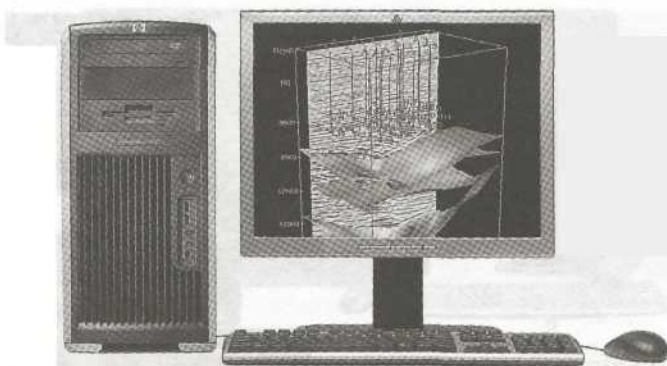


Рис. 2.9. Персональная рабочая станция

разнообразными устройствами для ввода графической информации и манипулирования изображениями — от простой мыши до больших графических планшетов или специальных шлемов для работы в режиме виртуальной реальности.

Персональные рабочие станции — графические рабочие станции, выполненные на вычислительной платформе, используемой в персональных ЭВМ, как правило, это платформа WLNTEL. Вычислительная платформа — совокупность центрального процессора (в данном случае — микропроцессор фирмы Intel) и ОС (в этом случае — вариант ОС Windows фирмы Microsoft). Обычно в качестве персональных рабочих станций используются высокопроизводительные персональные ЭВМ, укомплектованные дополнительными периферийными устройствами в зависимости от назначения станции (рис. 2.9).

Персональные компьютеры (ПК) — компьютеры, предназначенные для индивидуального использования одним пользователем автономно или в сети совместно с другими компьютерами. Персональные компьютеры бывают настольные, переносные и карманные. С точки зрения аппаратной и программной совместимости большинство современных ПК совместимы с IBM PC.

Персональные настольные компьютеры предназначены для работы в лабораторных условиях, в офисе или кабинете. Их располагают непосредственно на рабочем месте, обычно на столе, в соответствии с их названием. Это наиболее распространенные компьютеры, составляющие большую часть всех компьютеров в мире. Настольные персональные ЭВМ в зависимости от их возможностей и назначения можно разделить на профессиональные, офисные, учебные и бытовые.

Как правило, конструктивно настольные компьютеры и рабочие станции состоят из центральной части — системного блока — монитора, клавиатуры и мыши, подключенных к системному блоку. Конструктивное оформление системного блока отличается большим разнообразием — от классического горизонтального или вертикального до самых экзотических решений дизайнеров (рис. 2.10). В некоторых моделях ПК монитор и системный блок совмещены.



Рис. 2.10. Настольный компьютер с вертикальным (а) и горизонтальным (б) системным блоком

Переносные (мобильные) персональные компьютеры широко используются наравне с настольными компьютерами. Современные переносные компьютеры часто называют ноутбуками (от англ. notebook), или блокнотными компьютерами. Ноутбук функционально аналогичен настольному ПК и часто не уступает ему по техническим параметрам. В ноутбуках используется такое же ПО и ОС, что и в настольных ПК. Основная особенность ноутбука — возможность автономной работы с питанием от встроенного аккумулятора. Это позволяет использовать ноутбук в различных условиях при отсутствии питающей сети. Конструктивно ноутбук содержит жидкокристаллический дисплей, клавиатуру, совмещенную с системным блоком, жесткий диск и оптический дисковод (CD-ROM, CD-RW или комбинированный DVD+RW). Рядом с клавиатурой размещается манипулятор для управления курсором. Размеры ноутбуков соответствуют портфелю или небольшой сумке.

Широкое распространение ноутбуков сдерживалось их высокой стоимостью по сравнению с настольными компьютерами, однако по мере развития технологии изготовления для них комплектующих их стоимость снижалась, что обусловило повышение спроса и интенсивное развитие их производства. В настоящее время все основные производители настольных компьютеров предлагают большое число моделей ноутбуков, отличающихся функциональными возможностями и стоимостью. Более того, за последние годы появились новые виды ноутбуков:

- *мультимедийные* — отличаются достаточной производительностью и функциональными возможностями, необходимыми для комфортного решения большинства задач мультимедиа: качественное воспроизведение фильмов с DVD с многоканальным звуком; воспроизведение музыки на Hi-Fi уровне; редактирование и монтаж видеофайлов; 3D игры в высоком разрешении; большой экран с широкими углами обзора (как правило, 17" по диагонали); полный набор коммуникационных возможностей и интерфейсных портов; удобная полноразмерная клавиатура. Их стоимость обычно превышает 2500 долл.;



Рис. 2.11. Субноутбук (а) и планшетный ноутбук (б)

• *субноутбуки* — самые компактные и легкие модели (размеры менее И", вес до 2 кг, диагональ дисплея до 11") с достаточно высокой функциональностью, предназначенные в первую очередь тем, кто часто путешествует. Субноутбук может обеспечить неплохую комфортность в работе, однако высокой производительности от подобных компьютеров ожидать не следует. Часто их возможности ограничены офисными приложениями, интернет-браузером, почтовым агентом и прочими не особо требовательными к ресурсам приложениями (рис. 2.11, а). Их стоимость составляет около 2000 долл.;

• *планшетные ПК (Tablet PC)* — по оснащенности и габаритным размерам близки к субноутбукам, но оснащены сенсорным экраном, позволяющим выполнять различные операции с помощью стилуса или пальца, включая ввод рукописного текста и его распознавание. Выпускаются два вида планшетных ПК: чистые планшетные ПК (без клавиатуры) и планшетные ноутбуки (имеющие обычную клавиатуру и поворотно-откидной сенсорный экран (рис. 2.11, б)). Стоимость планшетных ПК составляет около 2 000 долл.

Карманные (или наладонные) переносные компьютеры (КПК) помещаются на ладони или в кармане. КПК также называют *наладонники* (от англ. — palmtop). Кроме палмтопов существуют карманные компьютеры, которые называют PDA (personal digital assistant). Общее название карманных компьютеров — *handhold computers* — компьютеры, которые держат в руках (рис. 2.12).

Все карманные компьютеры в зависимости от наличия клавиатуры делятся на две большие группы: КПК с клавиатурой и КПК без клавиатуры. КПК с клавиатурой внешне похожи на ноутбук, уменьшенный до карманных размеров. КПК без клавиатуры оснащены сенсорным экраном и информация вводится на экран при помощи специальной указки — стилуса, при этом может использоваться экранная клавиатура или написание символов стилусом непосредственно на экране. Стоимость карманных ПК колеблется от 300 до 1000 долл.

В карманных компьютерах программы хранятся в микросхемах энергонезависимой памяти. В набор программ обычно входит ОС, текстовый и графический редакторы, система баз данных и электронные таблицы, программы для



Рис. 2.12. Карманные компьютеры:

а — палмтоп; *б* — Pocket PC; *в* — PDA

работы в Интернете. Эти компьютеры позволяют обрабатывать документы, работать с базами данных, производить вычисления, читать электронные книги, слушать музыку, просматривать фильмы и работать в Интернете. Переносной и карманный компьютеры удобны для использования в поездках.

Карманные компьютеры в зависимости от используемой ОС также делятся на две группы — Palm OS и Windows Mobile. Причем в отличие от настольных компьютеров и ноутбуков конструкция КПК сильно связана с типом ОС и замена типа ОС для КПК возможна только теоретически. ОС Palm OS разработана специально для КПК и не предъявляет особых требований к их ресурсам. ОС Windows Mobile разработана фирмой Microsoft и хорошо интегрирована с ОС Microsoft для настольных компьютеров, что расширяет функциональные возможности КПК, однако при этом возрастают требования к ресурсам КПК.

Из всех перечисленных классов в России наибольшее распространение получили персональные ЭВМ, персональные рабочие станции и ноутбуки. Причем указанные классы ЭВМ базируются на процессорах семейства x86 фирмы Intel. Другие классы ЭВМ в России широкого применения не нашли. В связи с вышесказанным в дальнейшем основное внимание будет уделяться аппаратным средствам ЭВМ на платформе IBM PC с процессорами, совместимыми с архитектурой x86 фирмы Intel, относящимися к трем распространенным классам.

Каждая ВС обладает определенными функциональными возможностями обработки ГИ. Все возможности ВС реализуются совместно программными и аппаратными средствами и должны органично сочетаться с возможностями пользователя ЭВМ. Разделение функций между аппаратными и программными средствами направлено на повышение эффективности ВС при решении различных задач. Степень разделения функций между аппаратными и программными средствами зависит от уровня развития микроэлектроники и вычислительной техники.

Программные средства дешевы, гибки и доступны, аппаратные средства сложнее в реализации и специализированы. Соотношение по стоимостным затратам между аппаратными и программными средствами постоянно изменяется

шенными преобразованиями геометрических операндов являются булевские операции объединения, пересечения и вычитания. Возможности конструктивной твердотельной геометрии весьма значительны. Достаточно сказать, что ее средствами можно описать большую часть деталей из классификатора машиностроительных деталей.

Самый общий подход к моделированию состоит в представлении трехмерных тел в виде совокупности ограничивающих поверхностей, ребер и вершин. Каждая граничная оболочка представляет собой множество соединенных друг с другом поверхностей произвольной формы и включает в себя полное описание своих границ и связей с соседними фрагментами. Этот способ описания объектов называется граничным представлением, в англоязычной литературе и в интерактивных руководствах многих пакетов машинной графики он называется Boundary Representation, или В-гер. Представление тел с помощью границ позволяет моделировать трехмерные объекты самой сложной геометрии. Оно допускает множество формообразующих операций над телами, сохраняя при этом единый способ описания их внутреннего устройства.

3.4.2. Регулярные булевские операции

Пусть разработан эффективный способ описания геометрии объемных тел. Отвлечемся от технических деталей такого описания и будем опираться на гипотезу о его существовании. Из исходных простых геометрических форм можно построить более сложные производные объекты. Существует множество способов формообразования, основанных на комбинации геометрических операндов, но самой естественной техникой такого вида являются булевские операции. Операции сложения, вычитания и пересечения (рис. 3.37) являются обязательными в любой развитой системе геометрического моделирования. Иногда в их состав включают некоторые дополнительные способы синтеза, которые не обладают общностью трех перечисленных.

Если обычные булевские операции применить к объемным телам, то в результате можно получить объект с иной размерностью: поверхность, линию или даже

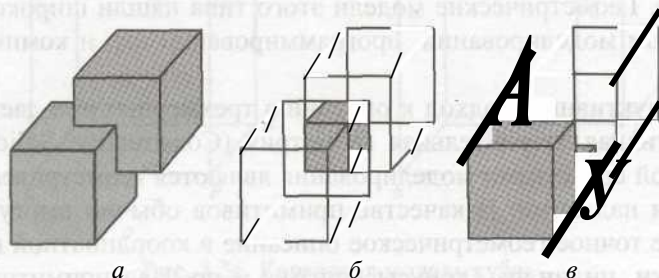


Рис. 3.37. Булевские операции на примере двух кубов:

$$a - A \cup B; \quad б - A \cap B; \quad в - A - B$$

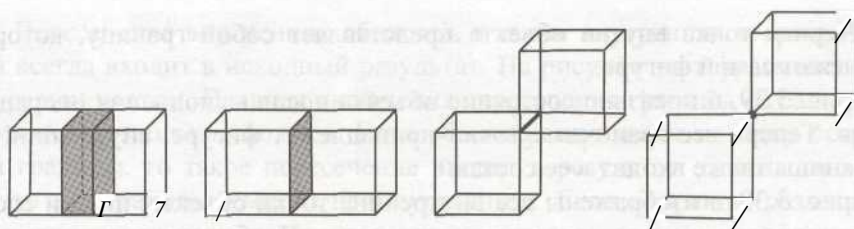


Рис. 3.38. Варианты операции булевского пересечения

точку. Справедливость этого тезиса подтверждает пример, приведенный на рис. 3.38, где показаны различные варианты пересечения двух кубических объемных тел.

Вместо обычных булевских операций будем использовать так называемые регулярные операции, которые обозначим привычными символами, но с добавлением верхнего индекса — звездочки (U^* , Γ^* , $-^*$). Регулярные операции, будучи примененными к объемным операндам, всегда дают геометрическое тело, обладающее объемом, например на рис. 3.38 регулярное пересечение дает непустой результат только в первом варианте.

Чтобы глубже исследовать разницу между обычными и регулярными булевскими операциями, рассмотрим объект, определенный как множество точек, которые разделены на внутренние и граничные. Граничными называются точки, имеющие нулевое расстояние до объекта и его дополнения. Множество граничных точек не всегда принадлежит самому объекту. Замкнутые объекты включают в себя границу, а открытые — нет. Объединение множества внутренних точек объекта и его границы обычно называется операцией замыкания. Регуляризацией множества называется замыкание множества внутренних точек. Множество, которое эквивалентно само себе после выполнения операции регуляризации, называется регулярным.

На рис. 3.39 приведен пример операции регуляризации. Исходное состояние объекта представлено на рис. 3.39, *а*. Он определен как множество внутренних точек, показанных серым цветом и множеством граничных точек, изображенных черным цветом. Часть границы, которая не входит в состав объекта, изображена пунктиром. Объект включает в себя несколько висячих фрагментов: две линии и

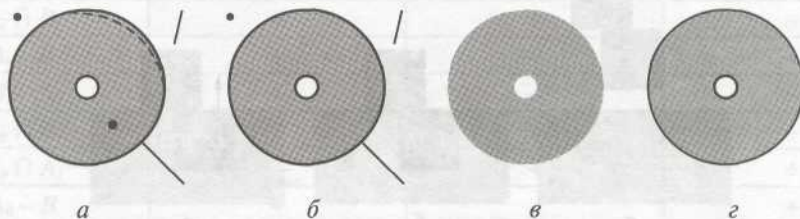


Рис. 3.39. Регуляризация объекта:

а — исходное состояние множества; *б* — множество после операции замыкания;
в — внутренняя область множества; *г* — регуляризованное множество

точку. Черная точка внутри объекта представляет собой границу, которая не принадлежит самой фигуре.

На рис. 3.39, б показано состояние объекта после выполнения операции замыкания. Теперь все граничные точки принадлежат фигуре, внутренний фрагмент границы также входит в ее состав.

На рис. 3.39, в изображены все внутренние точки объекта, иными словами, он представлен как открытое множество точек. Чтобы получить внутренние точки, следует отбросить все висячие и несвязные фрагменты.

На рис. 3.39, г изображен объект после выполнения операции регуляризации.

По определению, регулярное множество не может иметь граничных точек и фрагментов, не обладающих смежностью с некоторыми внутренними точками, поэтому для таких объектов исключается существование висячих граничных фрагментов любого вида (точек, линий или поверхностей).

Регулярные операции можно сформулировать в терминах обычных булевских операций:

$$A \cup^* B = \text{closure}(\text{interior}(A \cup B));$$

$$A \cap^* B = \text{closure}(\text{interior}(A \cap B));$$

$$A -^* B = \text{closure}(\text{interior}(A - B)).$$

Регулярные операции над регулярными операндами всегда дают регулярные множества.

Обычная булевская операция находит пересечение внутренних точек обоих операндов, и, кроме того, включает в состав новой фигуры пересечение границ с внутренними и граничными областями обеих исходных фигур. Регулярное пересечение состоит из пересечения внутренних областей операндов и пересечения внутренностей с границами другой фигуры за исключением пересечения границ.

Рассмотрим более подробно разницу между двумя типами операций на примере объектов, изображенных на рис. 3.40.

На рис. 3.40, а показаны два исходных объекта, которые являются операндами для обычного (рис. 3.40, б) и регулярного (рис. 3.40, в) булевого пересечения.

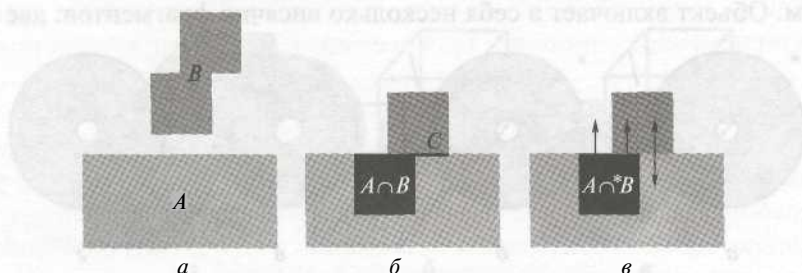


Рис. 3.40. Булевское пересечение:

а — исходное состояние операндов; б — булевское пересечение;

в — регулярное булевское пересечение

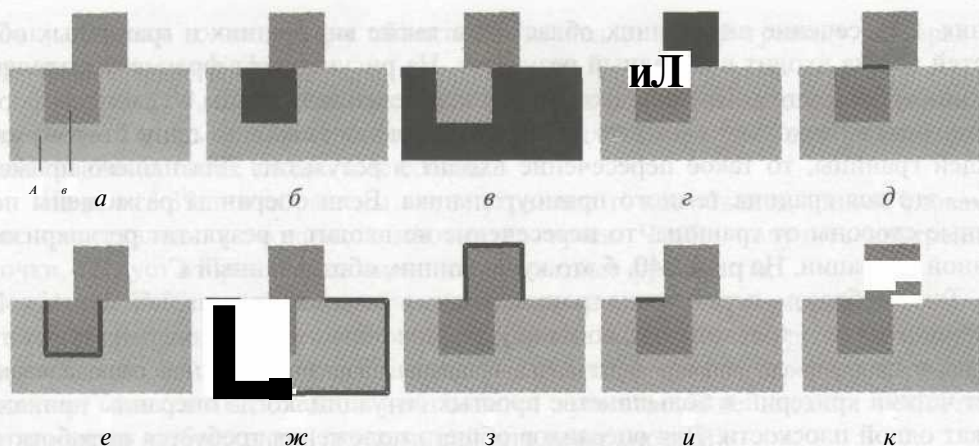


Рис. 3.41. Обычные булевские операции над элементами объектов:

a — операнды A и B ; $б$ — $A \cap B$; $в$ — $A \cup B$; $г$ — $B \setminus A$; $д$ — $A \setminus B$; $е$ — $B \cap A$; $ж$ — $A \setminus B$; $з$ — $B \setminus A$; $и$ — $\#(A_b \cap B_b)$; $к$ — $\sim(A_b \cap B_b)$

В табл. 3.2 приведены правила сведения регулярных операций к обычным. На рис. 3.41 на примере простейших форм показаны схемы этих операций. Буквами A и B обозначены операнды, нижние индексы $_i$ и $_b$ означают соответственно внутренние и граничные области, $\#(A_b \cap B_b)$ — часть общей границы объектов A и B , относительно которой A_i и B_i лежат по одну сторону, наконец, $\sim(A_b \cap B_b)$ означает часть общей границы, где A и B лежат по разные стороны. Каждая регулярная булевская операция сводится к набору обычных операций, которые в табл. 3.2 отмечены знаком «+».

Для всех регулярных операций каждый фрагмент результирующей границы принадлежит либо одному, или обоим операндам. Для операций $A \cup B$ и $A \cap B$ нормали к граням результирующего объекта наследуют свое положение от одного или обоих операндов. Для операции $A - B$ действует иное правило. Нормали граней результирующего тела, которое получено вычитанием B из A , имеют противоположное направление нормальям соответствующих граней B . Это следует из тождества $A - B = A \cap B^c$, где B^c — дополнение B . Тело B^c может быть получено дополнением внутренней части тела B с последующим обращением нормалей границы.

Во многих графических редакторах и системах автоматизированного проектирования пользовательский интерфейс основывается на использовании регулярных булевских операций. Эта техника предоставляет оператору удобный и надежный способ формирования сложных объектов из нескольких простых.

3.4.3. Параметрическое моделирование геометрии

Во многих отраслях техники сложный объект или система рассматривается как сочетание некоторого набора стандартных элементов. Очевидные преимущества

этого подхода к проектированию сделали привлекательной идею о такой организации процесса геометрического моделирования, когда облик изделия создается из стандартных форм — элементов некоторого геометрического конструктора. Эти элементы принято называть геометрическими примитивами. Примитивы, у которых зафиксированы все геометрические и конструктивные параметры, имеют ограниченную область применения, поэтому в геометрическом проектировании используются параметризованные примитивы. Например, примитивом будет многоугольник, для которого пользователь может задавать не только геометрические размеры, но и число сторон. Примитивом может быть элемент некоторого семейства объектов, члены которого различаются несколькими конструктивными, технологическими и геометрическими параметрами. Такой подход, названный групповая технология, используется в системах автоматизированного проектирования.

Параметризация примитивов часто применяется для определения таких сравнительно сложных объектов, как болты, гайки, зубчатые колеса и т. п. Конечно, эти формы допускают определение в терминах регулярных булевских операций, однако данный подход требует кропотливой работы над элементарными геометрическими телами, которые являются операндами сложных тел.

Примером сложного параметризованного примитива является зубчатое колесо, которое описывается набором из четырех геометрических и конструктивных параметров (рис. 3.42).

Привлекательная идея параметризации примитивов имеет несколько существенных ограничений. В этой парадигме не существует никаких средств для синтеза сложных объектов из набора простых. Для создания нового примитива требуется разрабатывать его код заново. Каждый примитив должен иметь программное окружение, состоящее из набора сервисных процедур и утилит. Например, в нее могут входить программы для расчета массы, объема, момента инерции и других конструктивных характеристик объекта. Как правило, утилиты такого типа являются уникальными: они создаются под спецификацию конкретного примитива и в общем случае не способны рассчитывать другие элементы базиса.

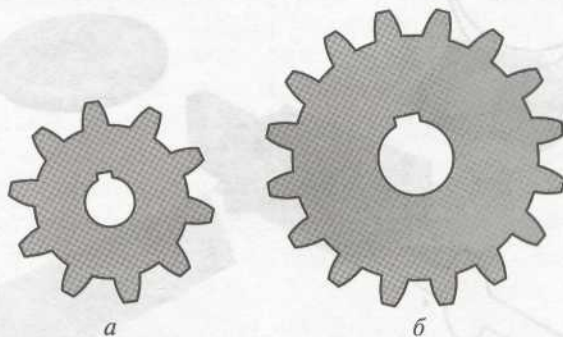


Рис. 3. 42. Параметризованное зубчатое колесо:

а — диаметр — 40, толщина — 10, отверстие — 10, число зубьев — 10;
б — диаметр — 60, толщина — 20, отверстие — 15, число зубьев — 16

3.4.4. Заметание

Заметанием (sweeping) называется способ формообразования при помощи движения объекта по некоторой траектории или согласно некоторому заданному закону. Это очень продуктивный метод, позволяющий получать трехмерные тела, которые с трудом воспроизводятся с помощью иных способов геометрического синтеза. Простейший пример этого метода генерации форм дает движение замкнутой двухмерной кривой вдоль прямолинейной траектории. Во многих пакетах машинной графики данный метод формообразования называют экструдированием (extrude). Экструдирование позволяет получить множество трехмерных форм, описывающих машиностроительные детали, элементы архитектурных конструкций, предметы интерьера и т. п.

Возможности формообразования значительно расширятся, если использовать траектории в виде кривых произвольной формы. В этом случае движение сечения ограничивается дополнительным условием, согласно которому в каждой точке пути нормаль сечения должна совпадать с касательной к траектории.

Вращение сечения вокруг некоторой фиксированной оси — еще один результативный способ генерации трехмерных форм, потенциально способный породить множество всех объектов, которые в инженерной практике именуются телами вращения. В англоязычной литературе этот способ называется rotational sweeping, что обычно переводят как заметание вращением.

Таким образом, если выбрать образующую, направляющую и закон движения, то множество точек пространства, которое заметет первый объект при смещении по второму, полностью определяет новое, в общем случае трехмерное тело. На рис. 3.43

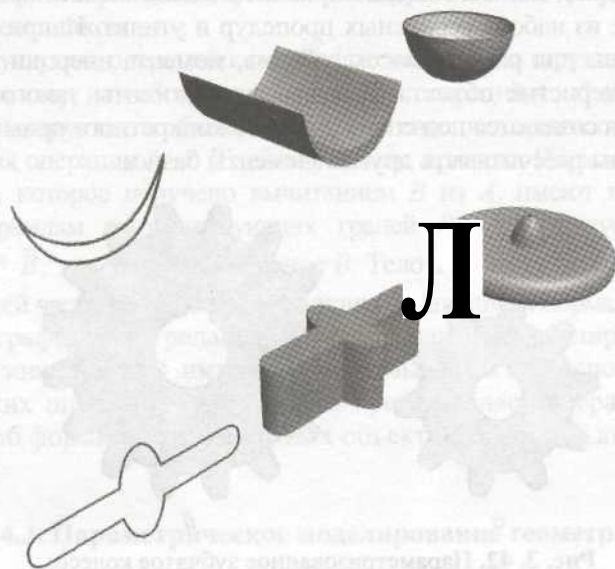


Рис. 3.43. Примеры кривых и трехмерных форм, порожденных их движением в пространстве

показаны примеры кривых и трехмерных тел, которые они порождают своим движением вдоль различных траекторий.

Чтобы получить трехмерную форму методом заметания, необязательно смещать двухмерное сечение, в некоторых случаях намного удобнее в качестве образующей использовать некоторый трехмерный объект. Этот метод формообразования используется при программировании металлообрабатывающих систем с числовым программным управлением и в системах управления роботами и робототехническими комплексами.

Следует отметить, что рассмотренный метод формообразования может иметь различные наименования в системах геометрического моделирования и литературе по КГ. Например, в популярной системе трехмерной графики 3Ds Max заметание называется Loft, в описаниях пакета на русском языке его именуют лофтингом. Заметание вращением (rotational sweeping) называется Lathe. Существуют и другие терминологические варианты, получившие распространение в кругах пользователей определенных пакетов трехмерной графики или систем автоматизированного проектирования.

Заметание — естественный способ генерации новых трехмерных форм. Он допускает простое управление и дает предсказуемые результаты. Однако его изобразительные возможности не безграничны, существуют трехмерные объекты, которые нельзя создать движением образующих. Заметание требует значительных вычислительных ресурсов для расчета трехмерных объектов, полученных движением по самопересекающимся траекториям. Кроме того, класс объектов, полученных этим способом, не является замкнутым относительно регуляризованных булевских операций.

На рис. 3.44 представлены две призмы, полученные смещением треугольников вдоль прямой траектории. Булевское объединение этих форм не может быть создано заметанием каких-либо двухмерных или трехмерных образующих.

Формообразование можно существенно расширить, если допустить использование нескольких образующих, связанных с одной траекторией. На рис. 3.45

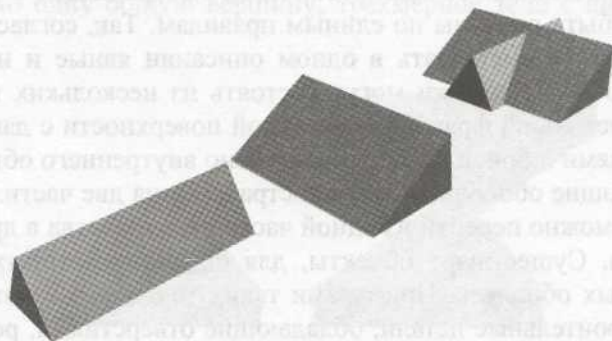


Рис. 3.44. Объединение двух трехмерных объектов, созданных экструдированием треугольников, не может быть получено заметанием

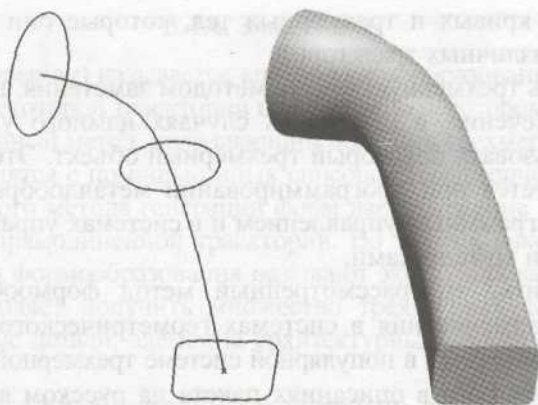


Рис. 3.45. Заметание с тремя образующими

показан пример трехмерного тела, полученного движением трех образующих вдоль одной кривой.

В некоторых пакетах машинной графики этот способ формообразования считается самостоятельным и называется скиннингом (skinning).

3.4.5. Граничное представление

Граничным представлением (Boundary Representation, B-rep) называется описание пространственных тел в терминах границ и их элементов. Это один из самых простых, выразительных и хорошо изученных способов определения трехмерной геометрии.

В описании границы тела участвуют объекты с интуитивно -понятным геометрическим содержанием: вершины, ребра, грани и оболочки. Рассмотрим подробнее топологические свойства граничных составляющих.

Оболочки тела должны обладать свойством однородности — это значит, что все они должны быть описаны по единым правилам. Так, согласно этому принципу, не разрешается смешивать в одном описании явные и неявные модели граничных оболочек. Оболочки могут состоять из нескольких граней. Каждая грань представляет собой фрагмент некоторой поверхности с данными о связях с соседними гранями и ориентации относительно внутреннего объема тела.

Ограничивающие оболочки делят пространство на две части: внутреннюю и внешнюю. Невозможно перейти из одной части пространства в другую без пересечения границы. Существуют объекты, для описания которых требуется несколько граничных оболочек. Примерами таких тел являются отливки с кавернами, машиностроительные детали, обладающие отверстиями, резервуары и пр. Геометрия всех названных примеров описывается, по крайней мере двумя оболочками: внешней и внутренней. Внутренние оболочки определяют пустоты и

располагаются внутри внешней. Оболочки любого типа не должны иметь пересечений друг с другом и самопересечений.

Во многих операциях автоматизированного конструирования и технологической подготовки производства важно различать внутренние и внешние точки деталей и узлов. Поэтому граничные оболочки считаются ориентированными, одна сторона их обращена внутрь тела, другая — наружу. Чтобы сделать ориентацию определенной каждой точке границы приписывается нормаль, которая направлена от оболочки в наружном направлении. Это соглашение представляется совершенно ясным для внешних оболочек, некоторые противоречия с геометрической интуицией могут возникать с нормальными внутренними фрагментами границы, которые имеют нормали, направленные внутрь ограничиваемой части пространства. Можно считать, что внутренние оболочки тела ведут себя как вывернутые наизнанку внешние.

Таким образом, для точного математического описания трехмерного тела требуется задать внешнюю оболочку и в зависимости от связности объекта одну или несколько внутренних оболочек.

Большая часть современных систем геометрического моделирования поддерживает представление, у которого фрагментами границ являются так называемые двумерные многообразия — геометрические объекты, каждая точка которых имеет окрестность, топологически подобную плоскому двумерному диску. Точное название отношения топологического подобия — гомоморфизм. Это абстрактная математическая категория, которая нуждается в четком и развернутом определении. С небольшой погрешностью подобными можно считать две геометрические фигуры, между точками которых можно установить непрерывное взаимно-однозначное соответствие.

На рис. 3.46 изображены образцы тел. Две призмы, показанные на рис. 3.46, б, соединены по одному ребру. Если взять любую точку на этом ребре и проанализировать ее окрестность, то можно заметить ее коренное топологическое отличие от двумерного диска. Примеры объектов, не относящихся к классу многообразий: соединение трехмерного тела с отдельным ребром; два многообразия, имеющие только одну общую вершину; трехмерное тело с внутренними перегородками.

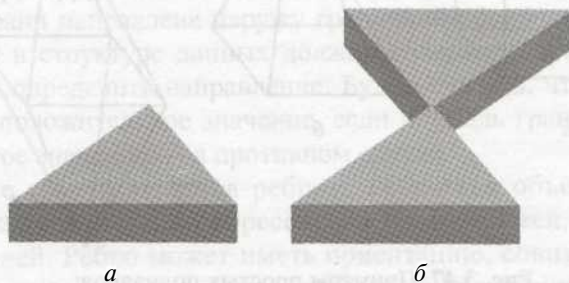


Рис. 3.46. Форма многообразного тела (а) и не многообразного тела (б)

Структура полуребер

Специалистам в области информатики и практикующим программистам хорошо известны все основные структуры данных и их сильные и слабые стороны. Так, очевидные недостатки массивов, которые служат простейшей формой описания таблиц, преодолеваются применением многосвязных списочных структур. Для хранения информации о граничном представлении трехмерных тел можно использовать такой способ организации данных, у которого основной единицей хранения будет двухсвязный список ребер грани.

Более того, для каждой грани создается указатель на первое ребро списка, а в описании ребра создаются указатели на предыдущее и последующие ребра. Пример такой организации данных в виде двухсвязного списка ребер грани F_1 показан на рис. 3.51.

Двухсвязный список позволяет использовать простой способ восстановления всей границы по любому предъявленному ребру. Для этого достаточно пройти весь граничный цикл, двигаясь по указателям. Если принять эту форму хранения для всех граней в виде двухсвязного списка, то появятся проблемы, связанные с целостностью представления всей оболочки. Пусть грань F_2 представлена списком со входом в E_2 (рис. 3.52). Очевидно, что упорядоченность ребер этого списка противоречит системе указателей списка, задающего грань F_1 . Общее для граней F_1 и F_2 ребро E_6 (см. рис. 3.49) получает различную ориентацию в списках обеих граней. Для разрешения этого противоречия можно разделить каждое ребро пополам и использовать половинки в описаниях тех граней, к которым они относятся. Части одного ребра приписывается противоположная ориентация, а описание граней представляет собой двухсвязный список полуребер. Элементы списочного описания грани связываются ссылками таким образом, что направление обхода каждого из них согласуется с направлением обхода грани (против часовой стрелки, если смотреть снаружи тела).

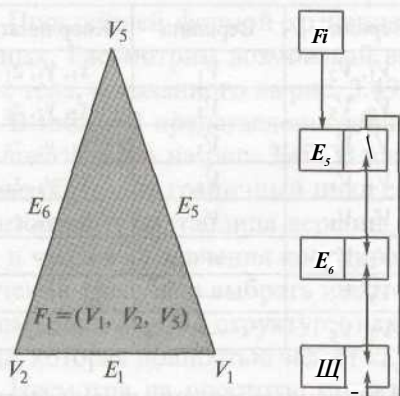


Рис. 3.51. Двухсвязный список ребер грани F_1

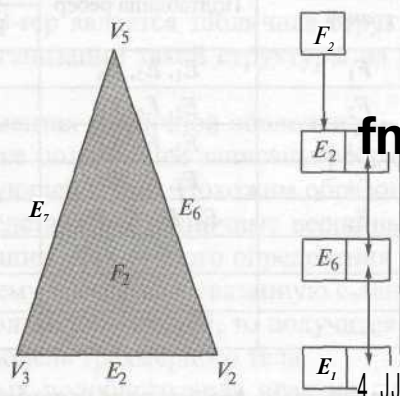


Рис. 3.52. Двухсвязный список ребер грани F_2

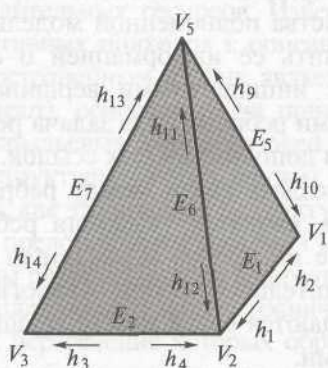
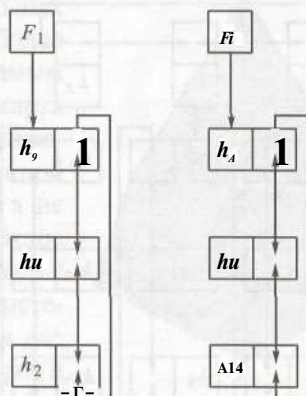


Рис. 3.53. Полуребра простого полиэдра

Рис. 3.54. Двухсвязные списки полуребер граней F_1 и F_2

Пример разбиения ребер на полуребра показан на рис. 3.53. Здесь представлены только две грани простого полиэдра, полное изображение которого дано рис. 3.49. На рис. 3.54 изображены две списочные структуры, которые задают границы граней F_1 и F_2 .

Структура полуребер позволяет компактно описывать грани с отверстиями без дополнительных ребер-мостиков. В общем случае грань может иметь сложную границу, состоящую из нескольких внутренних фрагментов, задающих отверстия, и одного внешнего граничного контура. Последовательность ребер, определяющая любой граничный фрагмент, представляет собой цикл, или петлю. Для описания грани с отверстиями можно организовать список списков, в котором входами служат циклы, а каждый цикл задается двухсвязным списком полуребер.

На рис. 3.55 приведен пример трехсвязной грани с разметкой полуребер, а на рис. 3.56 изображена списочная структура этого объекта. Из рис. 3.53 и 3.54 следует, что при этом способе организации данных грань ссылается на список полуребер косвенно, через двухсвязный список циклов. Входом этого списка служит обычно внешний цикл грани.

Последовательность перечисления полуребер в списках зависит от принятой ориентации циклов. Считается, что ребра внешнего цикла ориентированы против часовой стрелки, а ребра всех внутренних — по часовой, если смотреть на грань с внешней стороны тела (см. рис. 3.55). Это соглашение об ориентации сохраняет внутреннюю часть грани по одну сторону от границы. Внутренние точки грани всегда лежат по левую сторону при обходе циклов в выбранном направлении.

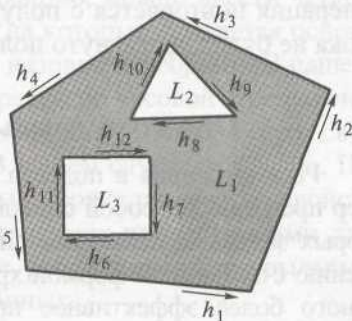


Рис. 3.55. Пример многосвязной грани

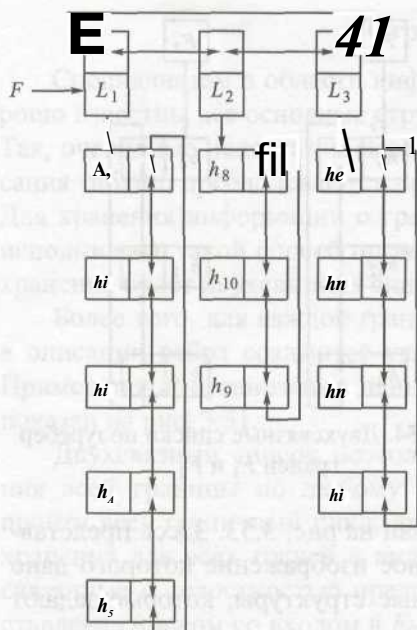


Рис. 3.56. Списочная структура грани с отверстием

Чтобы придать списочной структуре данных свойства полноценной модели, следует пополнить ее информацией о связях полуребер с инцидентными вершинами и родительскими ребрами. Эта задача решается созданием дополнительных ссылок.

Чтобы создать связи между ребрами и полуребрами, вводятся указатели ребер на порожденные полуребра и указатели полуребер на родительские ребра. Аналогичным образом создаются связи между вершинами и полуребрами.

Описанная структура данных имеет значительные преимущества по сравнению с табличным представлением. Отметим только самое главное. Эта форма описания позволяет сохранить информацию о связях вершин, ребер и граней и по этим данным синтезировать любые необходимые сведения о смежности.

Рассмотрим способ определения множества ребер, исходящих из вершины V_i , на примере простого многогранника, показанного на рис. 3.57.

Если вершина V_i является начальной вершиной (это справедливо для приведенного примера), то выбирается полуребро $prev\ h_u$ предшествующее h_i . Его родительское ребро представляет собой один из искомым объектов, соединенных с V_i . После этой операции вместо ребра hi рассматривается сопряженное ему полуребро (обозначим его $new\ hi$), и первый шаг алгоритма повторяется заново. Если V_i является конечной вершиной для h_i , то выбирается полуребро, следующее за h_i , а его родитель включается в число смежных ребер. Далее вместо h_i рассматривается полуребро, сопряженное со следующим за h_i первая операция повторяется с полуребром $new\ hi$. Процедура повторяется до тех пор, пока не будет достигнуто полуребро, сопряженное со стартовым hi .

Структура крыльевых ребер

Рассмотренная в подразд. 3.3 структура полуребер представляет собой список граней, каждая из которых задана двухсвязным списком ребер. По сравнению с табличной формой хранения данных это намного более эффективное представление оболочек твердых тел. В твердотельной вычислительной геометрии существуют задачи, вычисление которых в

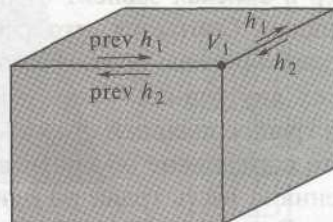


Рис. 3.57. Генерация данных о смежности ребер и вершин

структуре полуребер данных требует значительных вычислительных ресурсов. Известно несколько более эффективных подходов к описанию оболочек. Самым распространенным из них является подход Бомгарта (Baumgart), основанный на применении так называемых крыльевых ребер (winged edges). Сердцевиной этой структуры данных служит описание ребер, а не граней, как это принято в структуре полуребер. Каждое ребро представляется системой ссылок на связанные с данным ребром объекты многоугольника. В эту систему входят ссылки на две граничные вершины, на две грани, пересечение которых образует данное ребро, и четыре ребра, исходящих из граничных вершин.

Фрагмент многоугольника на рис. 3.58 иллюстрирует структуру данных крыльевых ребер. На рисунке в виде стрелок показаны все восемь ссылок ребра E_i .

Чтобы обеспечить более эффективное решение наиболее востребованных топологических задач, структура данных пополняется дополнительными ссылками:

- каждая вершина получает обратную ссылку на одно из ребер, исходящих из нее;
- каждая грань снабжается ссылкой на одно из граничных ребер.

При любом направлении обхода соседних граней (например, по часовой стрелке) вершины, принадлежащие общей грани, будут упорядочены в последовательности пройденных вершин в противоположном порядке. Пусть граничными вершинами некоторого ребра, общего для двух граней, являются вершины *нир*. Грань называют *p*-гранью, если при обходе ее границы по часовой стрелке вершина *p* встречается в перечислении вершин позже (стоит правее) вершины *n*. Грань называется *n*-гранью в противном случае.

Пусть граничная вершина V_1 ребра E_x (см. рис. 3.58) есть *n*, а вершина V_2 — *p*. Тогда, согласно введенной классификации, для ребра E_i грань F_1 есть *p*-грань, а грань F_2 — *n*-грань. Используя заданную классификацию, можно ввести четкое различие четырех смежных ребер (E_2, E_3, E_4, E_5), на которые ссылается основное ребро (E_i). Два ребра, инцидентные вершине *n*, назовем *и*-ребрами. В нашем случае (см. рис. 3.58) это ребра E_2, E_5 . При обходе граней по часовой стрелке они являются последующими для *и*-грани и предыдущими для *p*-грани. Ребра, связанные с вершиной *p*, будем называть *p*-ребрами. В нашем случае это E_4, E_3 . По сравнению с *n*-вершинами они демонстрируют прямо противоположные свойства. При фиксированном направлении обхода они являются последующими для *p*-грани и предыдущими для *и*-грани. Эти четыре ребра называют крыльями (wings), от них и пошло название всей структуры данных.

Описанная структура данных предназначена для хранения данных об оболочках, состоящих из односвязных граней. Существует несколько модификаций этой

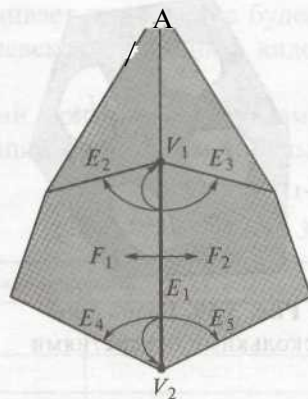


Рис. 3.58. Фрагмент многоугольника и ссылки структуры крыльевых ребер

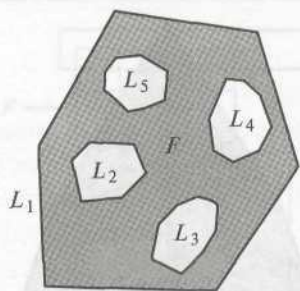


Рис. 3.59. Грань с несколькими отверстиями

структуры, позволяющих расширить ее возможности на грани с отверстиями. Например, можно хранить описание каждой грани в виде списка граничных циклов подобно тому, как это делается в структуре полуребер. Для каждой грани сохраняется указатель на внешний цикл, который, в свою очередь, ссылается на внутренний граничный цикл, если он имеется. Последний указывает на другой цикл при наличии нескольких отверстий в одной грани или на внешний цикл, в ином случае.

Пример такого представления иллюстрируют рис. 3.59 и 3.60. На рис. 3.59 показана грань с отверстиями, а на рис. 3.60 представлена списочная структура, которая задает топологию этой грани.

Выбор односвязного списка вместо двухсвязного не имеет принципиального значения. Односвязный список позволяет получить более экономную структуру, что достигается за счет некоторой потери эффективности расчета для некоторых классов задач.

Булевские операции в системе В-гер

Синтез новых тел из набора исходных объектов путем применения некоторого множества выбранных операций — мощный и естественный способ формообразования. В компьютерной геометрии известно несколько классов допустимых операций, применимых к трехмерным телам. Одними из самых распространенных являются так называемые булевские (булевы) операции. К их числу прежде всего относятся объединение, вычитание и пересечение двух тел. Можно утверждать, что не существует развитых систем геометрического моделирования, которые не поддерживают этот метод синтеза трехмерных и двумерных форм.

В общем случае результатом булевских операций могут быть не многообразные объекты, работа с которыми ограниченно поддерживается в современных системах машинной графики. Напомним, что i -мерным многообразием называется тело, расположенное в i -мерном пространстве, каждая точка которого (тела) имеет окрестность, топологически подобную n -мерной сфере.

Существует условие, которое немного ограничивает разнообразие синтезируемых форм, но гарантирует их высокую «технологичность». Это дополнительное условие называется регуляризацией, его формулировка и основные свойства подробно рассмотрены в начале главы. Способ

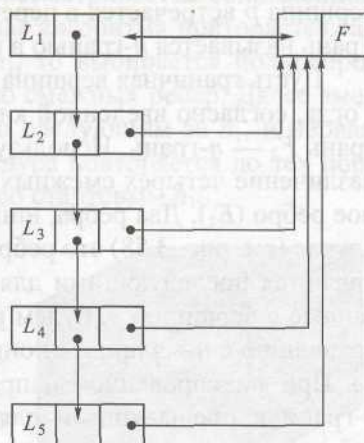


Рис. 3.60. Списочная структура, задающая грань с отверстиями

Техника булевого объединения

Рассмотрим операцию булевого сложения двух трехмерных тел. Суть операции достаточно проста. Требуется найти линии пересечения граничных оболочек двух тел, затем удалить часть первого тела, которая расположена внутри второго, и кусок второго тела, попавшего внутрь первого. Оставшиеся фрагменты образуют новое тело, которое представляет собой булевое объединение.

Операцию можно разбить на три этапа. Первый этап состоит в получении линий пересечения граничных оболочек двух тел. По этим линиям строят новые ребра, которые будем называть ребрами пересечения. На втором этапе ищут точки пересечения новых ребер со старыми ребрами, присутствовавшими в описании границ операндов до операции. Эти ребра разрежем на части по новым вершинам. Третий этап заключается в перестройке граничных циклов пересекающихся граней. Рассмотрим содержание этапов более подробно.

Первый этап операции булевого объединения начинается с поиска линий пересечения каждой грани первого тела с каждой гранью второго тела, если таковые (линии) имеются. Технически этот шаг реализуется при помощи хорошо известного алгоритма поиска линии пересечения двух поверхностей, который рассмотрен в гл. 4, посвященной алгоритмическому обеспечению КГ.

Пусть построены искомые линии пересечения. На их базе создадим ребра пересечения. Всем новым ребрам припишем направление, которое совпадает с ориентацией векторного произведения нормали грани первого тела с нормалью грани второго тела. Будем считать положительной ориентацию нормали, направленной наружу тела (рис. 3.61). Ребра пересечения должны целиком лежать внутри граничных циклов исходных тел. Они могут соединяться со старой границей только своими концевыми вершинами. Это условие, примененное к примеру, показанному на рис. 3.61, требует разделения ребер старой границы на две части в точках А, В и С.

На втором этапе процедуры построения булевого объединения требуется разрезать ребра старой границы в точках касания новых ребер. Это разделение

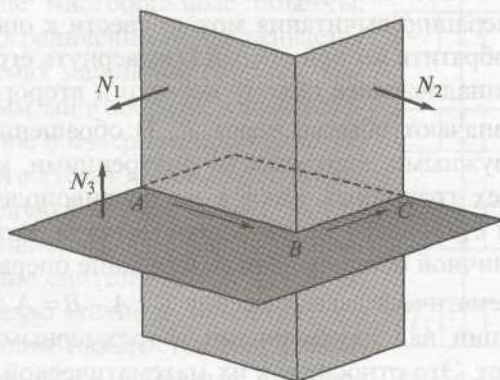


Рис. 3.61. Нормали граней и направления ребер пересечения

выполняется посредством рассечения кривых, которые являются носителями ребер. Из одной кривой получим две кривые, которые при соединении образуют исходную кривую. Одна из этих кривых останется геометрическим носителем разрезаемого ребра, а на базе второй построим новое ребро, которое наследует свойства исходного ребра. Граничные ребра строятся на основе кривой пересечения двух поверхностей. Рассечению подлежат два экземпляра этой двухмерной кривой, лежащие на разных поверхностях.

Точки пересечения нового ребра со старым ребром грани ищем как точки пересечения двухмерных кривых, заданных на общей для них плоскости параметров. От каждого ребра в формуле, определяющей точки пересечения линий, участвует по одной двухмерной кривой, входящей в линию пересечения. Параметры точек пересечения ребер и сами координаты кривых, являющиеся параметрами поверхности, должны быть определены с заданной точностью. Поскольку каждое ребро исходных тел входит в циклы двух смежных граней, то после резки ребер исходных тел необходимо произвести корректировку этих циклов с учетом проведенного разрезания.

В результате выполнения первых двух этапов получим совокупность ориентированных ребер пересечения, которые соединяются друг с другом и со старыми ребрами тел-операндов только в вершинах.

Третий этап является заключительным в операции построения булевского объединения. Его цель состоит в разрезании граней и перестройке граничных циклов в соответствии с новой структурой граничных ребер.

На рис. 3.62 показаны грани двух пересекающихся тел, представленных на рис. 3.61. Тонкими линиями со стрелками изображены направления граничных циклов исходных граней, толстыми линиями со стрелками — ребра разрезания. Требуется перестроить граничные циклы таким образом, чтобы учесть новую ситуацию, созданную разрезанием. На рис. 3.63 показаны части граней, которые войдут в результирующее объединение. Как и ранее, тонкие линии со стрелками

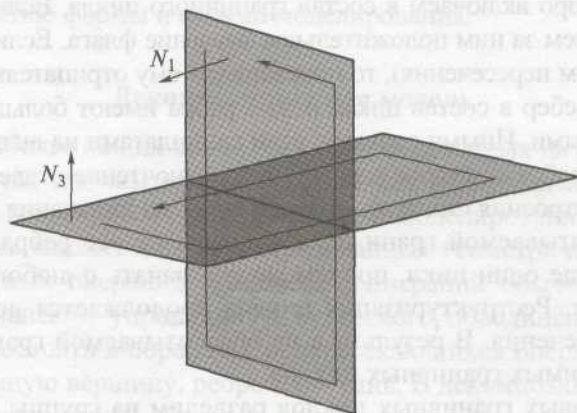


Рис. 3.62. Грани тел до разрезания