Федеральное агентство по образованию
Московский государственный строительный университет

Кафедра «Информационные системы и технологии управления в строительстве»

Конспект лекций
по дисциплине
«Компьютерная графика»
для студентов специальности 230102
«Автоматизированные системы обработки информации и управления»

Москва 2009 г.

ОГЛАВЛЕНИЕ

Глава 1. Основные понятия

Важнейшая функция компьютера - обработка информации. Особо можно выделить обработку информации, связанную с изображениями. Она разделяется на три основные направления: компьютерная графика (КГ), обработка и распознавание изображений.

Задача компьютерной графики (Computer Graphics) - визуализация, то есть создание изображения. Визуализация выполняется, исходя из описания (модели) того, что нужно отображать. Существует много методов и алгоритмов визуализации, которые различаются между собою в зависимости от того что и как отображать. Например, отображение того, что может быть только в воображении человека — график функций, диаграмма, схема, карта. Или наоборот, имитация трехмерной реальности — изображение сцен в компьютерных играх, художественных фильмах, тренажерах, в системах архитектурного проектирования. Важными и связанными между собою факторами здесь являются: скорость изменения кадров, насыщенность сцены объектами, качество изображения, учет особенностей графического устройства.

Обработка изображений (Computer Vision) — это преобразования изображений. Входными данными является изображение, и результат обработки — тоже изображение. Примерами обработки изображений могут служить: повышение контраста, чёткости, коррекция цветов, редукция цветов, сглаживание, уменьшение шумов и так далее. В качестве материала для обработки могут использоваться космические снимки, сканированные изображения, радиолокационные, инфракрасные изображения и т. п. Задачей обработки изображений может быть как улучшение в зависимости от определенного критерия (реставрация, восстановление), так и специальное преобразование, кардинально меняющее изображения. В последнем случае обработка изображений может быть промежуточным этапом для дальнейшего распознавания изображения. Например, перед распознаванием часто необходимо выделять контуры, создавать бинарное изображение, разделять по цветам. Методы обработки изображений могут существенно отличаться в зависимости от того, каким путем получено изображение — синтезировано системой КГ либо это результат оцифровки черно-белой или цветной фотографии.

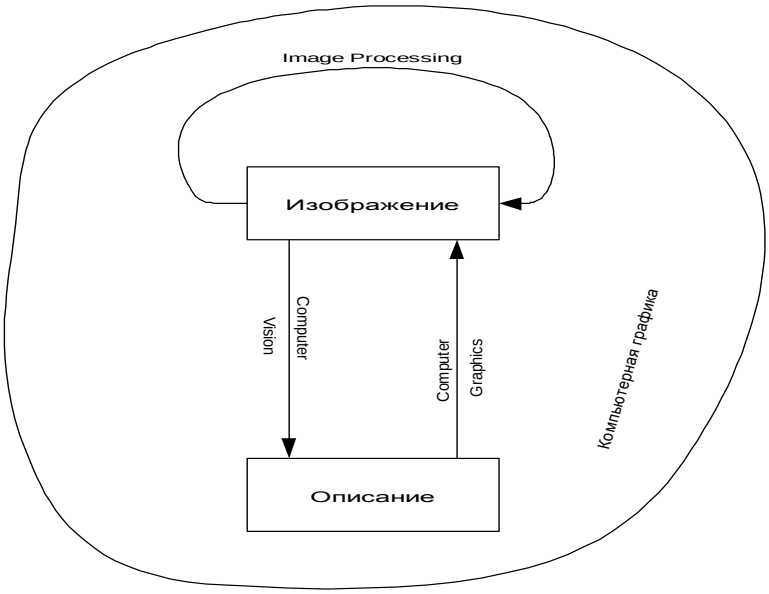


Рис. 1.1. Направления компьютерной графики

Для **распознавания изображений (Image Processing)** основная задача — получение описания объектов, представленных изображением. Методы и алгоритмы распознавания разрабатывались прежде всего для обеспечения зрения роботов и для систем специального назначения. Но в последнее время компьютерные системы распознавания изображений все чаще появляются в повседневной практике многих людей, например, офисные системы распознавания текстов, программы векторизации, создание трехмерных моделей человека.

Цель распознавания может формулироваться по-разному: выделение отдельных элементов (например, букв текста на изображении документа или условных знаков на изображении карты); классификация изображений в целом (например, проверка того, есть ли это изображение определенного летательного аппарата, или установление персоны по отпечаткам пальцев).

Методы классификации и выделение отдельных элементов могут быть тесно связаны между собою. Так, классификация может быть сделана на основе структурного анализа отдельных элементов объекта. Или для выделения отдельных элементов можно использовать методы классификации. Задача распознавания является обратной относительно визуализации.

Стоит отметить, что довольно популярным до недавнего времени было словосочетание **интерактивная компьютерная графика.** Им подчеркивалась способность компьютерной системы создавать графику и вести диалог с человеком. Прежде системы работали в пакетном режиме- способы диалога были не развиты. В настоящее время почти любую программу можно считать интерактивной системой КГ.

1.1 Разновидности компьютерной графики

Распространение компьютерной графики началось с полиграфии. Но вскоре она вырвалась из тесных помещений типографий на простор широкого применения. Огромную популярность завоевали компьютерные игры, научная графика и фильмы. Сейчас без развитой и изощренной графики не обходится ни один фантастический фильм, ни одна компьютерная игрушка. Создаются изображения настолько реальные, что трудно поверить в то, что все это создано на компьютере. Мощнейшие машины и талантливейшие команды математиков, программистов и дизайнеров работают над этим. Ни один приличный доклад в сфере бизнеса не обходится сейчас без компьютерной презентации.

Из простого перечисления областей применения видно, что понятие компьютерной графики довольно обширно — от алгоритмов, рисующих на экране причудливые узоры, до мощных пакетов 3D-графики и программ, имитирующих классические инструменты художника. Иными словами, компьютерная графика не является простым рисованием при помощи компьютера, а представляет собой довольно сложный комплекс, который находит применение во многих областях человеческой деятельности:

- двухмерная графика;
- полиграфия;
- web-дизайн;
- мультимедиа;
- 3D-графика и компьютерная анимация;
- видеомонтаж;
- САПР и деловая графика;
- геоинформационные системы.

Сферы применения компьютерной графики чрезвычайно разнообразны. Каждый ее раздел имеет свои отличительные особенности и тонкости «технологического производства». Для каждого из них создано свое программное обеспечение, включающее разнообразные специальные программы (графические редакторы). Вне зависимости от области использования каждый графический редактор, как правило, должен иметь:

- инструменты рисования на компьютер;
- библиотеку готовых изображений;
- набор шрифтов;
- набор спецэффектов;
- а также быть совместимым с другими графическими программами.

Остановимся на некоторых характерных чертах, присущих отдельным областям компьютерной графики, попутно затрагивая используемые в них программные средства.

Полиграфия

Компьютерная графика начала своё распространение с полиграфии. ***Полиграфия*** – довольно сложное направление, требующее от работающего в этой области наибольшей широты знаний. Даже на поверхностный взгляд работа в полиграфии довольно разнообразна: создание визиток, бланков, рекламных листовок, буклетов и плакатов; работа в периодических изданиях (часто имеющих свою специфику). Для реализации этих задач предназначены специальные программы верстки.

Программы верстки дают возможность соединять вместе текстовую и графическую информацию для создания информационных бюллетеней, журналов, брошюр и рекламной продукции. Среди наиболее популярных программ можно выделить Adobe PageMaker и QuarkXPress. Большинство программ верстки страниц используется для компоновки различных элементов на странице, а не для того, чтобы с нуля создавать в них текстовые или графические файлы. Тексты объёмных документов, как правило, пишутся (набираются) в системах обработки текстов (текстовых редакторах типа MS Word), а затем импортируются в программы верстки. Графика часто создаётся в программах черчения (деловой графики) и редактирования изображений, а затем импортируется в программу верстки страниц. Хотя все основные программы верстки страниц обладают примерно одними и теми же возможностями, свою популярность они завоевали по разным причинам. Например, PageMaker традиционно считается самым лёгким в использовании продуктом среди программ верстки страниц, в первую очередь из-за того, что в нём использован визуальный образ, знакомый большинству художников и дизайнеров. Конкурент и аналог PageMaker – QarkXPress – обычно используется для компьютеров на платформе Macintosh. Пакеты компьютерной графики для полиграфии позволяют дополнять текст иллюстрациями разного происхождения, создавать дизайн страниц и выводить полиграфическую продукцию на печать с высоким качеством.

Мультимедиа

Мультимедиа – это область компьютерной графики, связанная с созданием интерактивных энциклопедий, справочных систем, обучающих программ и интерфейсов к ним.

В отличие от полиграфии, где дизайнер-полиграфист сотрудничает с печатником, дизайнер-мультимедийщик сотрудничает с программистом. Здесь требования к графике уже другие. Так, в полиграфии, например, файлы должны были иметь достаточно большое разрешение. В результате размеры файлов могут составлять десятки и даже сотни мегабайтов. В мультимедиа же ограничением служит разрешение экрана монитора и требование минимизации размеров файлов. Здесь контроль за качеством проще, чем в полиграфии, для него достаточно хорошего монитора.

Для работы в этой области наряду с графическими редакторами необходимо знать программы создания мультимедиа- например Macromedia Director или MS Power Point. В создании новых версий презентационных пакетов можно отметить тенденцию всё более полного использования мультимедиа-возможностей и Интернета. Эти программы допускают удобный импорт видео- и звуковых файлов, в них предусмотрены средства анимации диаграмм.

World Wide Web (WWW)

Важным событием в жизни общества стало появление глобальной сети Internet. Сейчас происходит бурное развитие этой сети. Возрастают мощности каналов передачи данных, совершенствуются способы обмена и обработки информации. Сеть Internet используют всё больше людей в разных странах. Это способ общения людей, обмена информацией, сближения языков, распространения идей, новое пространство для бизнеса и т.п. Важное место в Internet занимает компьютерная графика. Всё больше совершенствуются способы передачи визуальной информации, разрабатываются более совершенные графические форматы, ощутимо желание использовать трёхмерную графику, анимацию, весь спектр мультимедиа.

Требования к созданию изображений для WWW очень противоречивы. С одной стороны, жёсткие ограничения по снижению размеров файлов для минимизации времени их передачи в сети, с другой - необходимость сохранения качества передаваемой по сети "картинки". Каждый формат графических изображений, применённый в WWW, имеет свои особенности: JPEG, например, хорош для фотографий, а GIF – для векторных изображений. К тому же WWW имеет свою область цветового охвата, что необходимо учитывать при создании изображений.

3D-графика и компьютерная анимация

Это ещё одно широкое и по-своему сложное направление, особый мир. 3D-графика – это создание искусственных предметов и персонажей, их анимация и совмещение с реальными предметами и интерьерами. В настоящий день определилось несколько перспективных направлений её использования.

- Широкое применение 3D-графика находит в индустрии компьютерных игр. Анимационные заставки, интерфейсы и персонажи компьютерных игр создаются в программах 3D-графики.
- Другая область применения 3D-графики–телевизионная реклама и оформление телевизионных каналов.
- Многие архитекторы и дизайнеры используют 3D-графику для построения макетов зданий и трёхмерных моделей архитектурных памятников, которых ещё не существует в природе.

Освоение 3D-графики требует немало времени и мощных системных ресурсов. Чтобы результат выглядел фотореалистично, необходимо освоить не только 3D-моделирование, но и уметь правильно осветить сцену, найти хороший ракурс камеры, подобрать материал и текстуры. Всё это существенно влияет на качество графики.

САПР и деловая графика

Системы автоматизированного проектирования были исторически первыми интерактивными системами *(САПР* - английская аббревиатура *CAD* - Computer Aided Design), кото*рые* появились в 60-х годах. Они представляют собой значительный этап эволюции *компьютеров* и программного обеспечения. В системе интерактивной КГ пользователь воспринимает на дисплее изображение, представляющее некоторый СЛОЖНЫЙ объект и может вносить изменения в описание (модель) объекта (рис. 1.2). Такими изменениями могут быть как ввод и редактирование отдельных элементов, так и задание числовых значений для любых параметров, а также другие операции по вводу информации на основе восприятия изображений.

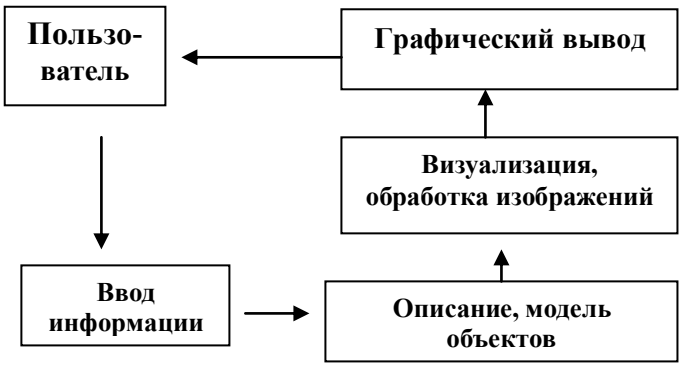


Рис. 1.2. Модель интерактивной компьютерной графики

Системы типа САПР активно используются во многих областях, например, в машиностроении и электронике. Одними из первых были созданы САПР для проектирования самолетов, автомобилей, системы для разработки микроэлектронных интегральных схем, архитектурные системы и т.п. Такие системы сначала функционировали на довольно больших компьютерах. Потом получили распространение быстродействующие компьютеры среднего класса с развитыми графическими возможностями — графические рабочие станции. С возрастанием мощностей персональных компьютеров все чаше САПР начали использовать на дешевых массовых компьютерах, которые сейчас имеют достаточное быстродействие и объемы памяти для решения многих задач. Это привело к широкому распространению систем САПР.

● Одно из главных применений составляет их использование в различных областях инженерной конструкторской деятельности – от проектирования микросхем до создания самолётов.

● Другой важной областью применения САПР является строительство и архитектура.

● САПР используется и в медицине. Например, автоматизированное проектирование имплантантов, особенно для костей и суставов, позволяет минимизировать необходимость внесения изменений в ходе операции, что сокращает время пребывания на операционном столе (результат положительный как с точки зрения пациента, так и с точки зрения врача).

Геоинформационные системы (ГИС)

Сегодня становятся все более популярными. Это относительно новая для массовых пользователей разновидность систем интерактивной компьютерной графики. Они интегрируют методы и технологии разнообразных областей - баз данных, геодезии, картографии, космонавтики, навигации и, конечно, компьютерной графики. Известны такие системы, как ArcGIS, AutoCAD Map, Maplnfo. Пример отечественных систем — ГИС "ОКО", "Визиком-Киев".

Системы типа ГИС могут использовать значительные ресурсы компьютерных систем как в плане работы с базами данных, так и для визуализации объектов, находящихся на поверхности Земли. Причем визуализацию необходимо делать с разными степенями детализации — как для Земли в целом, так и в границах отдельных участков.

Типичными для любой ГИС являются такие операции — ввод и редактирование объектов с учетом их расположения на поверхности Земли, формирование разнообразных цифровых моделей, запись в базы данных, выполнение разнообразных запросов к базам данных Важной функцией ГИС является анализ пространственных, топологических отношений множества объектов, расположенных на какой-то территории. Одной из функций также является спутниковая GPS-навигация.

1.2. Принципы организации графических программ

Многие пользователи ПК связывают понятие компьютерной графики с программами, предназначенными для редактирования двухмерных цифровых изображений. Это программное обеспечение по принципу действия и функциональному назначению можно разделить на 3 группы:

- растровая графика
- векторная графика
- фрактальная графика

Наиболее широко в компьютерной графике представлены первые 2 типа программ: растровые и векторные. Важно понимать принципиальные отличия между двумя этими типами ПО, так как каждый из них имеет свои сильные и слабые стороны.

О фрактальной графике разговор особый. Она занимает промежуточное положение между растровыми и векторными программами. Кроме того, фрактальные узоры часто используют в качестве красивых фронтальных заливок в редакторах растровой и векторной графики.

Двухмерная, или 2D-графика,– это основа всей компьютерной графики (в том числе и 3D-графики). Ни один компьютерный художник-дизайнер не может плодотворно работать над своими проектами без понимания базовых положений двухмерной графики.

Растровые программы

Большинство программ для редактирования изображений-Addobe Photoshop, Corel PHOTO-PAINT или MS Paint являются растровыми программами. В них изображение формируется из решётки крошечных квадратиков, именуемых *пикселами*. Поскольку каждый пиксел на экране компьютера отображён в специальном месте экрана, то программы, которые создают изображение таким способом, называют побитовыми, или программами с побитовым отображением (bitmap). Решётку (или матрицу), образуемую пикселами, называют растром. Поэтому программы с побитовым отображением также называются *растровыми* программами.

Как создаётся цифровое изображение? Многие программы для обработки изображений, такие как Addobe Photoshop, позволяют выбирать нужные электронные кисть, цвет и краску. Иногда конечный результат неотличим от традиционной живописи, но, в общем, возможности компьютера гораздо шире традиционных.

Большинство цифровых изображений сначала поступают в компьютер при помощи сканера или цифрового фотоаппарата. С помощью сканера можно оцифровать слайд, фотографию путём преобразования изображения в цифровые данные. Методика сканирования изображения с последующими операциями цветокоррекции и ретуширования наиболее часто используется в печатной компьютерной продукции, в первую очередь при создании рекламных объявлений и обложек журналов. Компьютер может поменять цвет вашей причёски или глаз, отретушировать родинку на щеке, изменить цвет или фон вашей фотографии, а также убрать все недостатки и дефекты. Для привлечения внимания зрителей компьютерные художники часто добавляют к фотографиям в журналах и рекламным объявлениям специальные эффекты, создавая сложные коллажи.

Процесс оцифровывания изображения посредством цифрового фотоаппарата несложен – человек просто направляет аппарат на объект съёмки и нажимает спуск. Изображение мгновенно оцифровывается и записывается в запоминающее устройство внутри фотоаппарата. Вам не нужно покупать и проявлять плёнку – её просто нет. Вместо вывода изображения на слайды или печати фотографий оно загружается в компьютер по кабельной линии. Когда изображение появляется на экране компьютера, вы можете изменять его цвета, ретушировать, крутить-вертеть, изгибать, искажать для создания специальных эффектов в программах-редакторах изображений: Adobe Photoshop, Corel PHOTO-PAINT или каких-то других, более удобных для пользователя.

Растровые программы предназначены в основном для редактирования изображений, обеспечивая возможность цветокоррекции, ретуши и создания специальных эффектов на базе цифровых изображений. Пользуясь программными продуктами для формирования изображений, такими как Adobe Photoshop или Corel PHOTO-PAINT, вы можете создавать коллажи, виньетки, фотомонтажи и подготавливать цветные изображения для вывода на печать. На сегодняшний день программы редактирования изображений используются при производстве практически всех печатных изображений, где необходима фотография. Их применяют для стирания морщин с лиц моделей, придания ярких красок мрачным и пасмурным дням и изменения общего настроения посредством специальных световых эффектов. Они также широко используются производителями мультимедиа для создания текстовых и фоновых эффектов и для изменения количества цветов изображения.

Векторные программы

Изображение, созданное в векторных программах, основывается на математических формулах, а не на координатах пикселов. Составляющие основу таких изображений кривые и прямые линии называются векторами. Так как при задании объектов на экране используются математические формулы, то отдельные элементы, изображения, создаваемые в векторных программах, – например, Adobe Illustrator, CorelDRAW и Macromedia FreeHand, – можно легко перемещать, увеличивать или уменьшать без проявления «эффекта ступенек». Так, для перемещения объекта достаточно перетащить его мышью. Компьютер автоматически пересчитывает его размер и новое местоположение.

Поскольку в этом случае изображение создаётся математически, векторные программы используются тогда, когда нужны чёткие линии. Они часто применяются при создании логотипов, шрифтов для вывода на плоттер и различных чертежей.

Когда вы видите изображение, созданное в векторной программе, его качество зависит не от исходного разрешения изображения, а от разрешающей способности устройства вывода (монитора, принтера, плоттера…). Так как качество изображения не основывается на разрешении, то изображение, созданное в векторных программах, как правило, имеет меньший объём файлов, чем построенное в программах побитового отображения. В векторных программах нет проблем и со шрифтами – большие шрифтовые массивы не образуют файлов огромного размера.

Фрактальные программы

Фрактал - это объект довольно сложной формы, которая получена в результате выполнения простого итерационного цикла над формой начальной, элементарной.

Одним из основных свойств фракталов является самоподобие. Объект называют самоподобным, когда увеличенные части объекта походят на сам объект и друг на друга.

Таким образом, в простейшем случае небольшая часть фрактала содержит информацию обо всем фрактале. Например, снежинка несет информацию о снежном сугробе, а горный камень имеет те же самые очертания, и что и горный хребет. Благодаря этому свойству можно использовать фракталы для генерирования поверхности местности, которая походит на саму себя, независимо от масштаба, в котором она отображена. Программы, получающие в последнее время широкое распространение и созданные по принципу генерации самоподобных фигур, являются прекрасным инструментом в руках дизайнера, художника, разработчика WEB-приложений.

Отдельное перспективно напрвление развития фрактальных программ — создание алгоритма фрактального сжатия графической информации.

Более подробно о фрактальной графике будет рассказано в главе 5.

Глава 2. Координаты и преобразования

2.1 Координатный метод

Координатный метод был введен в XVII веке французскими математиками *Р. Декартом и П. Ферма.* На этом методе основывается *аналитическая геометрия,* которую можно считать фундаментом КГ. В современной КГ координатный метод широко используется.

2.1.1. Преобразование координат

Сначала рассмотрим общие вопросы преобразования координат. Пусть задана *n-мерная* система координат в базисе (k_1, k_2,.... k_n), которая описывает положение точки в пространстве с помощью числовых значений κ_i. В КГ наиболее часто используются двумерная (n = 2) и трехмерная (n = 3) системы координат.

Если задать другую, *N-мерную,* систему координат в базисе (m_1, m_2,..., m) и поставить задачу определения координат в новой системе, зная координаты в старой, то решение (если оно существует) можно записать в таком виде:

$$\begin{cases} m_1 = f_1(k_1, k_2, \ldots, k_n), \\ m_2 = f_2(k_1, k_2, \ldots, k_n), \\ \quad \ldots \\ m_N = f_N(k_1, k_2, \ldots, k_n), \end{cases}$$

где f_i — функция пересчета *i*-й координаты, аргументами являются координаты в системе k_i,. Можно поставить и обратную задачу: по известным координатам (m_1, m_2,.... m) определить координаты (κ_1, k_2,..., κ_n). Решение обратной задачи запишем так:

$$\begin{cases} k_1 = F_1(m_1, m_2, \ldots, m_N), \\ k_2 = F_2(m_1, m_2, \ldots, m_N), \\ \quad \ldots \\ k_n = F_n(m_1, m_2, \ldots, m_N), \end{cases}$$

где F_i — функции *обратного* преобразования.

В случае если размерности систем координат не совпадают ($n \neq N$), осуществить однозначное преобразование координат чаще всего не удается. Например, по двумерным экранным координатам нельзя без дополнительных условий однозначно определить трехмерные координаты отображаемых объектов.

Линейные преобразования наглядно записываются в матричной форме:

$$\begin{bmatrix} m_1 \\ m_2 \\ \ldots \\ \ldots \\ m_N \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} & a_{1n+1} \\ a_{21} & a_{22} & \ldots & a_{2n} & a_{2n+1} \\ \ldots & & & & \\ \ldots & & & & \\ a_{N1} & a_{N2} & \ldots & a_{Nn} & a_{Nn+1} \end{bmatrix} \begin{bmatrix} k_1 \\ k_2 \\ \ldots \\ \ldots \\ k_N \end{bmatrix}.$$

Здесь матрица коэффициентов (a_{ij}) умножается на матрицу-столбец (k_i), и в результате будем иметь матрицу-столбец (m_i).

Мы и дальше часто будем использовать умножение матриц, поэтому сделаем небольшой экскурс в матричную алгебру. Для двух матриц —

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}, \qquad B = \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1p} \\ b_{21} & b_{22} & \dots & b_{2p} \\ \dots & \dots & \dots & \dots \\ b_{n1} & b_{n2} & \dots & b_{np} \end{bmatrix}$$

матрицы A размерами $(m \times n)$ и B — $(n \times p)$:
матричным произведением является матрица $C = AB$ размерами $(m \times p)$:

$$C = \begin{bmatrix} c_{11} & c_{12} & \dots & c_{1p} \\ c_{21} & c_{22} & \dots & c_{2p} \\ \dots & \dots & \dots & \dots \\ c_{m1} & c_{m2} & \dots & c_{mp} \end{bmatrix},$$

для которой элементы c_{ij} рассчитываются по формуле

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}.$$

Простейшие двумерные преобразования

Точки на xy-плоскости можно *перенести* в новые позиции путем добавления к координатам этих точек констант переноса. Для каждой точки $P(x, y)$, которая перемещается в новую точку $P'(x, y)$, сдвигаясь на Dx единиц параллельно оси x и на Dy единиц параллельно оси y, можно написать уравнения:

$$x' = x + Dx, \quad y' = y + Dy.$$

На рис. 2.1 показана точка с координатами (1, 2), которая смещается на расстояние (5, 7), преобразуясь в точку (6, 9). Определяя векторы-строки

$$P = [x\,y], \quad P' = [x'\,y'], \quad T = [Dx\,Dy],$$

можно переписать это уравнение в векторной форме или более кратко

$$[x'\ y'] = [x\ y] + [Dx\ Dy],$$

$$P' = P + T.$$

Объект можно перенести, применяя вышевыведенное уравнения к *каждой* его точке. Однако, поскольку каждый отрезок, описывающий объект, состоит из бесконечного числа точек, такой процесс длился бы бесконечно долго. К счастью, все точки, принадлежащие отрезку, можно перенести путем перемещения одних лишь крайних точек отрезка и последующего вычерчивания нового отрезка между получившимися в результате точками. Это справедливо также для масштабирования (растяжения) и поворота. На рис. 2.1 показан результат действия на контур домика операции переноса на расстояние (3, -4).

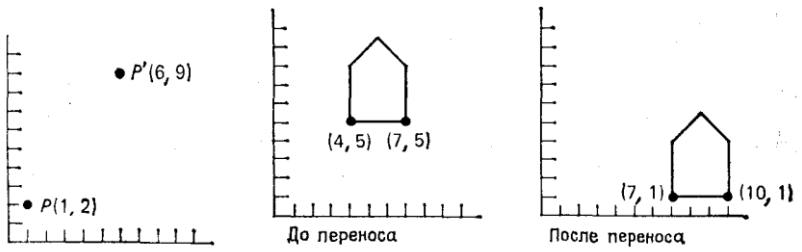


Рис. 2.1 Простейший перенос

Точки можно *промасштабировать* (растянуть) в *Sx* раз вдоль оси *x*: и в *Sy* раз вдоль оси *y,* получив в результате новые точки, с помощью умножения

$$x' = x \cdot Sx, \quad y' = y \cdot Sy.$$

Определяя S как $\begin{bmatrix} Sx & 0 \\ 0 & Sy \end{bmatrix}$, можно записать в матричной форме

$$[x'\ y'] = [x\ y] \begin{bmatrix} Sx & 0 \\ 0 & Sy \end{bmatrix}$$

или

$$P' = P \cdot S.$$

На рис. 2.2 отдельная точка (6, 6) масштабируется с коэффициентами 1/2 по оси *X* и 1/3 по оси *y.* На этом же рисунке показан контур домика, промасштабированный с коэффициентами 1/2 по оси *x* и 1/4 по оси *y.* Отметим, что масштабирование производится относительно начала координат; в результате преобразования домик стал меньше и ближе к началу координат. Если бы масштабные множители были больше 1, то домик увеличился бы и отдалился от начала координат. Способы проведения масштабирования относительно других точек, отличных от начала координат, рассматриваются в одном из последующих разделов главы. Пропорции домика также изменились· было применено *неоднородное* масштабирование, при котором $Sx \neq Sy$. *Однородное* масштабирование, для которого $Sx = Sy$, не влияет на пропорции.

Точки могут быть *повернуты* на угол θ относительно начала координат, как показано на рис. 2.2 для точки P (6, 1) и угла θ = 30°. Математически поворот определяется следующим образом:

$$x' = x \cdot \cos\theta - y \cdot \sin\theta,$$
$$y' = x \cdot \sin\theta + y \cdot \cos\theta.$$

В матричной форме мы имеем

$$[x'\ y'] = [x\ y] \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

или

$$P' = P \cdot R,$$

где через *R* обозначена матрица поворота. На рис. 2.2 показан квадрат, повернутый на 45°. Как и в случае масштабирования, поворот производится относительно начала координат.

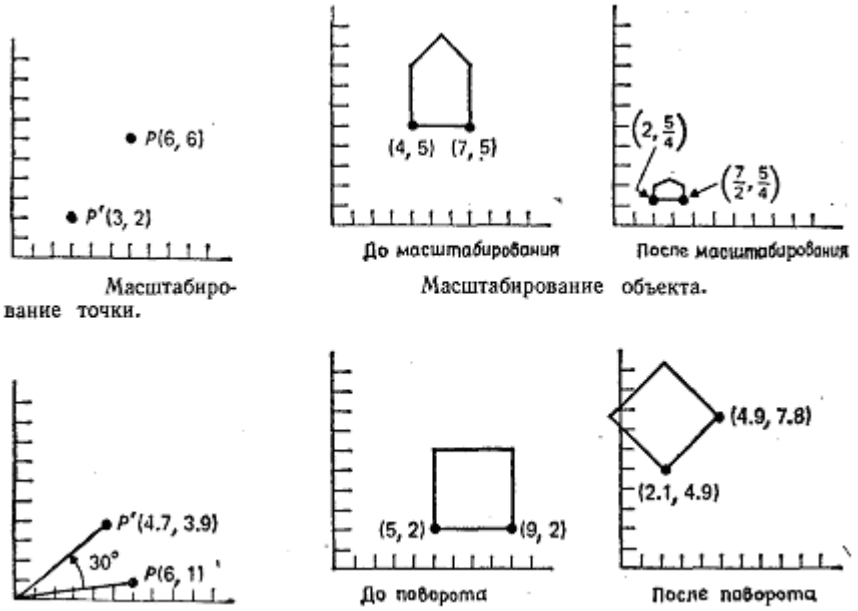


Рис. 2.2 Простейшие поворот и масштабирование

Однородные координаты и матричное представление двумерных преобразований

Преобразования переноса, масштабирования и поворота в матричной форме записываются в виде

$$P' = P + T,$$
$$P' = P \cdot S,$$
$$P' = P \cdot R.$$

К сожалению, перенос реализуется отдельно (с помощью сложения) от масштабирования и поворота (с помощью умножения). Хотелось бы представить их таким способом, чтобы все эти три элементарных преобразования можно было легко объединять вместе. Ниже в этом разделе показано, как это можно сделать.

Если мы выразим точки в *однородных координатах,* то все три преобразования можно реализовать с помощью умножений. Однородные координаты были введены в геометрии и впоследствии использованы в графике. С однородными координатами и преобразованиями над ними работают многие пакеты графических подпрограмм и некоторые дисплейные процессоры. В одних случаях эти координаты используются прикладной программой непосредственно при задании параметров для графического пакета, в других — применяются лишь внутри самого пакета и недоступны для программиста.

В однородных координатах точка $P\ (x, y)$ записывается как $P(W \cdot x, W \cdot y, W)$ для любого масштабного множителя $W \neq 0$. При этом если для точки задано ее представление в однородных координатах $P(X, Y, W)$, то можно найти ее двумерные декартовы координаты как $x=X/W$ и $y=Y/W$. В этой главе W всегда будет равно 1, поэтому операция деления не требуется. Однородные координаты можно представить как вложение промасштабированной с коэффициентом W двумерной плоскости в плоскость $z=W$ (здесь $z = 1$) в трехмерном пространстве.

Точки теперь описываются трехэлементными вектор-строками, поэтому матрицы преобразований, на которые умножается вектор точки, чтобы получить другой вектор точки, должны иметь размер 3x3. Уравнения переноса записываются в виде матрицы преобразования однородных координат следующим образом:

$$[x'\ y'\ 1] = [x\ y\ 1] \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ Dx & Dy & 1 \end{bmatrix}$$

$$P' = P \cdot T(Dx,\ Dy),$$

где

$$T(Dx,\ Dy) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ Dx & Dy & 1 \end{bmatrix}.$$

Что будет, если точку P перенести в точку P' на расстояние (Dx_1, Dy_1), а затем в P'' на расстояние $(Dx_2\ Dy_2)$. Интуитивно ожидаемый результат в этом случае представляет собой суммарный перенос на расстояние $(Dx_1+Dx_2,\ Dy_1+Dy_2)$. Чтобы доказать это, запишем данные в виде

$$P' = P \cdot T(Dx_1,\ Dy_1),$$
$$P'' = P' \cdot T(Dx_2,\ Dy_2).$$

Теперь получим:

$$P'' = (P \cdot T(Dx_1,\ Dy_1)) \cdot T(Dx_2,\ Dy_2) = P \cdot (T(Dx_1,\ Dy_1) \cdot T(Dx_2,\ Dy_2)).$$

Матричное произведение $T(Dx_1, Dy_1) \cdot T(Dx_2, Dy_2)$ есть

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ Dx_1 & Dy_1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ Dx_2 & Dy_2 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ Dx_1+Dx_2 & Dy_1+Dy_2 & 1 \end{bmatrix}.$$

Действительно, результирующий перенос есть $(Dx_1+Dx_2,\ Dy_1+\ Dy_2)$. Матричное произведение в разных случаях называют *объединением, соединением, конкатенацией* и *композицией* матриц $T(Dx_1,\ Dy_2)$ и $T(Dx_2,\ Dy_2)$. В этой главе мы будем использовать термин *композиция.*

Уравнения масштабирования в матричной форме записываются в виде

$$[x'\ y'\ 1] = [x\ y\ 1] \cdot \begin{bmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Определяя

$$S(Sx,\ Sy) = \begin{bmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

имеем

$$P' = P \cdot S(Sx,\ Sy).$$

Так же как последовательные переносы являются аддитивными, можно ожидать, что последовательные масштабирования будут мультипликативными. Если заданы

$$P' = P \cdot S(Sx_1,\ Sy_1),$$
$$P'' = P' \cdot S(Sx_2,\ Sy_2),$$

то получим:

$$P' = (P \cdot S(Sx_1,\ Sy_1)) \cdot S(Sx_2,\ Sy_2) = P \cdot (S(Sx_1,\ Sy_1) \cdot S(Sx_2,\ Sy_2)).$$

Матричное произведение $S(Sx_1, Sy_1) \cdot S(Sx_2, Sy_2)$ есть

$$\begin{bmatrix} Sx_1 & 0 & 0 \\ 0 & Sy_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} Sx_2 & 0 & 0 \\ 0 & Sy_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} Sx_1 \cdot Sx_2 & 0 & 0 \\ 0 & Sy_1 \cdot Sy_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Таким образом, масштабирования в самом деле мультипликативны. И, наконец, уравнения поворота можно представить в виде:

$$[x' \;\; y' \;\; 1] = [x \;\; y \;\; 1] \cdot \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Полагая

$$R(\theta) = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

Имеем

$$P' = P \cdot R(\theta).$$

Композиция двумерных преобразований

Понятие композиции было введено в предыдущем разделе. В данном разделе мы покажем, каким образом можно использовать композицию преобразований для объединения фундаментальных матриц *R, S и T* с целью получения желаемых общих результатов. Основное преимущество объединенных преобразований состоит в том, что к точке более эффективно применять одно результирующее преобразование, чем ряд преобразований друг за другом.

Рассмотрим, например, поворот объекта относительно некоторой произвольной точки *Pi.* Поскольку нам известно, лишь как поворачивать вокруг начала координат, разобьем исходную (трудную) проблему на три более легкие задачи. Таким образом, чтобы произвести поворот относительно точки *Pi,* необходимо выполнить последовательно три элементарных преобразования:

1. Перенос, при котором точка *Pi* перемещается в начало координат.
2. Поворот.
3. Перенос, при котором точка из начала координат возвращается в первоначальное положение *Pi.*

Эта последовательность показана на рис. 2.3, на котором вокруг точки *Pi(x, y)* поворачивается контур домика. Первый перенос производится на $(-x_1, -y_1)$, в то время как последующий — на (x_1, y_1) — является обратным ему. Результат существенно отличается от того, который получился бы, если бы применялся один только поворот.

Результирующее преобразование имеет вид

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -x_1 & -y_1 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x_1 & y_1 & 1 \end{bmatrix} =$$

$$= \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ x_1(1-\cos\theta) + y_1\sin\theta & y_1(1-\cos\theta) - x_1\sin\theta & 1 \end{bmatrix}.$$

Эта композиция преобразований путем умножения матриц служит примером того, как применение однородных координат упрощает задачу.

Используя аналогичный подход, можно промасштабировать объект относительно произвольной точки *Pi:* перенести *Pi* в начало координат, промасштабировать, перенести назад в точку *Pi.* Результирующее преобразование в этом случае будет иметь вид

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -x_1 & -y_1 & 1 \end{bmatrix} \cdot \begin{bmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x_1 & y_1 & 1 \end{bmatrix} =$$

$$= \begin{bmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ x_1(1-Sx) & y_1(1-Sy) & 1 \end{bmatrix}.$$

Предположим, что нам необходимо промасштабировать, повернуть и расположить в нужном месте домик, показанный на рис.2.3, где центром поворота и масштабирования является точка P_1. Последовательность преобразований заключается в переносе точки P_1 в начало координат, проведении масштабирования и поворота, а затем переносе из начала координат в новую позицию P_2, в которой домик должен оказаться (эта последовательность показана на рис. 2.3). В структуре данных, в которой содержится это преобразование, могут находиться масштабный множитель (множители), угол поворота и величины переноса или может быть записана матрица результирующего преобразования:

$$T(-x_1, -y_1) \cdot S(Sx, Sy) \cdot R(\theta) \cdot T(x_2, y_2).$$

Если известно, что M_1 и M_2 представляют собой элементарные перенос, масштабирование или поворот, то при каких условиях M_1 и M_2 коммутативны? В общем случае умножение матриц некоммутативно. Однако легко показать, что в следующих частных случаях коммутативность имеет место (в этих случаях можно не беспокоиться о порядке перемножения матриц – см. Таблицу 2.1).

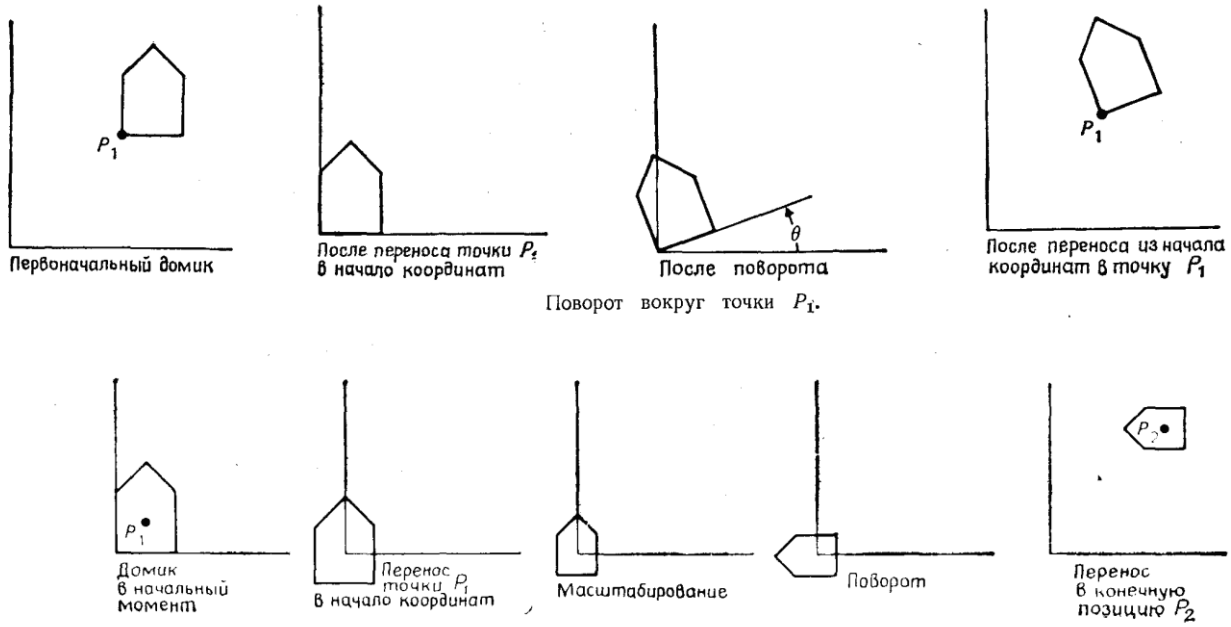


Рис. 2.3 Композиция преобразований

Таблица 2.1

M_1	M_2
Перенос	*Перенос*
Масштабирование	*Масштабирование*
Поворот	*Поворот*
Масштабирование (при Sx = Sy)	*Поворот*

Матричное представление трехмерных преобразований

Аналогично тому, как двумерные преобразования описываются матрицами размером 3x3, трехмерные преобразования могут быть представлены в виде матриц размером 4x4. И тогда трехмерная точка (x, *y, z)* записывается в однородных координатах как *(W·x, W·y, W·z, W),* где W≠0. Если *W≠1,* для получения трехмерных декартовых координат точки *(x, y, z)* первые три однородные координаты делятся на *W.* Отсюда, в частности, следует, что две точки H_1 и H_2 в пространстве однородных координат описывают одну и ту же точку трехмерного пространства в том и только в том случае, когда $H_1=cH_2$ для любой константы *c,* не равной нулю.

Трехмерная система координат, применяемая в этой книге, является *правосторонней* (рис.2.4). Примем соглашение, в соответствии с которым положительными будем считать такие повороты, при которых (если смотреть с конца положительной полуоси в направлении начала координат) *поворот на 90° против часовой стрелки* будет переводить одну положительную полуось в другую. На основе этого соглашения строится следующая таблица, которую можно использовать как для правых, так и для левых систем координат:

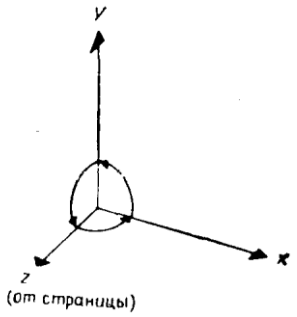


Рис. 2.4. Правосторонняя система координат

Таблица 2.2. Правосторонняя система координат

Если ось вращения	Положительным будет направление поворота
x	от y к z
y	от z к x
z	от x к y

Мы применяем здесь правостороннюю систему координат, поскольку она хорошо знакома большинству людей, хотя в трехмерной графике чаще более удобна левосторонняя система, так как ее легче представить наложенной на поверхность экрана дисплея. Это позволяет естественно интерпретировать тот факт, что точки с большими значениями z находятся дальше от наблюдателя. Отметим, что в левосторонней системе положительными будут повороты, *выполняемые по часовой стрелке,* если смотреть с конца положительной полуоси в направлении начала координат.

Трехмерный перенос является простым расширением двумерного:

$$T(Dx,\ Dy,\ Dz) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ Dx & Dy & Dz & 1 \end{bmatrix},$$

т. е. $[xyz1]\cdot T(Dx, Dy, Dz)=[x+Dx\ y+Dy\ z+Dz\ 1]$.

Масштабирование расширяется аналогичным образом:

$$S(Sx,\ Sy,\ Sz) = \begin{bmatrix} Sx & 0 & 0 & 0 \\ 0 & Sy & 0 & 0 \\ 0 & 0 & Sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

В самом деле, $[x\ y\ z\ 1]\cdot S(Sx, Sy, Sz)=[Sx\cdot x\ Sy\cdot y\ Sz\cdot z\ 1]$.

Двумерный поворот является в то же время трехмерным поворотом вокруг оси *z*. В трехмерном пространстве поворот вокруг оси *z* описывается выражением

$$Rz(\theta)=\begin{bmatrix}\cos\theta & \sin\theta & 0 & 0\\ -\sin\theta & \cos\theta & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}.$$

Это легко проверить: в результате поворота на 90° вектора [1 0 0 1], являющегося единичным вектором оси *x,* должен получиться единичный вектор [0 1 0 1] оси *y*. Вычисляя произведение

$$[1\ 0\ 0\ 1]\cdot\begin{bmatrix}0 & 1 & 0 & 0\\ -1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix},$$

получаем предсказанный результат [0 1 0 1]. Матрица поворота вокруг оси *x* имеет вид

$$Rx(\theta)=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & \cos\theta & \sin\theta & 0\\ 0 & -\sin\theta & \cos\theta & 0\\ 0 & 0 & 0 & 1\end{bmatrix}.$$

Матрица поворота вокруг оси *y* записывается в виде

$$Ry(\theta)=\begin{bmatrix}\cos\theta & 0 & -\sin\theta & 0\\ 0 & 1 & 0 & 0\\ \sin\theta & 0 & \cos\theta & 0\\ 0 & 0 & 0 & 1\end{bmatrix}.$$

Столбцы (и строки) верхней левой подматрицы размером 3x3 матриц *Rz, Rx* и *Ry* представляют собой взаимно ортогональные единичные векторы, интерпретация которых такая же, что и в двумерном случае.

Все эти матрицы преобразований имеют обратные матрицы. Матрица, обратная *T,* получается подстановкой знака минус перед *Dx, Dy* и *Dz;* обратная *S* — заменой *Sx, Sy* и *Sz* на обратные им значения, а для каждой из трех матриц поворота — выбором отрицательного угла поворота.

Композиция трехмерных преобразований

Путем объединения элементарных трехмерных преобразований можно получить другие преобразования. В этом разделе показано, как это сделать. Задача состоит в том, чтобы преобразовать отрезки P_1P_2 и P_1P_3 (рис. 2.5) из начальной позиции в конечную. Точка *Pi* переносится в начало координат, P_1P_2 располагается вдоль отрицательной полуоси *x*, а P_1P_3 помещается в плоскости *yz* в той ее половине, где ось *y* положительна. На длины отрезков преобразование не воздействует.

Как и прежде, разобьем сложную задачу на более простые. В данном случае преобразование можно выполнить за четыре шага:

1. Перенос точки *Pi* в начало координат.
2. Поворот вокруг оси *y* до совмещения P_1P_2 с плоскостью *yz.*
3. Поворот вокруг оси *x* до совмещения P_1P_2 с отрицательной полуосью *Z.*
4. Поворот вокруг оси *z* до совмещения P_1P_3 с плоскостью *yz.*

Шаг 1. Перенос P_1 в начало координат:

$$T(-x_1, -y_1, -z_1)=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ -x_1 & -y_1 & -z_1 & 1\end{bmatrix}.$$

Применение T к P_1, P_2 и P_3 дает

$$P_1' = P_1 \cdot T(-x_1, -y_1, -z_1) = [0\ 0\ 0\ 1],$$
$$P_2' = P_2 \cdot T(-x_1, -y_1, -z_1) = [x_2 - x_1\ \ y_2 - y_1\ \ z_2 - z_1\ \ 1],$$
$$P_3' = P_3 \cdot T(-x_1, -y_1, -z_1) = [x_3 - x_1\ \ y_3 - y_1\ \ z_3 - z_1\ \ 1].$$

Шаг 2. Поворот вокруг оси y. На рис. 2.5 показаны отрезки P_1P_2 после шага 1 и проекция P_1P_2 на плоскость xz. Поворот производится на положительный угол θ, для которого

$$\cos\theta = \frac{-z_2'}{D_1} = \frac{-(z_2 - z_1)}{D_1},$$
$$\sin\theta = \frac{x_2'}{D_1} = \frac{x_2 - x_1}{D_1},$$

где

$$D_1 = \sqrt{(z_2 - z_1)^2 + (x_2 - x_1)^2}.$$

Тогда

$$P_2'' = P_2' \cdot R_y(\theta) = \left[0\ \ y_2 - y_1 - \frac{(x_2 - x_1)^2}{D_1} - \frac{(z_2 - z_1)^2}{D_1}\ \ 1\right].$$

Как и ожидалось, x-компонента P_2 равна нулю.

Шаг 3. Поворот вокруг оси x. На рис. 2.6 показан отрезок P_1P_2

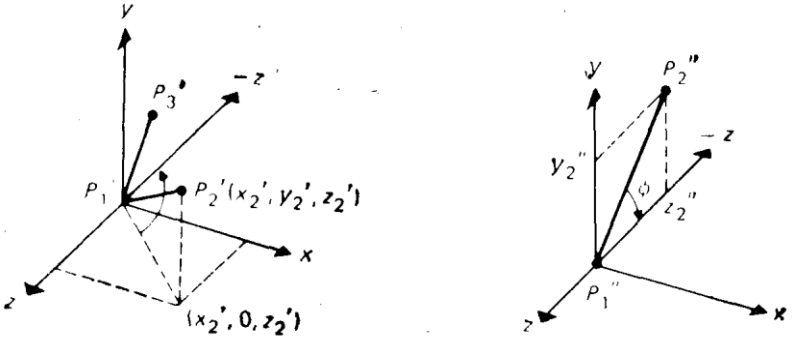


Рис. 2.5. Композиция преобразований

после шага 2. Поворот производится на отрицательный угол φ, для которого

$$\cos(-\varphi) = \cos\varphi = \frac{-z_2''}{\|\overline{P_1P_2}\|},$$
$$\sin(-\varphi) = -\sin\varphi = \frac{-y_2''}{\|\overline{P_1P_2}\|},$$

где

$$\|\overline{P_1P_2}\| = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}.$$

Запись $\|\overline{P_1P_2}\|$ обозначает длину $\overline{P_1P_2}$. Результатом поворота на шаге 3 является

$$P_2''' = P_2'' \cdot Rx(\varphi) = P_2' \cdot Ry(\theta) \cdot Rx(\varphi) =$$
$$= P_2 \cdot T \cdot Ry(\theta) \cdot Rx(\varphi) = [0\ \ 0\ \ -\|\overline{P_1P_2}\|\ \ 1],$$

т. е. $\overline{P_1P_2}$ теперь совпадает с отрицательной полуосью z.

Шаг 4. Поворот вокруг оси z. На рис. 2.6 показаны $\overline{P_1P_2}$ и $\overline{P_1P_3}$ после шага 3, когда P_2''' лежит на отрицательной полуоси z, а P_3''' - в точке

$$P_3''' = [x_3'''\ \ y_3'''\ \ z_3'''\ \ 1] = P_3 \cdot T(-x_1, -y_1, -z_1) \cdot Ry(\theta) \cdot Rx(\varphi).$$

Поворот производится на

$$\cos\alpha = y_3'''/D_2,\ \ \sin\alpha = x_3'''/D_2,\ \ D_2 = \sqrt{(x_3''')^2 + (y_3''')^2}.$$

положительный угол α, для которого Шаг 4 является последним шагом, после которого получается конечный результат, показанный на рис. 2.6. Результирующая матрица

$T(-x_1, -y_1, -z_1) \cdot Ry(\theta) \cdot Rx(\varphi) \cdot Rz(\alpha) = T \cdot R$

описывает искомое преобразование, где $R = Ry(\theta) \cdot Rx(\varphi) \cdot Rz(\alpha)$.

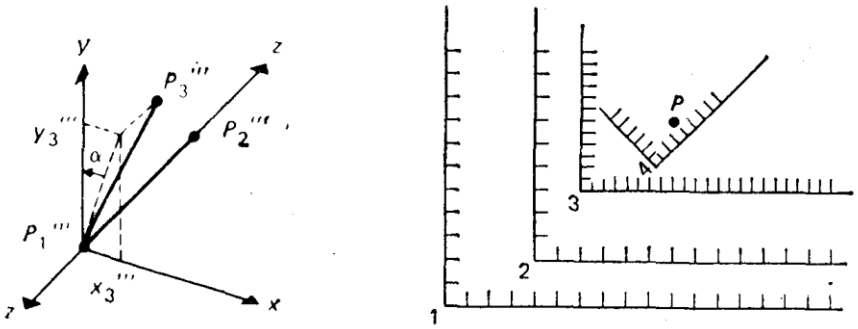


Рис. 2.6. Окончание композиции преобразований

Преобразование объектов

Преобразование объектов можно описать так. Пусть любая точка, принадлежащая определенному объекту, имеет координаты ($k_1, k_2,..., k_n$) в n-мерной системе координат. Тогда преобразование объекта можно определить как изменение положения точек объекта. Новое положение точки пространства соответствует новым значениям координат ($m_1, m_2,..., m_n$).

Соотношение между старыми и новыми координатами для всех точек объекта ($m_1, m_2,..., m_n$) = $F(k_1, k_2,..., k_n)$ и будет определять преобразование объекта, где F—функция преобразования.

Классифицировать преобразования объектов можно согласно типу функции преобразования и типу системы координат.

Например, преобразование объектов на плоскости можно определить так:

$$X = F_x(x, y),$$
$$Y = F_y(x, y).$$

В трехмерном пространстве:

$$X = F_x(x, y, z),$$
$$Y = F_y(x, y, z),$$
$$Z = F_z(x, y, z).$$

Преобразование как изменение систем координат

Мы рассматриваем преобразование множества точек, принадлежащих объекту, в некоторое другое множество точек, причем оба этих множества описаны в одной и той же системе координат. Таким образом, система координат остается неизменной, а сам объект преобразуется относительно начала координат до получения желаемого размера. Другим эквивалентным способом описания преобразования является смена систем координат. Такой подход оказывается полезным, когда желательно собрать вместе много объектов, каждый из которых описан в своей собственной локальной системе координат, и выразить их координаты в одной глобальной системе координат. Существует и еще один, третий подход, при котром происходит изменение глобальной системы координат по отношению к локальной системе координат объекта (см. рис.2.7).

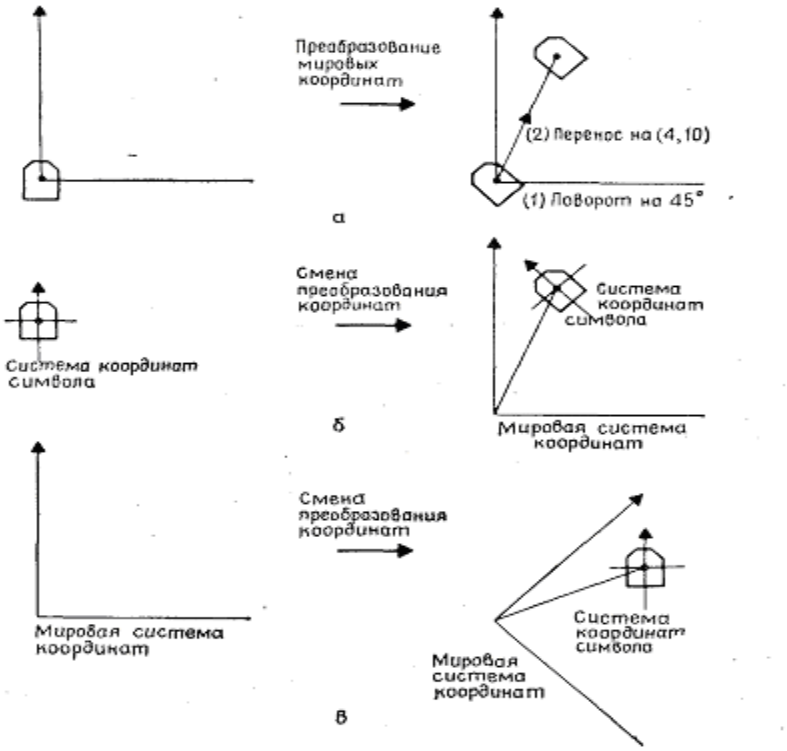


Рис. 2.7. Преобразования как изменение системы координат

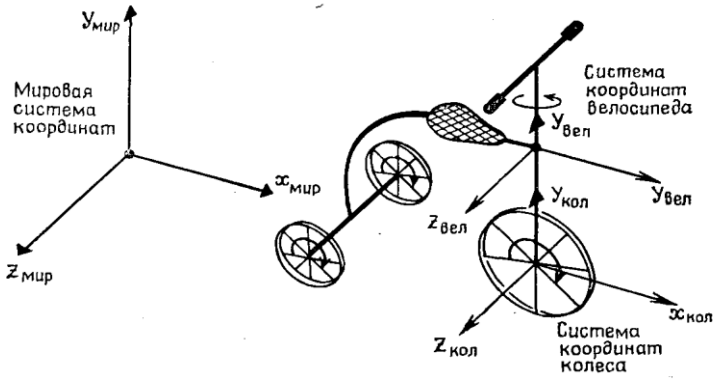


Рис. 2.8. Пример преобразования с изменением системы координат

Описание всех объектов (символов) в мировой системе координат и последующее размещение их в желаемом месте, приводит до некоторой степени к нереалистичному изображению всех символов, первоначально заданных один поверх другого в одной и той же мировой системе координат. Более естественно полагать, что каждый символ задан в своей собственной системе координат и затем промасштабирован, повернут и перенесен путем преобразования координат в новую мировую систему координат. Второй подход легко представить себе как сжатие или растяжение, поворот и позиционирование на мировой координатной плоскости отдельных листков бумаги, на каждом из которых изображен символ (или наоборот, сжатие или растяжение, поворот и перемещение плоскости относительно каждого из листков бумаги). С математической точки зрения оба подхода идентичны.

Подход, основанный на изменении систем координат, удобен в тех случаях, когда задается дополнительная информация для подобъектов в их локальных системах координат. Например, если к переднему колесу трехколесного велосипеда (рис. 2.8) приложить крутящий момент, то все его колеса повернутся. Нам необходимо определить, насколько велосипед переместится в пространстве как единое целое. Эта задача более сложная, чем та, которая связана с размещением символов, поскольку здесь требуется несколько последовательных изменений систем координат. В начальный момент системы координат велосипеда и его переднего колеса заданы относительно мировой системы

координат. При движении велосипеда вперед переднее колесо поворачивается вокруг оси *z* системы координат колеса, и одновременно системы координат колеса и велосипеда перемещаются относительно мировой системы координат. Системы координат колеса и велосипеда связаны с мировой системой координат с помощью зависящих от времени переносов вдоль осей *x* и *y* и поворота вокруг оси *y.* Координатные системы велосипеда и колеса между собой связаны с помощью зависящего от времени поворота вокруг оси *y,* вызываемого поворотом руля. (Система координат велосипеда связана с рамой велосипеда, а не с рулем).

2.1.2 Аффинные преобразования на плоскости

Это частный случай преобразований, который достаточно часто используется при создании графических пакетов.

Зададим некоторую двумерную систему координат (*x,y).* Аффинное преобразование на плоскости описывается формулами

$$\begin{cases} X = Ax + By + C, \\ Y = Dx + Ey + F, \end{cases}$$

где *A, B,..., F*— константы. Значение *(X, Y)* можно рассматривать как координаты в новой системе координат.

Обратное преобразование (*X, Y)* в (*x, y)* также является аффинным:

$$\begin{cases} x = A'X + B'Y + C', \\ y = D'X + E'Y + F'. \end{cases}$$

Аффинное преобразование удобно записывать в матричном виде. Константы *A, B..... F* образуют матрицу преобразования, которая, будучи умноженной на матрицу-столбец координат (x, *y),* дает матрицу-столбец (*X, Y).* Однако, чтобы учесть константы C и F, необходимо перейти к так называемым *однородным координатам* — прибавим еще одну строку в матрицах координат:

$$\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = \begin{bmatrix} A & B & C \\ D & E & F \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}.$$

Теперь рассмотрим частные случаи аффинного преобразования.

1. Параллельный сдвиг координат (рис. 2. 9).

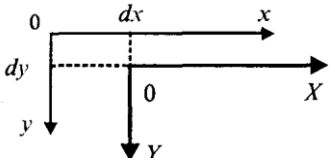


Рис. 2.9. Параллельный сдвиг координат

$$\begin{cases} X = x - dx, \\ Y = y - dy. \end{cases}$$

В матричной форме: $\begin{bmatrix} 1 & 0 & -dx \\ 0 & 1 & -dy \\ 0 & 0 & 1 \end{bmatrix}$.

Обратное преобразование: $\begin{cases} x = X + dx, \\ y = Y + dy, \end{cases}$

$$\begin{bmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{bmatrix}.$$

2. Растяжение-сжатие осей координат (рис. 2. 10).

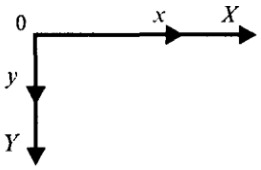


Рис. 2.10. Растяжение-сжатие осей координат

$$\begin{cases} X = x/k_x, \\ Y = y/k_y, \end{cases}$$

$$\begin{bmatrix} 1/k_x & 0 & 0 \\ 0 & 1/k_y & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Обратное преобразование: $\begin{cases} x = X\,k_x, \\ y = Y\,k_y, \end{cases}$ $\begin{bmatrix} k_x & 0 & 0 \\ 0 & k_y & 0 \\ 0 & 0 & 1 \end{bmatrix}.$

Коэффициенты k_x и k_y могут быть отрицательными. Например, k_x = -1 соответствует зеркальному отражению относительно оси y.

3. Поворот (рис. 2. 11).

Рис.2.11. Поворот

$$\begin{cases} X = x\cos\alpha + y\sin\alpha, \\ Y = -x\sin\alpha + y\cos\alpha, \end{cases} \qquad \begin{bmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Обратное преобразование соответствует повороту системы *(X, Y)* на угол (-α).

$$\begin{cases} x = X\cos\alpha - Y\sin\alpha, \\ y = X\sin\alpha + Y\cos\alpha, \end{cases}$$

$$\begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Свойства аффинного преобразования.

• Любое аффинное преобразование может быть представлено как последовательность операций из числа указанных простейших: сдвиг, растяжение/сжатие и поворот.

• Сохраняются прямизна линии, параллельность прямых, отношение длин отрезков, лежащих на одной прямой, и соотношение площадей фигур.

Трехмерное аффинное преобразование

Запишем в виде формулы:

$X = Ax + By + Cz + D,$
$Y = Ex + Fy + Gz + H,$
$Z = Kx + Ly + Mz + N,$ где *A, B,..., N*— константы.

Дадим также запись в матричной форме: $\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} A & B & C & D \\ E & F & G & H \\ K & L & M & N \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}.$

Для трехмерного пространства любое аффинное преобразование также может быть представлено последовательностью простейших операций. Рассмотрим их.

1. Сдвиг осей координат соответственно на *dx, dy, dz:*

$$\begin{cases} X = x - dx, \\ Y = y - dy, \\ Z = z - dz, \end{cases}$$

$$\begin{bmatrix} 1 & 0 & 0 & -dx \\ 0 & 1 & 0 & -dy \\ 0 & 0 & 1 & -dz \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

2. Растяжение/сжатие на κ_x, κ_y, κ_z.

$$\begin{cases} X = x / k_x, \\ Y = y / k_y, \\ Z = z / k_z, \end{cases} \qquad \begin{bmatrix} 1/k_x & 0 & 0 & 0 \\ 0 & 1/k_y & 0 & 0 \\ 0 & 0 & 1/k_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

3. Повороты. Можно сказать, что в трехмерном пространстве существует больше разновидностей поворота, сравнительно с двумерным пространством. Рассмотрим несколько частных случаев поворота.

Поворот вокруг оси x на угол φ (рис. 2. 12).

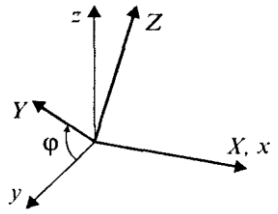
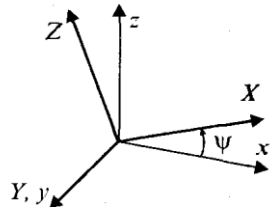


Рис. 2.12. Поворот вокруг оси X

$$\begin{cases} X = x, \\ Y = y\cos\varphi + z\sin\varphi, \\ Z = -y\sin\varphi + z\cos\varphi, \end{cases}$$

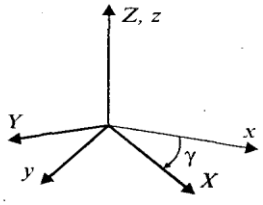
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\varphi & \sin\varphi & 0 \\ 0 & -\sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Поворот вокруг оси *y* на угол ψ (рис. 2. 13, сверху).



$$\begin{cases} X = x\cos\psi + z\sin\psi, \\ Y = y, \\ Z = -x\sin\psi + z\cos\psi, \end{cases}$$

$$\begin{bmatrix} \cos\psi & 0 & \sin\psi & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\psi & 0 & \cos\psi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$



Поворот вокруг оси z на угол γ (рис. 2. 13, снизу).

$$\begin{cases} X = x\cos\gamma + y\sin\gamma, \\ Y = -x\sin\gamma + y\cos\gamma, \\ Z = z, \end{cases}$$

$$\begin{bmatrix} \cos\gamma & \sin\gamma & 0 & 0 \\ -\sin\gamma & \cos\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Рис. 2.13. Поворот вокруг осей y и z

2.2 Проекции

При использовании любых графических устройств обычно используют проекции. Проекция задает способ отображения объектов на графическом устройстве. Мы будем рассматривать только проекции на плоскость.

Проецирование - отображение точек, заданных в системе координат с размерностью N, в точки в системе меньшей размерности.

Проекторы (проецирующие лучи) - отрезки прямых, идущие из центра проекции через каждую точку объекта до пересечения с плоскостью проекции (картинной плоскостью).

2.2.1 Мировые и экранные координаты

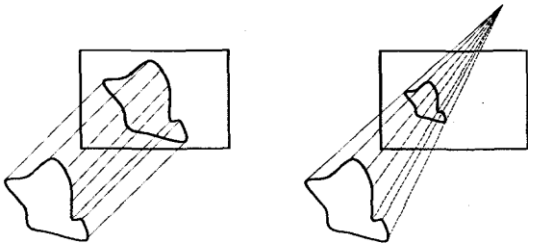
При отображении пространственных объектов на экране или на листе бумаги с помощью принтера необходимо знать координаты объектов. Мы рассмотрим две системы координат. Первая — *мировые координаты,* которые описывают истинное положение объектов в пространстве с заданной точностью. Вторая — система координат устройства отображения, в котором осуществляется вывод изображения объектов в заданной проекции. Назовем систему координат графического устройства *экранными координатами* (хотя это устройство и не обязательно должно быть подобно монитору компьютера).

Пусть мировые координаты будут трехмерными прямоугольными координатами. Где должен размещаться центр координат, и какими будут единицы измерения вдоль каждой оси, для нас сейчас не очень важно. Важно то, что для отображения мы будем знать любые числовые значения координат отображаемых объектов.

Для получения изображения в определенной проекции необходимо вычислить координаты проекции. Для синтеза изображения на плоскости экрана или бумаге используем двумерную систему координат. Основная задача — задать преобразования координат из мировых в экранные.

2.2.2 Основные типы проекций

Изображение объектов на плоскости (экране дисплея) связано с геометрической операцией проектированием. В компьютерной графике используется несколько видов проектирования, но основных - два вида: параллельное и центральное.



Проектирующий пучок лучей направляется через объект на картинную плоскость, на которую в дальнейшем находят координаты пересечения лучей (или прямых) с этой плоскостью.

Рис. 2.14. Основные типы проекций

При центральном проектировании все прямые исходят из одной точки.

При параллельном - считается, что центр лучей (прямых) бесконечно удален, а прямые параллельны.

Каждый из этих основных классов разбивается еще на несколько подклассов в зависимости от взаимного расположения картинной плоскости и координатных осей.

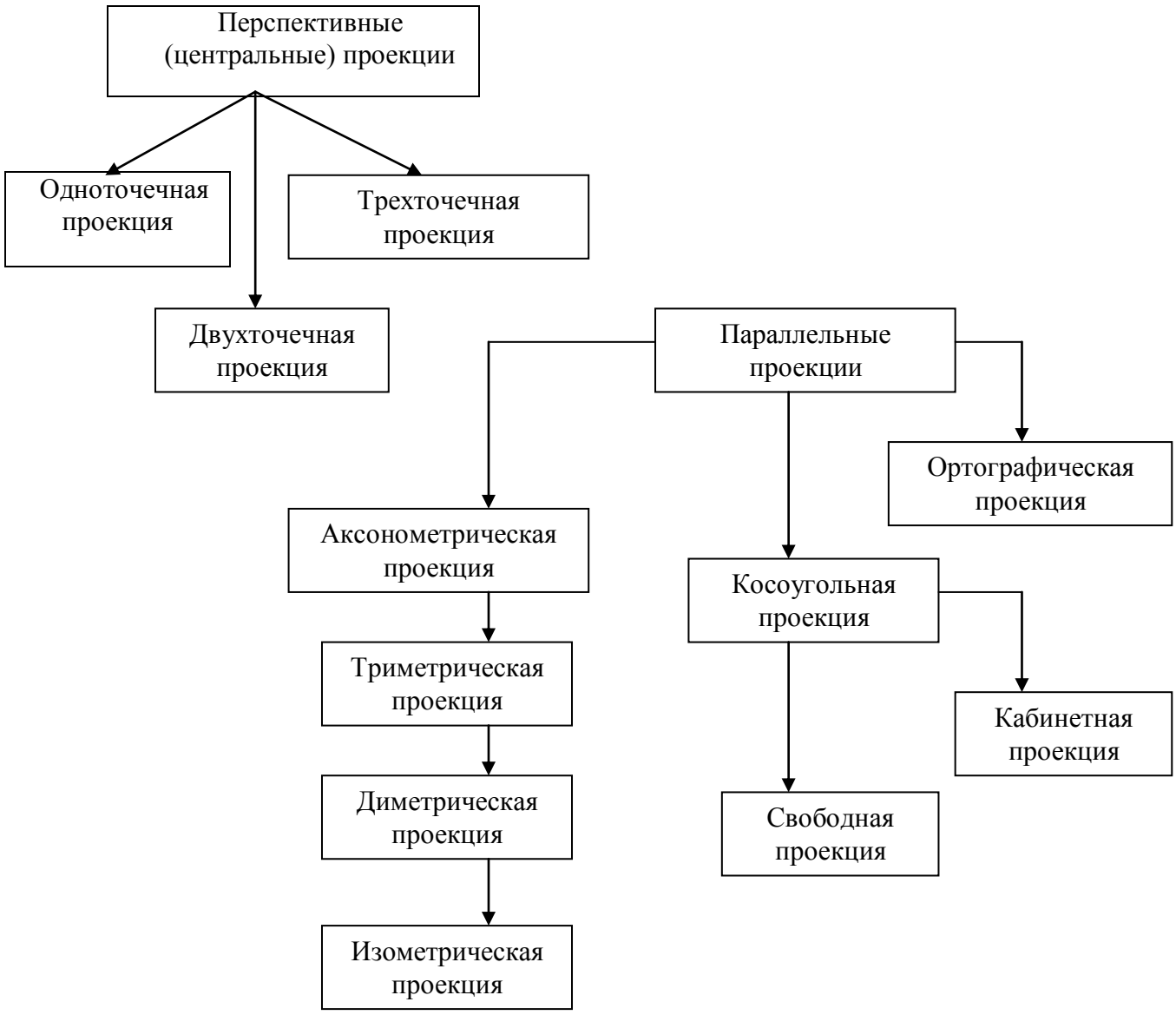


Рис. 2.15. Классификация плоских проекций

У параллельных проекций центр проекции расположен в бесконечности от плоскости проекции:

- ортографические (ортогональные),
- аксонометрические (прямоугольные аксонометрические) - проекторы перпендикулярны к плоскости проекции, расположенной под углом к главной оси,
- косоугольные (косоугольные аксонометрические) - плоскость проекции перпендикулярна к главной оси, проекторы расположены под углом к плоскости проекции.

У центральных проекций центр проекции находится на конечном расстоянии от плоскости проекции. Имеют место так называемые перспективные искажения.

Ортогональные проекции (основные виды)

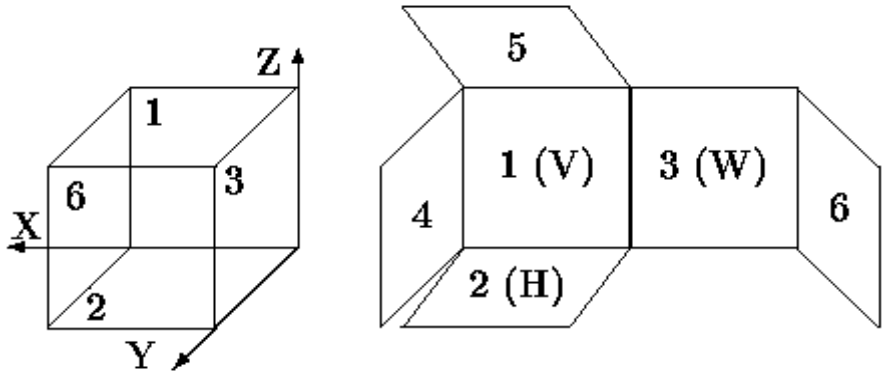


Рис. 2.16. Ортогональные проекции

1. Вид спереди, главный вид, фронтальная проекция, (на заднюю грань V),
2. Вид сверху, план, горизонтальная проекция, (на нижнюю грань H),
3. Вид слева, профильная проекция, (на правую грань W),
4. Вид справа (на левую грань),
5. Вид снизу (на верхнюю грань),
6. Вид сзади (на переднюю грань).

Матрица ортогональной проекции на плоскость YZ вдоль оси X имеет вид:

$$P_X = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Если же плоскость параллельна, то эту матрицу надо умножить на матрицу сдвига, тогда:

$$P_X = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ p & 0 & 0 & 1 \end{bmatrix}, \text{ где p - сдвиг по оси X;}$$

Для плоскости ZX вдоль оси Y

$$P_X = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & q & 0 & 1 \end{bmatrix}, \text{ где q - сдвиг по оси Y;}$$

Для плоскости XY вдоль оси Z:

$$P_X = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & R & 1 \end{bmatrix},$$ где R - сдвиг по оси Z.

При аксонометрической проекции проектирующие прямые перпендикулярны плоскости картинки.

Изометрия - все три угла между нормалью картинки и координатными осями равны.

Диметрия - два угла между нормалью картинки и координатными осями равны.

Триметрия - нормальный вектор плоскости картинки образует с координатными осями различные углы.

Каждый из трех видов этих проекций получается комбинацией поворотов, за которой следует параллельное проектирование.

При повороте на угол β относительно оси У (ординат), на угол α вокруг оси X (абсцисс) и последующем проектировании оси Z (аппликат) возникает матрица

$$M = \begin{vmatrix} cos\beta & \sin\alpha\cos\alpha & 0 & 0 \\ 0 & \cos\alpha & 0 & 0 \\ \sin\beta & -\sin\alpha\cos\beta & 0 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \Leftarrow \begin{vmatrix} \cos\beta & 0 & -\sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \bullet \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha & 0 \\ 0 & -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 0 \end{vmatrix} \bullet \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

Изометрическая проекция

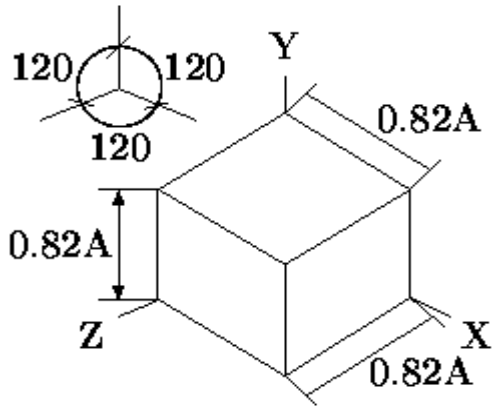


Рис. 2.17. Изометрические проекции

Диметрическая проекция

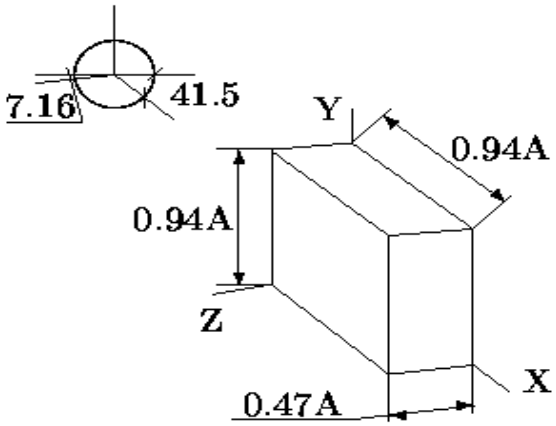


Рис. 2.18. Диметрические проекции

Косоугольные проекции

Классический пример параллельной косоугольной проекции — *кабинетная проекция* (рис. 2. 26). Эта проекция часто используется в математической литературе для черчения объемных форм. Ось *y* изображается наклоненной под углом 45 градусов. Вдоль оси *y* масштаб 0. 5, вдоль других осей — масштаб 1. Запишем формулы вычисления координат плоскости проецирования

$$X_{ПР} = x - 0.5y,$$
$$Y_{ПР} = 0.5y - z.$$

Здесь, как и раньше, ось $Y_{пр}$ направлена вниз.

Для косоугольных параллельных проекций лучи проецирования не перпендикулярны плоскости проецирования.

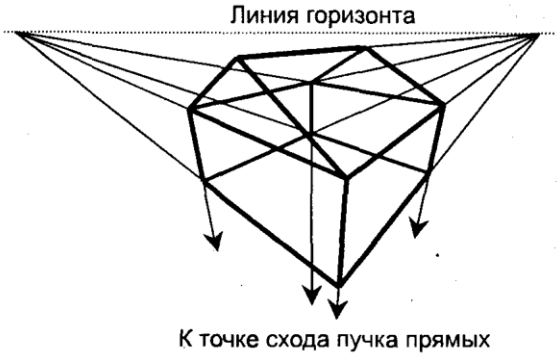


Рис. 2.19. Косоугольные проекции

Теперь относительно центральной проекции. Поскольку для нее лучи проецирования не параллельны, то будем считать *нормальной* такую *центральную проекцию,* главная ось которой перпендикулярна плоскости проецирования. Для *центральной косоугольной проекции* главная ось не перпендикулярна плоскости проецирования.

Рассмотрим пример центральной косоугольной проекции, которая показывает параллельными линиями все вертикальные линии изображаемых объектов. Расположим плоскость проецирования вертикально, ракурс показа зададим углами a, β и положением точки схода (рис. 2. 21).

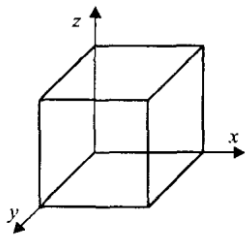


Рис.2.20. Кабинетная проекция

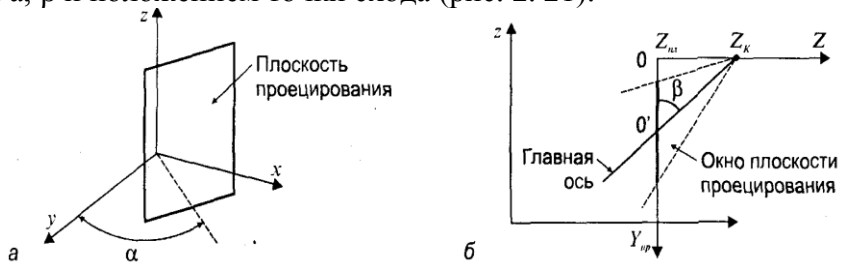


Рис. 2.21. Вертикальная центральная косоугольная проекция: а – расположение плоскости проецирования, б – вид с левого торца плокости проецирования

Будем считать, что ось *Z* видовых координат располагается перпендикулярно плоскости проецирования. Центр видовых координат — в точке (*xc*, *yc, zc)*. Запишем соответствующее видовое преобразование:

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} \text{Поворот} \\ \text{на угол } -90 \end{bmatrix} \begin{bmatrix} \text{Поворот} \\ \text{на угол } -\alpha \end{bmatrix} \begin{bmatrix} \text{Сдвиг} \\ \text{координат} \\ \text{на} \\ xc,\ yc,\ zc \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}.$$

Как и для нормальной центральной проекции, точка схода лучей проецирования располагается на оси Z на расстоянии Z_k от центра видовых координат. Необходимо учесть наклон главной оси косоугольной проекции. Для этого достаточно отнять от Y_{np} длину отрезка 0-0' (рис. 2.21). Эта длина равняется $(Z_k - Z_{пл})\ ctg\beta$. Теперь запишем результат — формулы вычисления координат косоугольной вертикальной проекции

$$X_{ПР} = \Pi x\ (X, Z),$$
$$Y_{ПР} = \Pi y\ (Y, Z) - (Z_к - Z_{пл})\ \text{ctg}\beta,$$

где *Пх* и *Пу* — это функции проецирования для нормальной проекции.

Следует отметить, что для такой проекции нельзя сделать вид сверху (β = 0), поскольку здесь $ctgP = \infty$.

Свойство рассмотренной вертикальной косоугольной проекции, заключающееся в сохранении параллельности вертикальных линий, иногда полезно, например, при изображении домов в архитектурных компьютерных системах. Сравните рис. 2. 22 (верх) и рис. 2.22 (низ). На нижнем рисунке вертикали изображаются вертикалями — дома не "разваливаются".

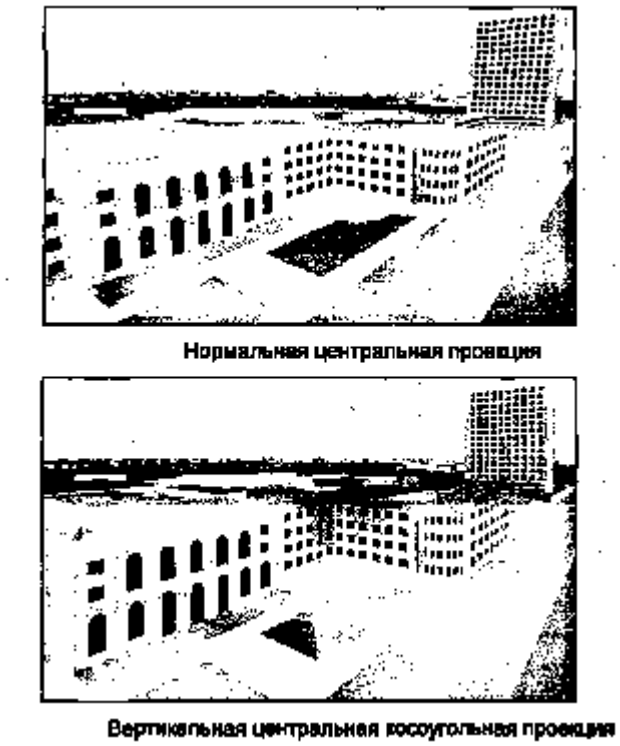


Рис. 2.21. Сравнение проекций

Кабинетная проекция (аксонометрическая косоугольная фронтальная диметрическая проекция)

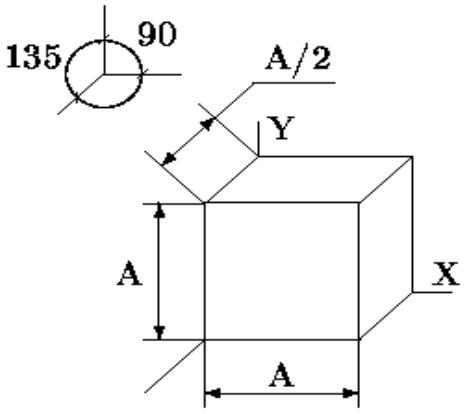


Рис. 2.23.Кабинетная проекция

Свободная проекция (аксонометрическая косоугольная горизонтальная изометрическая проекция)

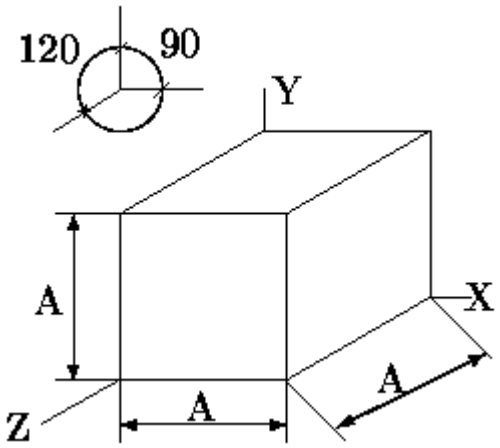


Рис. 2.24.Свободная проекция

Центральная проекция

Центральные проекции параллельных прямых, не параллельных плоскости проекции, сходятся в точке схода.

В зависимости от числа координатных осей, которые пересекает плоскость проекции, различаются одно, двух и трехточечные центральные проекции.

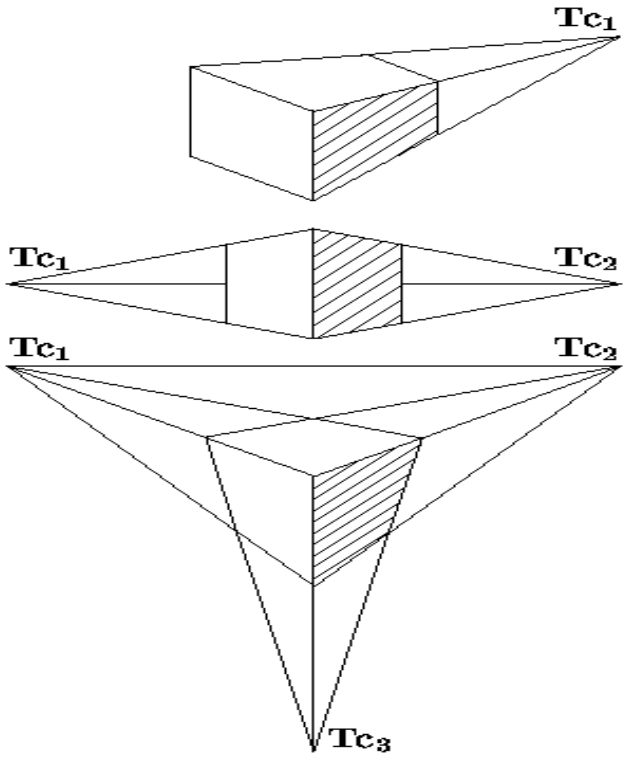


Рис. 2.25. Центральная проекция

Рассмотрим пример перспективной (центрально) проекции для вертикального расположения камеры, когда $\alpha = \beta = 0$. Такую проекцию можно себе представить как изображение на стекле, через которое смотрит наблюдатель, расположенный сверху в точке $(x, y, z) = (0, 0, z_k)$. Здесь плоскость проецирования параллельна плоскости *(x 0 y),* как показано на рис. 2. 26.

Для произвольной точки пространства (Р), исходя из подобия треугольников, запишем такие пропорции:

$$X_{пр}/(z_k - z_{пл}) = x/(z_k - z)$$
$$Y_{пр}/(z_k - z_{пл}) = y/(z_k - z)$$

Найдем координаты проекции, учитывая также координату *Znp:*

$$\begin{cases} X_{ПР} = x\,(z_к - z_{пл}) / (z_к - z), \\ Y_{ПР} = y\,(z_к - z_{пл}) / (z_к - z), \\ Z_{ПР} = z - z_{пл}. \end{cases}$$

Запишем такие преобразования координат в функциональном виде

$$\begin{cases} X_{ПР} = \boldsymbol{П}x\,(x, z), \\ Y_{ПР} = \boldsymbol{П}y\,(y, z), \\ Z_{ПР} = \boldsymbol{П}z\,(z), \end{cases}$$

где *П* — функция перспективного преобразования координат.

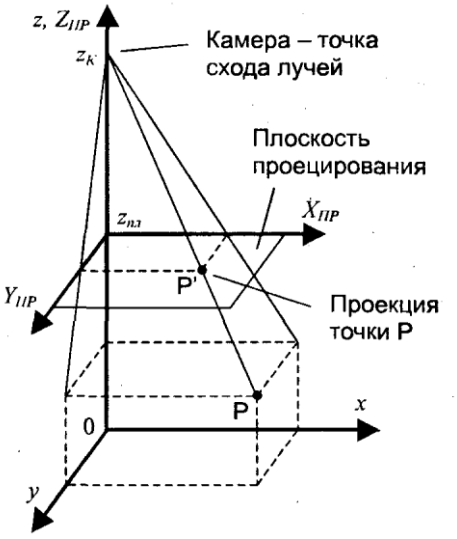


Рис. 2.26.Перпективная проекция

В матричной форме преобразования координат можно записать так:

$$\begin{bmatrix} X_{ПР} \\ Y_{ПР} \\ Z_{ПР} \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{z_к - z_{пл}}{z_к - z} & 0 & 0 & 0 \\ 0 & \frac{z_к - z_{пл}}{z_к - z} & 0 & 0 \\ 0 & 0 & 1 & -z_{пл} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}.$$

Обратите внимание на то, что здесь коэффициенты матрицы зависят от координаты z (в знаменателе дроби). Это означает, что преобразование координат — нелинейное (а точнее, *дробно-линейное),* оно относится к классу *проективных* преобразований.

Мы получили формулы вычисления координат проекции для случая, когда точка схода лучей находится на оси *z*. Теперь рассмотрим общий случай. Введем видовую систему координат *{X, Y,* Z), произвольно расположенную в трехмерном пространстве *(x, y, z*). Пусть точка схода находится на оси *Z* видовой системы координат, а направление обзора — вдоль оси *Z* противоположно ее направлению. Будем считать, что преобразование в видовые координаты описывается трехмерным аффинным преобразованием

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} \text{Матрица} \\ \text{видового} \\ \text{аффинного} \\ \text{преобразования} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}.$$

После вычисления координат (*X, Y, Z)* можно вычислить координаты в плоскости проецирования в соответствии с формулами, уже рассмотренными нами ранее. Поскольку точка схода находится на оси Z видовых координат, то

$$X_{ПР} = \Pi x\,(X, Z),$$
$$Y_{ПР} = \Pi y\,(Y, Z),$$
$$Z_{ПР} = \Pi z\,(Z).$$

Последовательность преобразования координат можно описать так:

$$(x\ y\ z) \xrightarrow{\text{Аффинное преобразование}} (X\ Y\ Z) \xrightarrow{\text{Перспектива } П(X, Y, Z)} (X_{ПР}\ Y_{ПР}\ Z_{ПР})$$

Такое преобразование координат позволяет моделировать расположения камеры в любой точке пространства и отображать в центре плоскости проецирования любые объекты обзора.

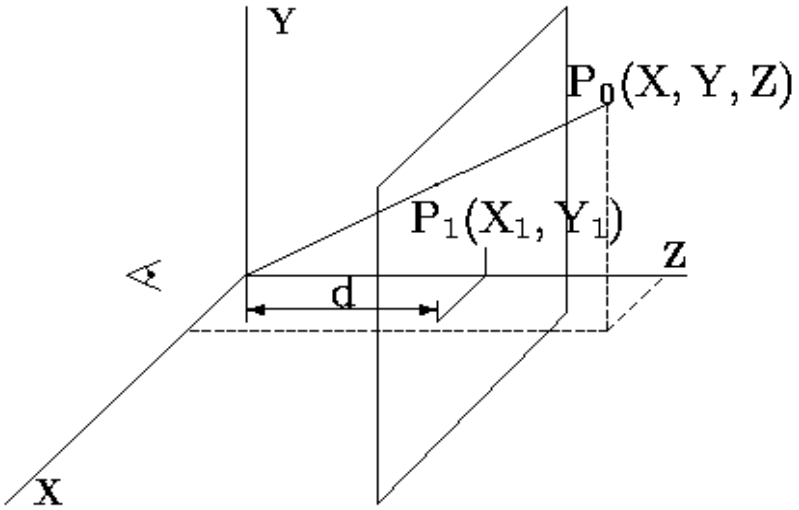


Рис. 2.27. Центральная проекция точки P_0 в плоскость Z = d

Глава 3. Растровая графика. Базовые растровые алгоритмы
3.1 Растровые изображения и их основные характеристики

Растр — это матрица ячеек (пикселов). Любой пиксел (*pixel — Picture Element)* имеет свой цвет. Совокупность пикселов различного цвета образует изображение. В зависимости от расположения пикселов в пространстве различают квадратный, прямоугольный, гексагональный или иные типы растра. Для описания расположения пикселов используют разнообразные системы координат. Общим для всех таких систем является то, что координаты пикселов образуют дискретный ряд значений (необязательно целые числа). Часто используется система целых координат — номеров пикселов с (0, 0) в левом верхнем углу. Такую систему мы будем использовать и в дальнейшем, ибо она удобна для рассмотрения алгоритмов графического вывода.

Какие основные характеристики растровых изображений?

Геометрические характеристики растра

Размер растра обычно измеряется количеством пикселов по горизонтали и вертикали.

Разрешающая способность. Она характеризует расстояние между соседними пикселами — шаг дискретной сетки растра. Разрешающую способность измеряют количеством пикселов на единицу длины. Наиболее популярная единица измерения — **dpi** *(dots per inch)* — количество пикселов в одном дюйме длины (2.54 см). Не следует отождествлять шаг с размерами пикселов — размер пикселов может равняться шагу, а может быть как меньше, так и больше шага.

Можно сказать, что для КГ наиболее удобен растр с одинаковым шагом для обеих осей, то есть dpi X= dpi Y. Это удобно для многих алгоритмов вывода графических объектов. Иначе — проблемы, например, при рисовании окружности на экране дисплея EGA (устаревшая модель компьютерной видеосистемы, ее растр — прямоугольный, пикселы растянуты по высоте, поэтому для изображения окружности необходимо генерировать эллипс).

Форма пикселов растра определяется особенностями устройства графического вывода (рис. 3.2). Например, пикселы могут иметь форму прямоугольника или квадрата, которые по размерам равны шагу растра (дисплей на жидких кристаллах); пикселы могут иметь круглую форму и по размерам могут не равняться шагу растра (принтеры).

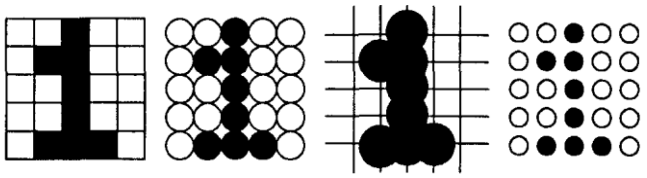


Рис. 3.1. Одно изображение на разных растрах

Количество цветов (глубина цвета) — важная характеристика любого изображения, не только растрового. В соответствии с психофизиологическими исследованиями, глаз человека способен различать 350 000 цветов.

Классифицируем изображения следующим образом.

- Двухцветные (бинарные) — 1 бит на пиксел. Среди двухцветных наиболее часто встречаются черно-белые изображения.
- Полутоновые — градации серого или другого цвета. Например, 256 градаций (1 байт на пиксел).
- Цветные изображения (2 бита на пиксел и больше). Глубина цвета 16 битов на пиксел (65536 цветов) получила название ***High Color,*** 24 бита на пиксел (16. 7 млн. цветов) — ***True Color.*** В компьютерных графических системах используют и большую глубину цвета — 32, 48 и более битов на пиксел.

В качестве примера рассмотрим растровый рисунок (рис. 3.1).

Количество цветов — 256 градаций серого, разрешающая способность — примерно 100 dpi. Отметим, что в книге вы видите черно-белый типографский оттиск, поэтому о количестве цветов и разрешающей способности можно говорить лишь условно.

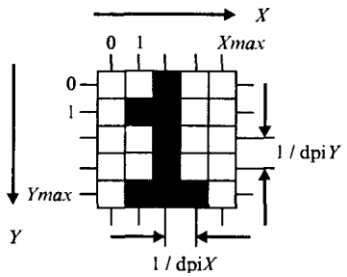


Рис. 3.2. Растр

Недостаточное количество цветов приводит к появлению лишних контуров на гладких поверхностях цилиндра и шара.

Оценка разрешающей способности растра

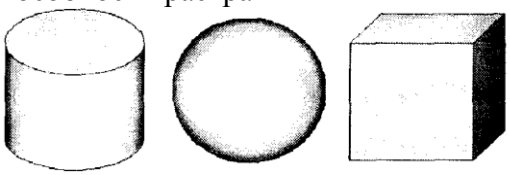


Рис. 3.3. 256 градаций серого, разрешающая способность 100 dpi

Изображение одних и тех же объектов, но дня других параметров растра (рис. 3.4).

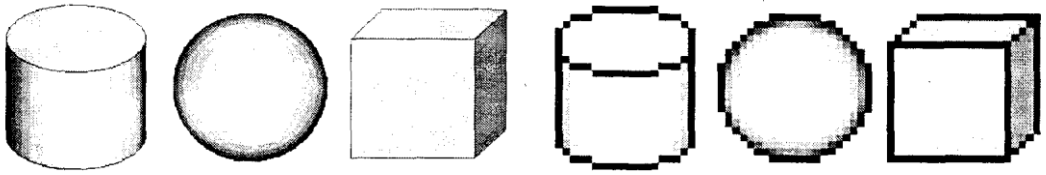


Рис. 3.4. Количество градаций серого составляет 8 Количество цветов сохранено (256 градаций), а разрешающая способность уменьшена в 8 раз

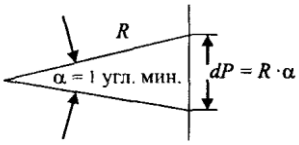


Рис. 3.5. Оценка разрешающей способности растра

Таблица 3.1. Разрешающая способность в зависимости от расстояния

Расстояние *R*, мм	Размер *dP*, мм	Разрешающая способность dpi
500	0. 14	181
300	0. 09	282

Глаз человека с нормальным зрением способен различать объекты с угловым размером около одной минуты. Если расстояние до объекта равно *R,* то можно приблизительно оценить этот размер *(dP),* как длину дуги, равную $R \cdot \alpha$ (рис. 3.5). Можно предположить, что человек различает дискретность растра (шаг) также соответственно этому минимально различимому размеру. Иначе говоря, если расстояние между отдельными тачками (пикселами) меньше чем *dP,* то эти точки уже не воспринимаются как отдельные точки. Тогда можно оценить минимальную разрешающую способность растрового изображения, которое человеком уже не воспринимается как растровое, следующей величиной; dpi = 25, 4 / *dP* [мм].

Приведем несколько значений dpi для разных *R* (табл. 3. 1).

Если считать расстояние, с которого человек обычно разглядывает печатные документы, равным 300 мм, то можно оценить минимальную разрешающую способность,

при которой уже не заметны отдельные пикселы, как примерно 300 dpi (приблизительно 0.085 мм). Лазерные черно-белые принтеры полностью удовлетворяют такому требованию.

Дисплеи обычно рекомендуется разглядывать с расстояния не ближе 0.5 м. В соответствии с приведенной выше оценкой минимальной разрешающей способности расстоянию 0. 5 м соответствуют приблизительно 200 dpi, В современных дисплеях разрешающая способность составляет 100-120 dpi - это плохо; например, дисплей размером 15" по диагонали должен обеспечивать не 1024x768 пикселов, а вдвое больше. Но на современном уровне развития техники это пока что невозможно.

3.2 Вывод изображений на растровые устройства

Для иллюстрации работы реальных растровых устройств рассмотрим результаты отображения рисунка-образца на разнообразных графических устройствах. Поскольку в этой книге невозможно показать цветные изображения, в качестве тестового образца выбран черно-белый рисунок, который состоит из текста и простейшей графики - текст «Строчка текста».

Графика — векторный рисунок из линий минимально возможной толщины. Тестовый образец изготовлен и выведен на устройства с помощью редактора Word 2000.

Почему именно такой образец? Для того чтобы оценить погрешности отображения, тест следует подобрать так, чтобы устройства работали в режиме близком к предельно допустимому. Тогда и следует оценивать их возможности. Однако задача усложняется тем, что проверяются устройства разного класса. Оказалось, что некоторые устройства не в состоянии удовлетворительно отобразить даже такой простой образец, а некоторые устройства продемонстрировали значительный запас точности — для них нужны другие тесты.

После вывода образца на графическом устройстве, соответствующее растровое изображение оцифровывалось сканером с оптическим разрешением 600x600 dpi (2400x2400 в режиме интерполяции). Также использовалась фотокамера в режиме макросъемки.

Безусловно, погрешность сканера важна для полученных на устройствах изображений, обладающих сопоставимым, а также более высоким разрешением. Однако приведенные здесь результаты не следует рассматривать как точные измерения. Здесь ставились иные цели — проиллюстрировать геометрические свойства растров (расположение, форму и размеры отдельных пикселов) для устройств различного типа, показать наиболее характерные особенности отображения.

Для сравнения были выбраны графические устройства, которые можно встретить практически в любом современном офисе — это дисплеи и принтеры.

Торговые марки устройств не приводятся. Наше изучение особенностей их работы не следует рассматривать как тестирование или рекламу.

Первый пример — изображение на экране цветного монитора, на электронно-лучевой трубке (рис. 3.6). Следует заметить, что в данном случае изображение черно-белого образца на самом деле — цветное, в книге оно напечатано в градациях серого.

На рис. 3.6 показано увеличенное изображение фрагмента. Здесь уже четко видно "триадную" структуру растра, присущую цветному кинескопу.

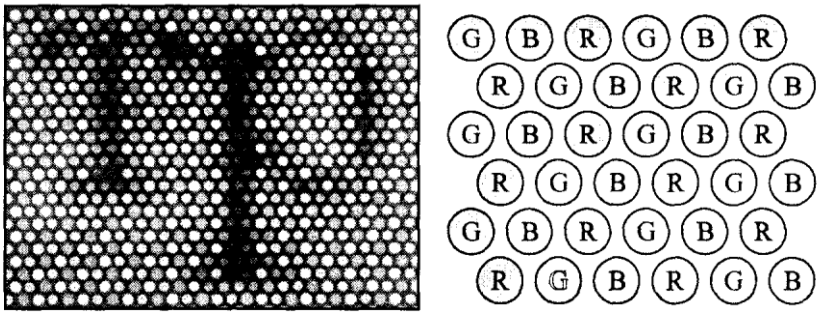
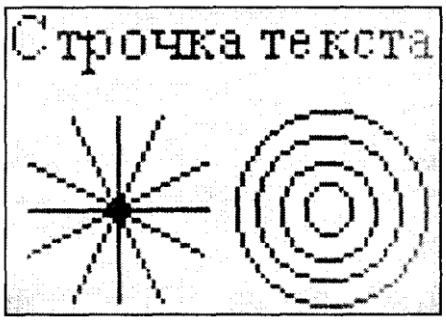


Рис. 3.6. Растр электронно-лучевой трубки - триады RGB

Растровый характер изображения монитора на жидких кристаллах (рис. 3.7) выражен значительно четче, чем для монитора на электронно-лучевой трубке. Четкость отдельных пикселов обуславливает заметный ступенчатый эффект наклонных линий.



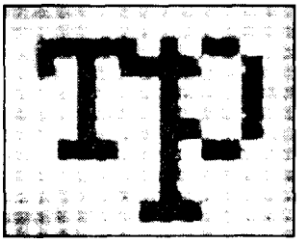


Рис. 3.7. Монитор на жидких кристаллах. Видеорежим 1024 на 768, экран ноутбука 14”

Растр монитора на жидких кристаллах

Качество печати для матричных принтеров определяется погрешностями механики и износом красящей ленты (рис. 3.8). Здесь красящая лента выработала свой ресурс наполовину, поэтому изображение получилось как бы "в градациях серого цвета". Кроме того, изображение имеет полутоновый характер и из-за того, что чернота уменьшается на краях впадин оттиска игл. Вообще говоря, матричные принтеры могут печатать намного лучше. Даже испытуемый принтер может печатать с разрешением 240 на 216 dpi. Однако драйвер для Windows позволяет установить только 240x144 dpi, а качество практически не улучшается по сравнению с 120x144 (вероятно, из-за износа механики).

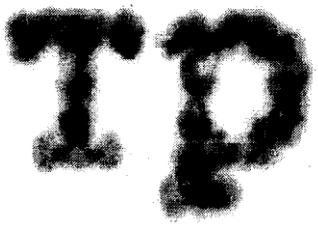


Рис. 3.8. Матричный 9-игольчатый принтер 120 на 144 dpi

Увеличенный фрагмент

Лазерные принтеры, как правило, безупречно отрабатывают свою паспортную разрешающую способность (рис. 3.9). Немаловажным является то, что качество печати стабильно и практически не зависит от качества бумаги. Принтеры данного типа вне конкуренции (по крайней мере, в настоящее время) по быстродействию и качеству черно-белой печати среди других типов принтеров. Более дорогие модели лазерных принтеров обладают в несколько раз большей паспортной разрешающей способностью, при этом качество печати, как правило, возрастает соответственно. Оптического разрешения сканера в 600 dpi (2400 dpi интерполяция) уже недостаточно, чтобы точно отобразить фрагмент растра в мельчайших деталях.

Рис. 3.9. Лазерный черно-белый принтер, 600 dpi Фрагмент изображения

Качество печати струйных принтеров достаточно редко соответствует заявленной паспортной разрешающей способности (рис. 3.10). Данная модель, возможно, — исключение из общего правила. В черно-белом режиме здесь фактически продемонстрирована точность печати на уровне 600 dpi лазерного принтера. Многие другие струйные принтеры с рекламируемым разрешением более тысячи dpi работают еще хуже. И это при печати на специальной бумаге.

Достоинством струйных принтеров является то, что это относительно недорогое устройство для цветной печати. С приемлемым качеством для цветной фотографии работают струйные фотопринтеры. Технология струйной печати также используется и в достаточно популярных крупноформатных (А3-А1) цветных растровых принтерах.

Рис. 3.10. Струйный цветной фотопринтер, черно-белый режим, 1440 dpi, печать на специальной фотобумаге Фрагмент изображения

3.3 Методы улучшения растровых изображений

Рассмотрим некоторые из существующих методов улучшения качества изображений, которые основываются на субъективном восприятии разрешающей способности и количества цветов. При одних и тех же значениях технических параметров устройства графического вывода можно создать иллюзию увеличения разрешающей способности или количества цветов. Причем, субъективное улучшение одной характеристики происходит за счет ухудшения другой.

Устранение ступенчатого эффекта

В растровых системах при невысокой разрешающей способности (меньше 300 dpi) существует проблема ступенчатого эффекта *(aliasing)* — при большом шаге сетки растра пикселы линий образуют как бы ступени лестницы.

Рассмотрим это на примере отрезка прямой линии. Вообще говоря, растровое изображение объекта определяется алгоритмом закрашивания пикселов, соответствующих телу изображаемого объекта. Разные алгоритмы могут дать существенно отличающиеся варианты растрового изображения одного и того же объекта. Можно сформулировать условие корректного закрашивания следующим образом — если в контур изображаемого объекта попадает больше половины площади ячейки сетки растра, то соответствующий пиксел закрашивается цветом объекта (С), иначе — пиксел сохраняет цвет фона (Сф).

На рис. 3.11 показано растровое изображение толстой прямой линии, на которую для сравнения наложен идеальный контур исходной линии.

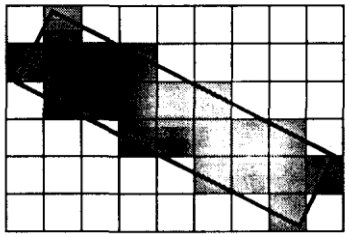


Рис. 3.11. Растровое изображение отрезка линии

Устранение ступенчатого эффекта называется на английском языке *antialiasing.* Для того чтобы растровое изображение линии выглядело более ровным, можно цвет угловых пикселов "ступенек лестницы" заменить определенным оттенком, промежуточным между цветом объекта и цветом фона. Будем вычислять цвет пропорционально части площади ячейки растра, покрываемой идеальным контуром объекта. Если площадь всей ячейки обозначить как *S,* а часть площади, покрываемой контуром, — *Sx,* то искомый цвет равняется

$$C_x = \frac{C \cdot S_x + C\phi \cdot (S - S_x)}{S}.$$

На рис. 3.12 показано сглаженное растровое изображение, построенное указанным выше методом.

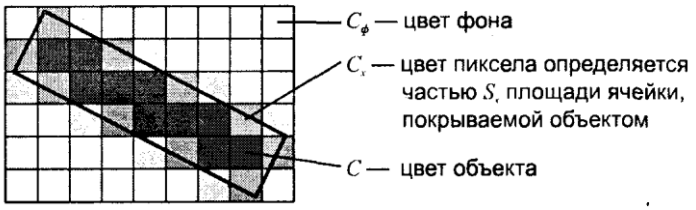


Рис. 3.12. Сглаживание

Методы визуализации сглаженных растровых изображений можно разделить на две группы. Первую группу составляют алгоритмы растеризации для отдельных примитивов — линий, фигур с заполнением. В ходе вывода последовательности примитивов для любого пиксела текущего примитива рассчитывается соответствующий цвет с учетом сглаживания.

Другую группу методов сглаживания составляют методы обработки уже существующего изображения. Для сглаживания растровых изображений часто используют алгоритмы цифровой фильтрации. Один из таких алгоритмов — локальная фильтрация. Она осуществляется путем взвешенного суммирования яркости пикселов, расположенных вокруг текущего обрабатываемого пиксела. Можно представить себе, что в ходе обработки изображения по растру скользит прямоугольное окно, которое выхватывает пикселы (рис. 3.13).

Для определения цвета текущего пиксела вычисляется некоторая функция, учитывающая значение цветов пикселов этого окна. Базовую операцию такого фильтра можно представить так:

$$F_{x,y} = \frac{1}{K} \sum_{i=i_{min}}^{i_{max}} \sum_{j=j_{min}}^{j_{max}} P_{x+j,y+i} \cdot M_{i-i_{min},j-j_{min}},$$

где P — значение цвета текущего пиксела, F — новое значение цвета пиксела, K — нормирующий коэффициент, M — двумерный массив коэффициентов, который определяет свойства фильтра (обычно этот массив называют "маской").

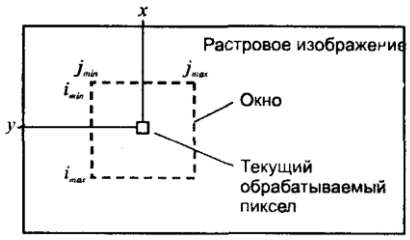


Рис. 3.13. Окно локального цифрового фильтра

Размеры окна фильтра: $(j_{max} - j_{min} + 1)$ — по горизонтали и $(i_{max} - i_{min} + 1)$ — по вертикали.

При $i_{min}, j_{min} = -1$, и $i_{max}, j_{max} = +1$ имеем фильтр с окном 3x3, частоиспользуемый на практике.

Для обработки всего растра необходимо выполнить указанные выше вычисления для каждого пиксела. Если в ходе обработки новые значения цвета пикселов записываются в первоначальный растр и вовлекаются в вычисления для очередных пикселов, то такую фильтрацию называют ре*курсивной.* При *нерекурсивной фильтрации* в вычисления вовлекаются только прежние значения цвета пикселов. Нерекурсивность можно обеспечить, если новые значения записывать в отдельный массив. Рекурсивный фильтр может дать больший эффект сравнительно с нерекурсивным фильтром — изменение цвета одного пиксела может привести к изменениям пикселов всего изображения. Поведение нерекурсивного фильтра в этом смысле более предсказуемо.

На рис. 3.14 представлены результаты работы двух вариантов нерекурсивного сглаживающего фильтра, использующего окно (маску) 3x3.

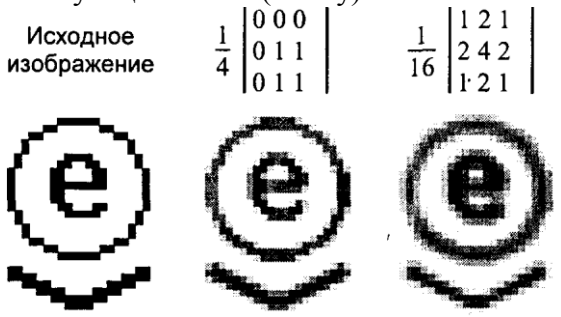


Рис. 3.14. Два сглаживающих фильтра

Значение нормирующего коэффициента здесь выбрано равным сумме элементов маски. Этим обеспечивается сохранение масштаба яркости преобразованного растра. Отметим, что маска — это не матрица, а массив коэффициентов, располагающихся соответственно пикселам окна. Средний фильтр можно задать и маской 2x2 — отбросить нулевые коэффициенты.

При сглаживании цветных изображений можно использовать модель RGB и производить фильтрацию по каждому компоненту.

С помощью локальной цифровой фильтрации можно выполнять довольно разнообразную обработку изображений — повышение резкости, выделение контуров и многое другое.

Дизеринг

Хорошо, если растровое устройство отображения может прямо воссоздавать тысячи цветов для любого пиксела. Не так уже и давно это было проблемой даже для компьютерных дисплеев (а точнее — для видеоадаптеров). Современные растровые дисплеи достаточно качественно воспроизводят миллионы цветов, благодаря чему без проблем можно отображать цветные фотографии. Но для растровых устройств, которые печатают на бумаге, положение другое. Устройства печати обычно имеют высокую разрешающую способность (dpi), часто на порядок выше, чем дисплеи. Однако они не могут непосредственно воссоздать даже сотню градаций серого для пикселов черно-белых фотографий, не говоря уже о миллионах цветов. Вы можете возразить, что в любой газете

или журнале мы видим иллюстрации. Возьмите лупу и посмотрите, например, на изображение любой напечатанной фотографии. В большинстве случаев можно увидеть, что оттенки цветов (для цветных изображений) или градации серого (для черно-белых) имитируются комбинированием, смесью точек. Чем качественнее полиграфическое оборудование, тем меньше отдельные точки и расстояние между ними.

Вообразим себе, что отдельные точки на фотографии нельзя различить даже с помощью лупы. Это может быть в таких случаях: или нам посчастливилось увидеть печать многими сотнями красок, или разрешающая способность устройства печати очень высокая. Много красок при высокой разрешающей способности растра — это пока что фантастика. Однако, безусловно, с течением времени будут изобретены способы печати если не многими тысячами красок (что маловероятно), то хотя бы красками, которые плавно изменяют свой цвет, или будет изобретена бумага с соответствующими свойствами.

Для устройств печати на бумаге проблема количества красок достаточно важна. В полиграфии для цветных изображений обычно используют три цветных краски и одну черную, что в смеси дает восемь цветов (включая черный цвет и белый цвет бумаги). Встречаются образцы печати с большим количеством красок — например, карты, напечатанные с использованием восьми красок, однако такая технология печати намного сложнее. Состояние дел с цветной печатью можно оценить также на примере относительно простых офисных принтеров. Недавно появились струйные принтеры с увеличенным количеством красок. В таких принтерах в состав обычных CMYK-красок добавлены бледно-голубая, бледно-лиловая и бледно-желтая краски (семицветные принтеры). В шестицветных принтерах отсутствует бледно-желтая краска. Увеличение количества красок значительно улучшило качество печати, однако и этого пока явно мало.

Если графическое устройство не способно воссоздавать достаточное количество цветов, тогда используют *растрирование* — независимо от того, растровое это устройство или нерастровое. В полиграфии растрирование известно давно. Оно использовалось несколько столетий тому назад для печати гравюр. В гравюрах изображение создается многими штрихами, причем полутоновые градации представляются или штрихами разной толщины на одинаковом расстоянии, или штрихами одинаковой толщины с переменной густотой расположения. Такие способы используют особенности человеческого зрения и в первую очередь — пространственную интеграцию. Если достаточно близко расположить маленькие точки разных цветов, то они будут восприниматься как одна точка с некоторым усредненным цветом. Если на плоскости густо расположить много маленьких разноцветных точек, то будет создана визуальная иллюзия закрашивания плоскости определенным усредненным цветом. Однако, если увеличивать размеры точек и (или) расстояние между ними, то иллюзия сплошного закрашивания исчезает — включается другая система человеческого зрения, которая обеспечивает способность различать объекты, подчеркивать контуры.

В компьютерных графических системах часто используют эти методы. Они позволяют увеличить количество оттенков цветов за счет снижения пространственного разрешения растрового изображения. Иначе говоря — это обмен разрешающей способности на количество цветов. В литературе по КГ такие методы растрирования получили название *dithering* (разрежение, дрожание).

Рассмотрим методы дизеринга. Простейшим вариантом дизеринга можно считать создание оттенка цвета парами соседних пикселов.

Если рассмотреть ячейки из двух пикселов (рис. 3.15), то ячейка номер 1 дает оттенок цвета C:

$$C = \frac{C_1 + C_2}{2},$$

где C_1 и C_2 — цвета, которые графическое устройство непосредственно способно воссоздать для любого пиксела. Числовые значения C, C_j и C_2 можно рассчитать в полутоновых градациях или в модели RGB — в отдельности для любого компонента.

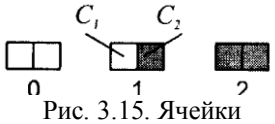


Рис. 3.15. Ячейки

Пример растра с использованием ячеек из двух пикселов приведен на рис. 3.16. Как видим, для создания промежуточного оттенка (С) ячейки образуют вертикальные линии, которые очень заметны. Для того чтобы человек воспринял это как сплошной оттенок, необходимо, чтобы угловой размер ячеек был меньше одной угловой минуты. Можно изменять положения таких ячеек в растре, располагая их, например, по диагонали. Это лучше, но не намного.

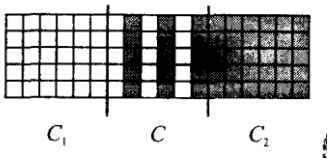


Рис. 3.16. Простейший дизеринг

Наиболее часто используют квадратные ячейки больших размеров. Приведем пример ячеек размером 2x2 (рис. 3.17). Такие ячейки дают 5 градаций, из них три комбинации (1, 2, 3) образуют новые оттенки.

Предоставим также примеры ячеек других размеров (рис. 3.18).

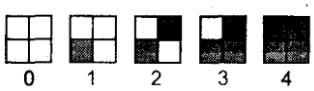


Рис. 3.17. Ячейки 2x2

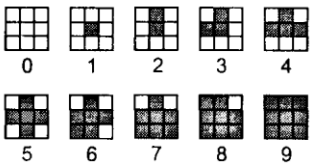
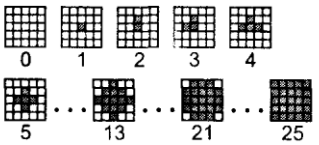


Рис. 3.18. Ячейки 3x3 предоставляют 10 градаций

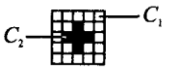


Ячейки 5x5 дают 26 градаций

Расчет цвета, который соответствует одной из комбинаций пикселов в ячейке, можно выполнить следующим образом. Если пикселы ячейки могут быть только двух цветов (C_1 и C_2), то необходимо подсчитать часть площади ячейки для пикселов каждого цвета. Цвет ячейки C можно оценить соотношением

$$C = \frac{S_1 C_1 + (S - S_1) C_2}{S} = \frac{S_1 C_1 + S_2 C_2}{S},$$

где S— общая площадь ячейки; S_1 и S_2 — части площади, занятые пикселами цветов C_1 и C_2 соответственно, причем $S_1 + S_2 = S$. Проще всего, если пикселы квадратные, а их размер равняется шагу расположения пикселов. Примем площадь одного пиксела за единицу. В этом случае площадь, занятая пикселами в ячейке, равняется их количеству (рис. 3.19).



$$S = 25, S_1 = 20, S_2 = 5, C = \frac{20 C_1 + 5 C_2}{25}$$

Рис. 3.19. Площадь определяется количеством пикселов

Для ячейки 5x5, изображенной на рисунке 3.18 (справа), рассчитаем цвет С для некоторых цветов C_1 и C_2. Если C_1 — белый цвет (R_1 G_1 B_1) = (255, 255, 255), а C_2 — черный (R_2 G_2 B_2) = (0, 0, 0), тогда

$$C = \begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} (S_1R_1 + S_2R_2)/S \\ (S_1G_1 + S_2G_2)/S \\ (S_1B_1 + S_2B_2)/S \end{bmatrix} = \begin{bmatrix} 204 \\ 204 \\ 204 \end{bmatrix},$$

то есть мы получили светло-серый цвет.

Еще пример. Если C_1 — желтый (R_1 G_1 B_1) = (255, 255, 0), а C_2 — красный (R_2 G_2 B_2) = (255, 0, 0), то C= (255, 204, 0). Это оттенок оранжевого.

Итак, если в ячейке размерами n x n использованы два цвета, то с помощью этой ячейки можно получить $n^2 + 1$ разных цветовых градаций. Две комбинации пикселов ячейки — если все пикселы ячейки имеют цвет C_1 или C_2 — дают цвет соответственно C_1 или C_2. Все другие комбинации дают оттенки, промежуточные между C_1 и C_2.

Можно считать, что ячейки размером *nxn* образуют растр с разрешающей способностью в n раз меньшей, чем у начального растра, а глубина цвета возрастает пропорционально n^2. Для характеристики изображений, которые создаются методом дизеринга, используют термин — *линиатура растра.* Линиатура вычисляется как количество линий (ячеек) на единицу длины — сантиметр, миллиметр, дюйм. В последнем случае единица измерения для линиатуры — lpi (по аналогии с dpi).

Как реализовать метод дизеринга в графической системе? Рассмотрим примеры преобразования растрового изображения размером *pxq* с определенной глубиной цвета в другой растр, предназначенный для отображения с помощью графического устройства, в котором используется ограниченное количество основных цветов. В таком случае надо выбрать размеры ячейки *mxn,* которые обеспечивают достаточное количество цветовых градаций. Потом любой пиксел растра превращается в пиксел растра отображения. Это можно осуществить двумя способами.

Первый способ. Любой пиксел заменяется ячейкой из mxn пикселов. Это самое точное преобразование по цветам, но размер растра увеличивается и составляет *mpxnq* пикселов.

Второй способ. Здесь размеры растра не изменяются. Цвет любого пиксела преобразованного растра вычисляется следующим образом.

- Определяем координаты пиксела *(x, y)* для преобразуемого растра.
- Находим цвет пиксела *(x, y).*
- По цвету пиксела находим номер *(к)* ячейки, наиболее адекватно представляющей этот цвет.
- По координатам *(x, y)* вычисляем координаты пиксела внутри ячейки:

$$x_k = x \bmod m,$$
$$y_k = y \bmod n.$$

- Находим цвет (C) пиксела ячейки с координатами (x_k, y_k).
- Записываем в преобразованный растр пиксел *(x, y) с* цветом C.

Такой способ можно использовать не для любых вариантов расположения пикселов в ячейках. Конфигурации пикселов должны быть специально разработаны для таких преобразований. Одно из требований можно сформулировать так. Если ячейки разработаны на основе двух цветов, например, белого и черного, а градации изменяются пропорционально номеру ячейки, то необходимо, чтобы ячейка с номером (i) для более темной градации серого содержала бы все черные пикселы ячейки номер (i - 1).

Рассмотрим пример изображения, созданного на основе ячеек 5x5.

Для создания такого изображения специально была выбрана небольшая разрешающая способность, чтобы подчеркнуть структуру изображения. Ячейки образуют достаточно заметный квадратный растр (рис. 3.20).

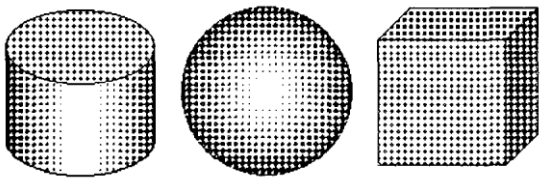
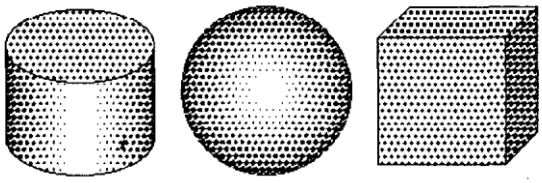


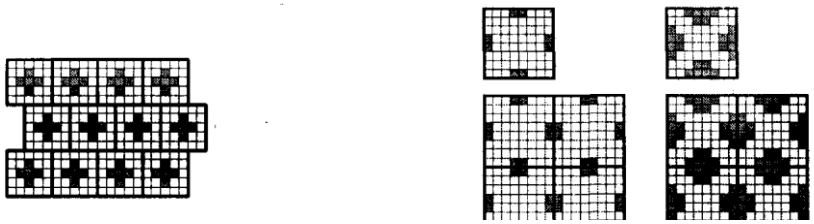
Рис. 3.20. Пример квадратного растра с ячейками 5x5

Для улучшения восприятия изображения можно использовать другое расположение ячеек, например, диагональное (рис 3.21).



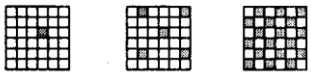
3.21. Диагональное расположение ячеек 5x5

Диагональное расположение можно получить, если сдвигать четные строки ячеек. А для того чтобы получить диагональную структуру растра, подобную той, что используется для печати газет, можно использовать квадратное расположение ячеек другого типа (рис. 3.22).



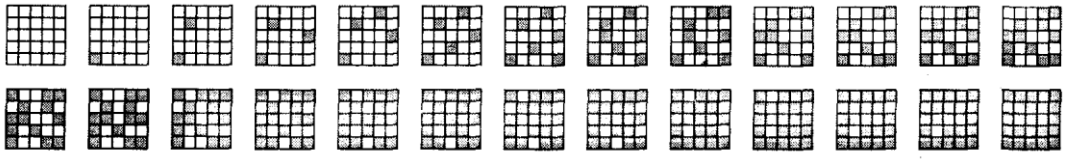
3.22. Диагональные структуры: а - сдвиг строк ячеек, б - ячейки другого типа

Вы, наверно, уже заметили, что для всех приведенных выше примеров дизеринга ячейки образуют точки переменного размера с постоянным шагом. Однако часто используется другой подход — переменная плотность расположения точек постоянного размера. Такой способ получил название частотной модуляции (ЧМ) (рис. 3.23).



3.23. ЧМ-ячейки 6x6

Положительная черта способа ЧМ — меньшая заметность структуры растра. Однако его использование усложнено в случае, когда размер пикселов больше, чем шаг. Начиная с определенной плотности, пикселы смыкаются. Кроме того, на дискретном растре невозможно обеспечить плавное изменение плотности (частоты), в особенности для ячеек небольшого размера. Рассмотрим пример ячеек 5x5, реализующих ЧМ-дизеринг (рис. 3.24).



3.24. Набор ЧМ-ячеек 5x5

Для изображений, созданных методом ЧМ-дизеринга, растрирование менее заметно (рис. 3.25).

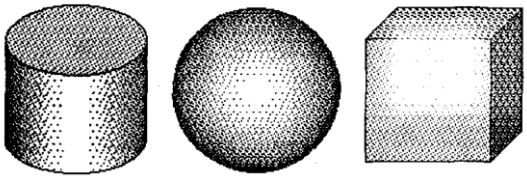


Рис. 3.25. Диагональное расположение ЧМ-ячеек 5x5

Общим недостатком методов, использующих регулярное расположение одинаковых ячеек, является то, что всегда образуется текстура, появляется муар, лишние контуры. Одной из важных задач исследований в этой области считалась разработка таких вариантов ячеек, которые обуславливают наименее заметную растровую структуру (кроме тех случаев, когда наоборот, такую структуру нужно подчеркнуть для создания изображения в стиле гравюры).

Другой разновидностью дизеринга являются методы, в которых вообще не используются ячейки. Одним из популярнейших методов дизеринга в настоящее время является ***"'error diffusion"*** метод *Флойда-Стейнберга*. Он обеспечивает высокое качество изображений и часто используется в драйверах принтеров и графических редакторах (рис. 3.26).

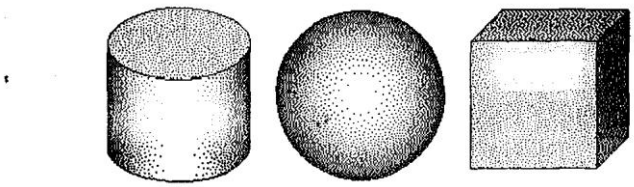


Рис. 3.26. Дизеринг методом «error-diiffusion» Флойда-Стейнберга

Рассмотрим алгоритм Флойда-Стейнберга. При обработке первой строки растра для первого пиксела цвет (С) заменяется на ближайший из возможных *(X).* Например, пиксел серого цвета заменяется соответствующим черным или белым. Для этого пиксела вычисляется ошибка *E = C-X.* Эта ошибка в пропорции (7/16, 5/16, 3/16, 1/16) распределяется по соседним пикселам (отсюда и название метода — *"error diffusion* "). При обработке следующего пиксела рассматривается уже сумма его собственного цвета плюс значение ошибки, которое дошло к этому пикселу. Как распространяется ошибка — это зависит от направления сканирования строки (рис. 3.27).

Рис. 3.27. Добавление частей ошибки к соседним символам Напрвления сканирования

Для уменьшения вероятности образования регулярных узоров рекомендуется соседние строки сканировать в противоположных направлениях — "змейкой".

Этот метод был предложен для градаций серого, однако, он с успехом используется, например, для преобразования 24-битных изображений в 256-цветные.

3.4. Базовые растровые алгоритмы

Алгоритмы вывода прямой линии

Рассмотрим растровые алгоритмы для отрезков прямой линии. Предположим, что заданы координаты (*x1, yl - x2, y2)* концов отрезка прямой. Для вывода линии необходимо

закрасить определенным цветом все пикселы вдоль линии. Для того чтобы закрасить любой пиксел, необходимо знать его координаты.

Наиболее просто нарисовать отрезок горизонтальной линии.

Вычисление текущих координат пиксела выполняется как приращение по *x* (необходимо, чтобы $x1 \le x2$), а вывод пиксела обеспечивается специальной функцией.

Аналогично рисуется отрезок вертикали.

В цикле вывода горизонтального и вертикального отрезков выполняются простейшие операции — приращение на единицу, проверка на "< =" и запись пиксела в буфер растра. Поэтому операция рисования таких отрезков выполняется быстро и просто. Ее используют как базовую операцию для других операций, например, в алгоритмах заполнения полигонов.

Можно задать такой вопрос: какая линия рисуется быстрее — горизонталь или вертикаль? На первый взгляд — одинаково быстро. Если учитывать только математические аспекты, то скорость должна быть одинаковой при равной длине отрезков линий, поскольку в обоих случаях выполняется одно и то же количество одинаковых операций. Однако если кроме вычисления координат анализировать вывод пикселов в конкретный растр, то могут быть отличия. В растровых системах рисование пиксела обычно означает запись одного или нескольких битов в память, где сохраняется растр. И здесь уже не все равно — по строкам или по столбцам заполняется растр. Необходимо учитывать логическую организацию памяти компьютера, в которой хранятся биты или байты растра. Даже для компьютеров одного типа (например, персональных компьютеров) для разных поколений процессоров и памяти скорость записи по соседним адресам может существенно отличаться от скорости записи по не соседним адресам. В особенности это заметно, если для растра используется виртуальная память с сохранением отдельных страниц на диске и (или) в оперативной памяти (RAM). При работе графических программ в среде операционной системы Windows часто случается так, что горизонтали рисуются быстрее вертикалей, так как в каждой странице памяти сохраняются соседние байты — пикселы вдоль горизонтали растра. Подобный эффект также имеет место при использовании кэш-памяти. А может быть, что RAM достаточно, и даже весь растр размещается в кэше, а скорости рисования все же отличаются. Например, если используется черно-белый растр в формате один бит на пиксел, то для вертикали битовая маска одинакова для всех пикселов линии, а для горизонтали маску нужно изменять на каждом шаге. Здесь необходимо заметить, что рисование черно-белых горизонталей можно существенно ускорить, если записывать сразу восемь соседних пикселов — байт в памяти.

Горизонтали и вертикали представляют собой частный случай линий. Рассмотрим линию общего вида. Для нее также необходимо вычислять координаты любого пиксела. Известны несколько методов расчетов координат точек линии.

Прямое вычисление координат

Пусть заданы координаты конечных точек отрезка прямой. Найдем координаты точки внутри отрезка.

Запишем соотношения катетов для подобных прямоугольных треугольников:

$$\frac{x - x_1}{y - y_1} = \frac{x_2 - x_1}{y_2 - y_1}.$$

Перепишем это соотношение как x =*f(y):* $x = x_1 + (y - y_1)\frac{x_2 - x_1}{y_2 - y_1}$,

а также как

$y = F(x)$: $y = y_1 + (x - x_1)\frac{y_2 - y_1}{x_2 - x_1}$.

В зависимости от угла наклона прямой выполняется цикл по оси x или по y (рис. 3. 28).

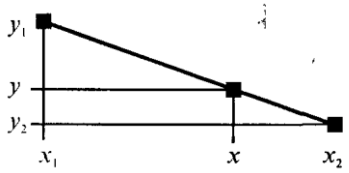


Рис. 3.28. Отрезок прямой.

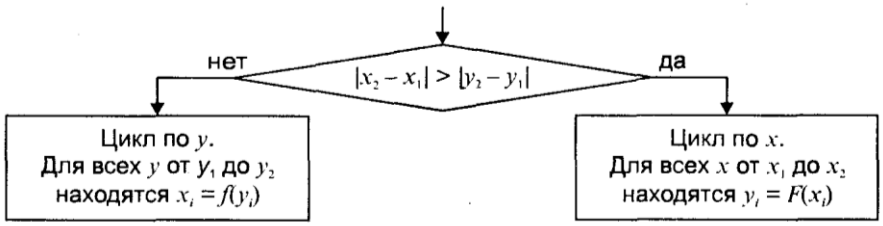


Рис. 3.29. Общая схема алгоритма вывода отрезка прямой линии

Положительные черты прямого вычисления координат.

1. Простота, ясность построения алгоритма.
2. Возможность работы с нецелыми значениями координат отрезка. (Как вы считаете, в каком варианте из четырех корректно вычисляются координаты пикселов, если *x1, y1, x2 и y2* — дробные?)

Недостатки.

1. Использование операций с плавающей точкой или целочисленных операций умножения и деления обуславливает маленькую скорость. Однако это зависит от процессора, и для разных типов компьютеров может быть по-разному. В современных компьютерах, в которых процессоры используют эффективные средства ускорения (например, конвейер арифметических операций с плавающей точкой), время выполнения целочисленных операций уже не намного меньше. Для старых компьютеров разница могла составлять десятки раз, поэтому и старались разрабатывать алгоритмы только на основе целочисленных операций.
2. При вычислении координат добавлением приращений может накапливаться ошибка вычислений координат.

Последнюю разновидность прямого вычисления координат, рассмотренную здесь, можно было бы назвать "инкрементной". Но такое название получили алгоритмы другого типа.

Инкрементные алгоритмы

Брезенхэм предложил подход, позволяющий разрабатывать так называемые *инкрементные алгоритмы растеризации.* Основной целью при разработке таких алгоритмов было построение циклов вычисления координат на основе только целочисленных операций сложения/вычитания без использования умножения и деления. Были разработаны инкрементные алгоритмы не только для прямых, но и для кривых линий.

Инкрементные алгоритмы выполняются как последовательное вычисление координат соседних пикселов путем добавления приращений координат. Приращения рассчитываются на основе анализа функции погрешности. В цикле выполняются только целочисленные операции сравнения и сложения/вычитания. Достигается повышение быстродействия для вычислений каждого пиксела по сравнению с прямым способом.

Рассмотрим пример работы приведенного выше алгоритма Брезенхэма для отрезка (*x1y1 - x2y2)* = (2, 3 - 8, 6). Этот алгоритм восьмисвязный, то есть при выполнении приращений координат для перехода к соседнему пикселу возможны восемь случаев (рис. 3. 30).

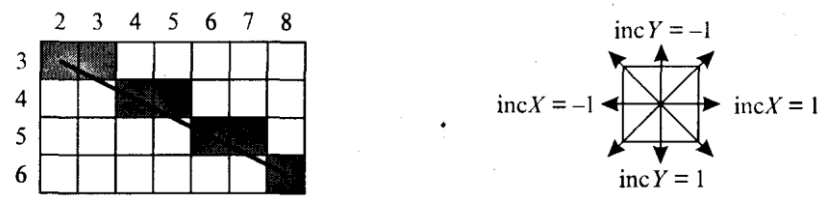


Рис. 3.30. Восьмисвязность

Известны также четырехсвязные алгоритмы (рис. 3. 31).

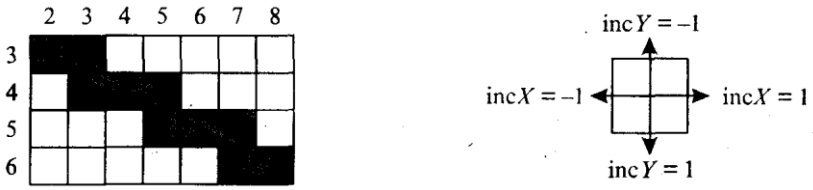


Рис. 3.31. Четырехсвязность

Четырехсвязные алгоритмы проще, но они генерируют менее качественное изображение линий за большее количество тактов работы. Для приведенного примера четырехсвязный алгоритм работает 10 тактов, а восьмисвязный — только 7.

Кривая Безье

Разработана математиком *Пьером Безье.* Кривые и поверхности Безье были использованы в 60-х годах компанией "Рено" для компьютерного проектирования формы кузовов автомобилей. В настоящее время они широко используются в компьютерной графике.

Кривые Безье описываются в параметрической форме: $x = P_x(t), \quad y = P_y(t)$.

Значение t выступает как параметр, которому соответствуют координаты отдельной точки линии. Параметрическая форма описания может быть удобнее для некоторых кривых, чем задание в виде функции y =f(x), поскольку функция f(x) может быть намного сложнее, чем $P_x(t)$ и $P_y(t)$, кроме того, f(x) может быть неоднозначной.

Многочлены Безье для P_x и P_y имеют такой вид:

$$P_x(t) = \sum_{i=0}^{m} C_m^i t^i (1-t)^{m-i} x_i, \quad P_y(t) = \sum_{i=0}^{m} C_m^i t^i (1-t)^{m-i} y_i,$$

где x_i и y_i — координаты точек-ориентиров P_i, а величины C_m^i — это известные из *комбинаторики,* так называемые *сочетания* (они также известны как коэффициенты *бинома Ньютона):*

$$C_m^i = \frac{m!}{i!(m-i)!}.$$

Значение да можно рассматривать и как степень полинома, и как значение, которое на единицу меньше количества точек-ориентиров.

Рассмотрим кривые Безье, классифицируя их по значениям m.

<u>$m = 1$</u> (по двум точкам)

Кривая вырождается в отрезок прямой линии, которая определяется конечными точками P_0 и P_1, как показано на рис. 3. 30:

$$P(t) = (1-t)P_0 + t_1.$$

<u>m=2</u> (по трем точкам, рис. 3. 32): $P(t) = (1-t)^2 P_0 + 2t(1-t)P_1 + t^2 P_2$.

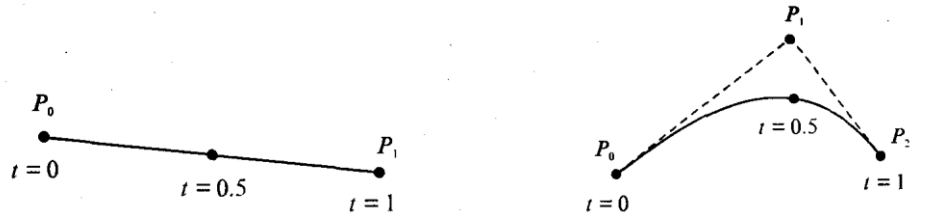


Рис. 3.32. Кривая Безье (m=1) Кривая Безье (m=2)

$m = 3$ (по четырем точкам, кубическая, рис 3.33). Используется довольно часто, в особенности в сплайновых кривых:

$$P(t) = (1-t)^3 P_0 + 3t(1-t)^2 P_1 + 3t^2(1-t)P_2 + t^3 P_3.$$

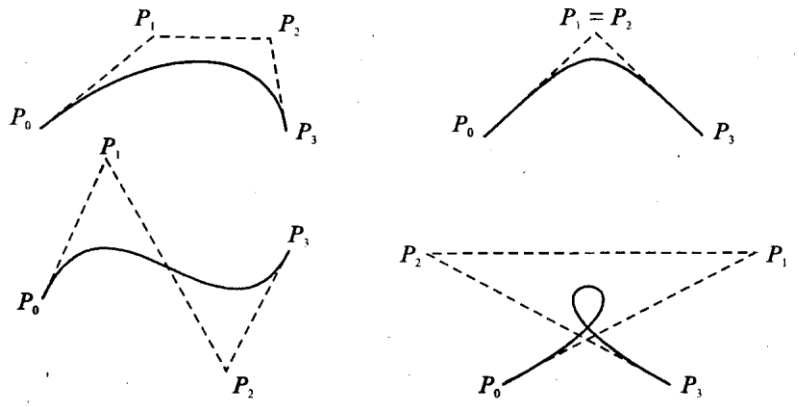


Рис. 3.33. Кубические кривые Безье (m=3)

Геометрический алгоритм для кривой Безье

Этот алгоритм позволяет вычислить координаты (x, *y)* точки кривой Безье по значению параметра *t.*

1. Каждая сторона контура многоугольника, который проходит по точкам-ориентирам, делится пропорционально значению *t.*

2. Точки деления соединяются отрезками прямых и образуют новый многоугольник. Количество узлов нового контура на единицу меньше, чем количество узлов предшествующего контура.

3. Стороны нового контура снова делятся пропорционально значению t. И так далее. Это продолжается до тех пор, пока не будет получена единственная точка деления. Эта точка и будет точкой кривой Безье (рис. 3.34).

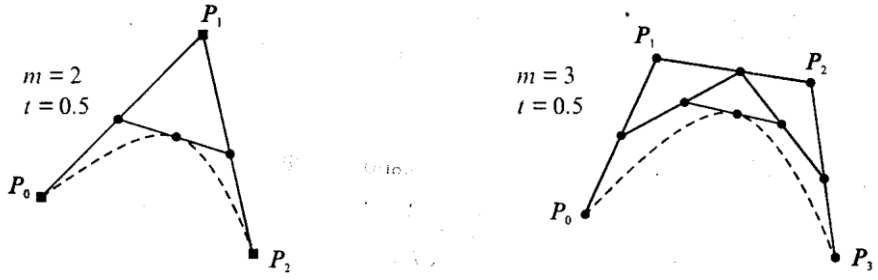


Рис. 3.34. Геометрический алгоритм для кривых Безье

Алгоритмы вывода фигур

Фигурой здесь будем считать плоский геометрический объект, который состоит из линий контура и точек заполнения, которые помещаются внутри контура. Контуров может быть несколько — например, если объект имеет внутри пустоты (рис. 3.35). В некоторых графических системах одним объектом может считаться и более сложная многоконтурная фигура - совокупность островов с пустотами.

Графический вывод фигур делится на две задачи: вывод контура и вывод точек заполнения. Поскольку контур представляет собой линию, то вывод контура проводится

на основе алгоритмов вывода линий. В зависимости от сложности контура, это могут быть отрезки прямых, кривых или произвольная последовательность соседних пикселов.

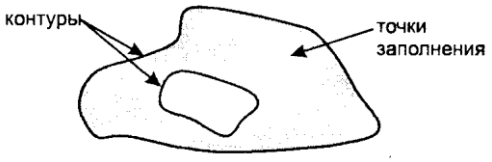


Рис. 3.35. Пример фигуры

Для вывода точек заполнения известны методы, которые разделяются в зависимости от использования контура на два типа: алгоритмы закрашивания от внутренней точки к границам произвольного контура и алгоритмы, которые используют математическое описание контура.

Алгоритмы закрашивания

Рассмотрим алгоритмы закрашивания произвольного контура, который уже нарисован в растре. Сначала определяются координаты произвольного пиксела, находящегося внутри очерченного контура фигуры. Цвет этого пиксела изменяем на нужный цвет заполнения. Потом проводится анализ цветов всех соседних пикселов. Если цвет некоторого соседнего пиксела не равен цвету границы контура или цвету заполнения, то цвет этого пиксела изменяется на цвет заполнения. Потом анализируется цвет пикселов, соседних с предшествующими. И так далее, пока внутри контура все пикселы не перекрасятся в цвет заполнения.

Пикселы контура образуют границу, за которую нельзя выходить в ходе последовательного перебора всех соседних пикселов. Соседними могут считаться только четыре пиксела (сосед справа, слева, сверху и снизу — четырехсвязность), или восемь пикселов (восьми-связность). Не всякий контур может считаться границей закрашивания, например, для восьмисвязного алгоритма (рис. 3. 36).

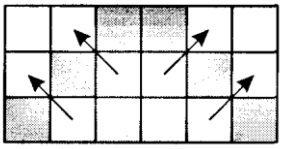


Рис. 3.36.Особенности восьмисвязного закрашивания - выход за границу контура на следующих шагах закрашивания

Простейший алгоритм закрашивания. Для всех алгоритмов закрашивания нужно задавать начальную точку внутри контура с координатами x_0, y_0. Простейший алгоритм можно описать всего двумя шагами.

Алгоритм закрашивания линиями. Данный алгоритм получил широкое распространение в компьютерной графике. От приведенного выше простейшего алгоритма он отличается тем, что на каждом шаге закрашивания рисуется горизонтальная линия, которая размещается между пикселами контура. Алгоритм также рекурсивный, но поскольку вызов функции осуществляется для линии, а не для каждого отдельного пиксела, то количество вложенных вызовов уменьшается пропорционально длине линии. Это уменьшает нагрузку на стековую память компьютера и обеспечивает высокую скорость работы.

Пример работы алгоритма закрашивания линиями приведен на рис 3. 37.

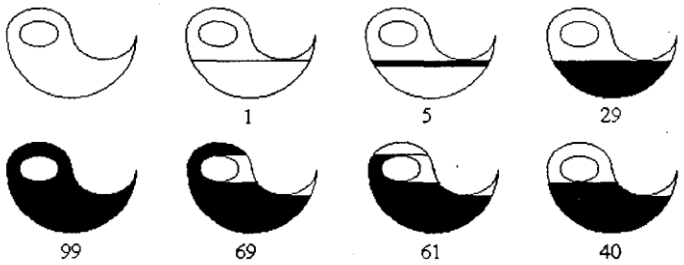


Рис. 3.37. Количество циклов LineFill

Алгоритмы заполнения, которые используют математическое описание контура

Математическим описанием контура фигуры может служить уравнение $y = f(x)$ для контура окружности, эллипса или другой кривой. Для многоугольника (полигона) контур задается множеством координат вершин (x_i, y_i). Возможны и другие формы описания контура. Общим для рассматриваемых ниже алгоритмов есть то, что для генерации точек заполнения не нужны предварительно сформированные в растре пикселы границы контура фигуры. Контур может вообще не рисоваться в растре ни до, ни после заполнения.

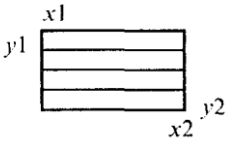


Рис. 3.38. Заполнение прямоугольника

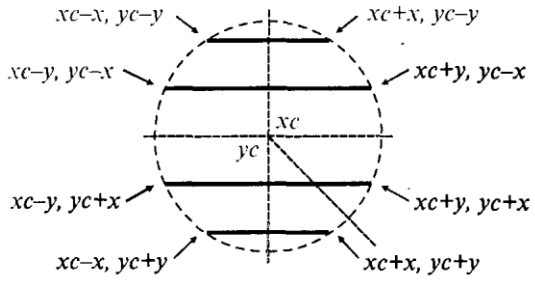
Заполнение прямоугольников. Среди всех фигур прямоугольник заполнять наиболее просто. Если прямоугольник задан координатами противоположных углов, например, левого верхнего *(x1, y1)* и правого нижнего (*x2, y2),* тогда алгоритм может состоять в последовательном рисовании горизонтальных линий заданного цвета (рис. 3. 38).

Заполнение круга. Для заполнения круга можно использовать алгоритм вывода контура. В процессе выполнения этого алгоритма последовательно вычисляются координаты пикселов контура в границах одного октанта. Для заполнения следует выводить горизонтали, которые соединяют пары точек на контуре, расположенные симметрично относительно оси y (рис. 3.39, слева).

Так же можно создать и алгоритм заполнения эллипса.

Заполнение полигонов. Контур полигона определяется вершинами, которые соединены отрезками прямых — ребрами (рис. 3.39, справа). Это — векторная форма описания фигуры.

Рассмотрим один из наиболее популярных алгоритмов заполнения полигона. Его основная идея — закрашивание фигуры отрезками прямых линий. Удобней использовать горизонтали. Алгоритм представляет собою цикл вдоль оси *y,* в ходе этого цикла выполняется поиск точек пересечения линии контура с соответствующими горизонталями. Этот алгоритм получил название *XY* .



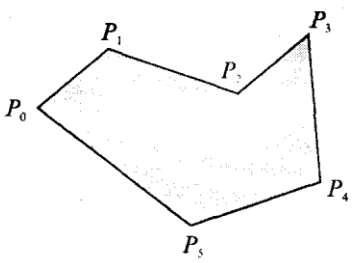


Рис. 3.39. Заполнение круга Пример полигона

В этом алгоритме использовано топологическое свойство контура фигуры. Оно состоит в том, что любая прямая линия пересекает любой замкнутый контур четное

количество раз (рис. 3.40). Для выпуклых фигур точек пересечения с любой прямой всегда две.

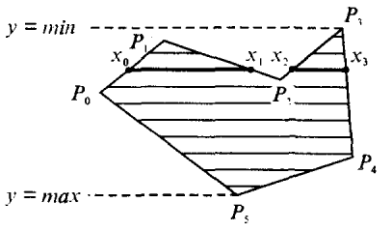


Рис. 3.40. Заполнение полигона

При нахождении точек пересечения горизонтали с контуром необходимо принимать во внимание *особые* точки. Если горизонталь имеет координату *(y),* совпадающую с координатой y_i вершины P_i тогда надлежит анализировать то, как горизонталь проходит через вершину. Если горизонталь при этом *пересекает* контур, как, например, в вершинах P_0 ли P_4, то в массив записывается одна точка пересечения. Если горизонталь *касается* вершины контура (в этом случае вершина соответствует локальному минимуму или максимуму, как, например, в вершинах P_1, P_2, P_3 или P_5), тогда координата точки касания или Не записывается, или записывается в массив два раза. Это является условием четного количества точек пересечения, хранящихся в массиве $\{x_j\}$.

Процедура определения точек пересечения контура с горизонталью, учитывая анализ на локальный максимум, может быть достаточно сложной. Это замедляет работу. Радикальным решением для упрощения поиска точек пересечения может быть сдвиг координат вершин контура или горизонталей заполнения таким образом, чтобы ни одна горизонталь не попала в вершину.

Сдвиг можно выполнять разными способами, например, ввести в растр дробные координаты для горизонталей; y_{min} + 0. 5, y_{min} + 1. 5,..., y_{min} - 0. 5. Но такое упрощение процедуры нахождения точек пересечения приводит к некоторому искажению формы полигона.

Для определения координат (x) точек пересечения для каждой горизонтали необходимо перебирать все η ребер контура. Координата пересечения ребра p_i - p_k с горизонталью *(y)* равняется

$$x = x_i + (y_k - y)(x_к - x_i)/(y_k - y_i).$$

Количество тактов работы этого алгоритма:

$$N_{тактов} \approx (y_{\max} - y_{\min})\, N_{гор},$$

где y_{max}, y_{min} — диапазон координат y, $N_{гор}$— число тактов, нужных для одной горизонтали. Оценим величину $N_{гор}$ как пропорциональную числу вершин

$$N_{гор} \approx k \cdot n,$$

где $к$— коэффициент пропорциональности, n — число вершин полигона.

Возможна модификация приведенного алгоритма для ускорения его работы. Например, можно принять во внимание то, что каждая горизонталь обычно пересекает небольшое количество ребер контура. Поэтому, если при поиске точек пересечения делать предшествующий отбор ребер, которые находятся вокруг каждой горизонтали, то можно добиться уменьшения количества тактов работы с $N_{гор} = к \cdot n$ до $к \cdot n_p$, где n_p — количество отобранных ребер. Например, разделим диапазон y_{min} -y_{max} пополам. Если для диапазона от y_{min} до $y_{ср}$, составить список ребер (или вершин), попадающих в этот диапазон, то в список будет включено приблизительно вдвое меньшее количество, чем для всего диапазона от y_{min} до y_{max}. Почему приблизительно — ибо это зависит от размещения вершин контура полигона. Таким образом, при работе алгоритма для каждой горизонтали в диапазоне от y_{min} до $y_{ср}$ уже нужно не $(к \cdot n)$ тактов, а $(k \cdot n/2)$.

Аналогично для диапазона $y_{ср}$-y_{max} также можно составить список ребер, который также будет почти вдвое меньшим. Если принять, что подсчеты для каждой горизонтали

теперь выполняются за вдвое меньшее количество тактов, а именно за $(N_{гор}/2)$, то общее число тактов:

$$N_{тактов} \approx (y_{\max} - y_{\min})\, N_{гор}/2 + N_{доп},$$

где $N_{доп}$ — количество тактов, которые необходимы при создании списка ребер.

Такой способ повышения быстродействия заполнения полигонов эффективен для большого количества вершин. Контур можно делить не пополам, а на более мелкие части — соответственно повышается скорость.

Приведенные выше алгоритмы заполнения могут быть использованы не только для рисования фигур. На основе алгоритмов заполнения могут быть разработаны алгоритмы для других целей. Например, для определения площади фигуры (если считать площадь пропорциональной количеству пикселов заполнения). Или, например, алгоритм для поиска объектов по внутренней точке — эта операция часто используется в векторных графических редакторах.

Логическое условие будет определять стиль линии. Например, если условием будет четность значения *C,* то получим линию из разрозненных точек. Для рисования пунктирной линии можно анализировать остаток от деления C на *S.* Например, если рисовать пикселы линии только при C mod S < *S/2,* то получим пунктирную линию с длиной штрихов *S/2.* шаг штрихов — S.

При выводе полилиний, а также сплайновых кривых, аппроксимируемых прямолинейными отрезками, необходимо предотвратить обнуление значения счетчика в начале каждого отрезка и обеспечить продолжение непрерывного прироста вдоль всей сложной линии. Иначе будут нестыковки пунктира. Использование переменной-счетчика затруднено при генерации пунктирных линий в алгоритмах, которые используют симметрию, например, при выводе окружности или эллипса. Здесь необходимо обеспечивать стыковку пунктира на границах октантов или квадрантов.

Стиль заполнения

Кисть и текстура

При выводе фигур могут использоваться разные стили заполнения. Простейшее — сплошное заполнение — это когда все пикселы внутри контура фигуры имеют одинаковый цвет. Для обозначения стилей заполнения, отличных от сплошного, используют такие понятия, как *кисть* и *текстура.* Их можно считать синонимами, однако, понятие «текстура» обычно используется применительно к трехмерным объектам, а "кисть" — при изображении двумерных объектов. Текстура — это стиль заполнения, закрашивание, которое имитирует внешний вид, материал поверхности, рельефность трехмерного объекта.

Для описания алгоритмов заполнения фигур в определенном стиле, используем тот же способ, что и для описания алгоритмов рисования линий. Мы уже ранее рассмотрели некоторые алгоритмы заполнения, и описание всех разновидностей подобных алгоритмов можно дать с помощью такой обобщенной схемы:

```
- - - -
Вывод пиксела заполнения цвета C с координатами (x, y)
- - - -
```

Например, в алгоритме вывода полигонов пикселы заполнения рисуются в теле цикла горизонталей, а все другие операции предназначены для расчета координат (x, *y)* этих пикселов. При сплошном заполнении *C = const*. Для других стилей нам нужно в течение цикла вывода фигуры как-то изменять цвет пикселов заполнения, чтобы получить определенный узор. Преобразуем алгоритм заполнения таким образом:

```
- - - -
C = f(x, y)
Вывод пиксела заполнения (x, y) цвета C
- - - -
```

Функция *f*(x, *y)* будет определять стиль заполнения. Аргументами функции цвета являются координаты текущего пиксела заполнения. Однако в отдельных случаях эти аргументы не нужны. Например, если цвет *C* вычислять как случайное значение в определенных границах: *C = random()*, то можно создать иллюзию шершавой матовой поверхности (рис. 3.41).

Другой стиль заполнения — штриховой (рис. 3.42). Для него функцию цвета также можно записать в аналитической форме:

$$f(x, y) = \begin{cases} C_ш, & \text{если } (x + y) \bmod S < T, \\ C_ф & \text{— в других случаях.} \end{cases}$$

где *S* — период, *а T* — толщина штрихов, $C_ш$ — цвет штрихов, $C_ф$ — цвет фона.

Если не рисовать пикселы фона, то можно создать иллюзию полупрозрачной фигуры. Подобную функцию можно записать и для других типов штриховки. Аналитическая форма описания стиля заполнения позволяет достаточно просто изменять размеры штрихов при изменении масштаба показа, что весьма удобно для использования, например, в векторных графических системах. Недостаток аналитической формы - сложность формул для описания естественных материалов, поскольку большое количество вычислительных операций на каждый пиксел заполнения существенно уменьшает скорость визуализации.

Рис. 3.41. Матовая поверхность

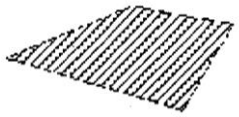


Рис. 3.42. Штриховка

Наиболее часто при использовании кистей и текстур используется наложение специально изготовленных растровых изображений. Такой алгоритм заполнения можно описать вышеупомянутой общей схемой, если строку *C =f(x,y)* заменить двумя другими строками:

```
- - - -
Координаты пиксела заполнения (x, y) преобразуем в растровые
координаты образца кисти (xT, yT)
По координатам (xT, yT) определяем цвет (C) пиксела в образце кисти
Вывод пиксела заполнения цвета C с координатами (x, y)
- - - -
```

Преобразование координат пиксела заполнения (x, *y)* в координаты внутри образца кисти можно выполнить таким образом:

$$x_T = x \bmod m,$$
$$y_T = y \bmod n,$$

где *m, η* — размеры растра образца кисти по горизонтали и вертикали. При этом координаты (x_T, y_T) попадают в диапазон $x_T = 0 \ldots m - 1$, $y_T = 0 \ldots n - 1$ при любых значениях *x и y.* Таким образом, обеспечивается циклическое копирование фрагментов кисти внутри области заполнения фигуры (рис. 3.43).

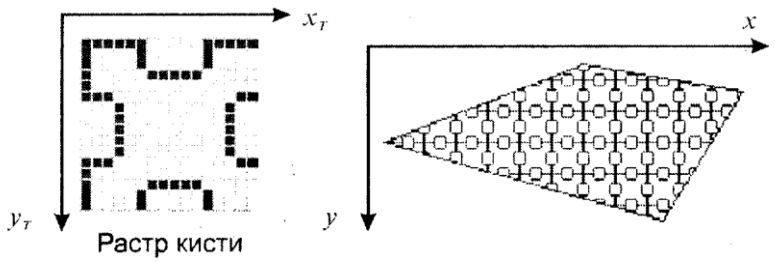


Рис. 3.43. Копирование растра кисти

Удобно, когда размеры кисти равняются степени двойки. В этом случае вместо операций взятия остатка *(mod*) можно использовать более быстрые для цифровых компьютеров поразрядные двоичные операции. Приведем пример быстрого вычисления остатка от деления на 16.

X = **x..xxxxx**

X mod 16 = **0..0xxxx**

Еще один пример. Если необходимо вычислить (X mod 256), а значение X записано в регистре AX микропроцессора (совместимого с 80x86), то для выполнения такой операции достаточно взять содержимое младшей байтовой части этого регистра — AL.

Для пикселов текстур часто употребляется название *тексел.*

Растровые текстуры и кисти широко используются в современной компьютерной графике, в том числе и в 3D-графике. Для отображения трехмерных объектов часто применяются полигональные поверхности, каждая грань отображается с наложенной текстурой. Поскольку объекты обычно показываются из разных ракурсов — повороты, изменения размеров и т. п., необходимо соответственно трансформировать и каждую грань с текстурой. Для этого используются *проективные текстуры.*

Общая схема алгоритма заполнения контуров полигонов для проективных текстур такая же, как приведенная выше. Однако растровый образец здесь представляет всю грань, а преобразование координат из *(x, y)* в (x_T, y_T) сложнее.

Для параллельной проекции можно использовать аффинное преобразование:

$$x_T = A\,x + B\,y + C,$$
$$y_T = D\,x + E\,y + F,$$

где коэффициенты A, *B,...,* F являются константами при пересчете координат всех пикселов для отдельной текстурированной грани. Такое преобразование координат можно использовать, если привязать текстуру к грани по трем опорным точкам. Пример наложения проективной текстуры приведен на рис. 3.44.

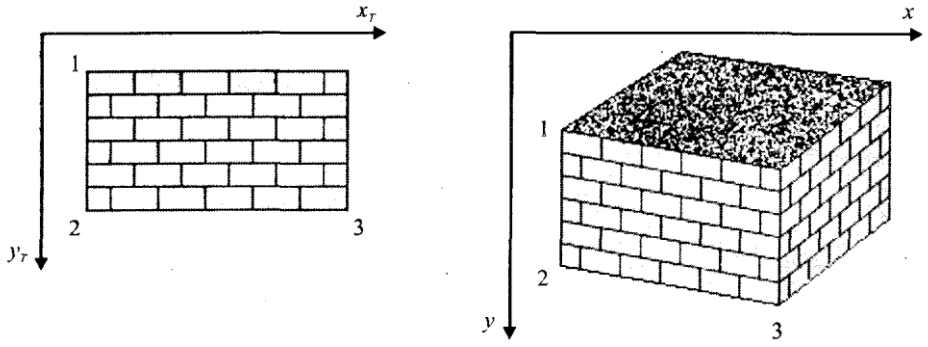


Рис. 3.44. Наложение проективной текстуры (1,2, 3 - опорные точки грани)

Привязывание по трем точкам соответствует системе уравнений

$$x_{Ti} = A\,x_i + B\,y_i + C,$$
$$y_{Ti} = D\,x_i + E\,y_i + F,$$

где i = 1, 2, 3. По известным координатам (x_{Ti}, y_{Ti}) и (*xi, yi*) можно найти коэффициенты *A, B,..., F,* если решить систему линейных уравнений.

Эта система распадается на две независимые системы третьего порядка. Для упрощения записи заменим x_{Ti}, на X_i а y_{Ti} на Y_i,:

$$X_1 = A\,x_1 + B\,y_1 + C, \qquad Y_1 = D\,x_1 + E\,y_1 + F,$$
$$X_2 = A\,x_2 + B\,y_2 + C, \quad \text{и} \quad Y_2 = D\,x_2 + E\,y_2 + F,$$
$$X_3 = A\,x_3 + B\,y_3 + C, \qquad Y_3 = D\,x_3 + E\,y_3 + F.$$

Для решения систем линейных уравнений известно много способов. Используем способ детерминантов. Решение первой системы для коэффициентов *A, B и C* можно записать в виде:

$$A = \det A / \det,$$
$$B = \det B / \det,$$
$$C = \det C / \det,$$

где $\det = \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}$ —это главный детерминант системы,

а детерминанты *detA*, *detB* и *detC* получаются заменой соответствующих столбцов в det столбцом свободных членов

$$\det A = \begin{vmatrix} X_1 & y_1 & 1 \\ X_2 & y_2 & 1 \\ X_3 & y_3 & 1 \end{vmatrix}, \quad \det B = \begin{vmatrix} x_1 & X_1 & 1 \\ x_2 & X_2 & 1 \\ x_3 & X_3 & 1 \end{vmatrix}, \quad \det C = \begin{vmatrix} x_1 & y_1 & X_1 \\ x_2 & y_2 & X_2 \\ x_3 & y_3 & X_3 \end{vmatrix}.$$

Если главный детерминант равняется нулю, то это означает; что решить систему невозможно. Это может быть, например, тогда, когда все три точки *(x_1, y_1), (x_2, y_2) и (x_3, y_3)* располагаются вдоль прямой линии. Однако в этом случае рисовать текстуру и не нужно — грань мы видим с торца. Вычислить детерминант третьей степени можно, например, с помощью "правила Саррюса" . Для этого справа нужно дописать первые два столбца, а потом прибавить (отнять) произведение по диагоналям:

$$\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} = \begin{vmatrix} a_{11} & a_{12} & a_{13} & a_{11} & a_{12} \\ a_{21} & a_{22} & a_{23} & a_{21} & a_{22} \\ a_{31} & a_{32} & a_{33} & a_{31} & a_{32} \end{vmatrix} =$$

$$= a_{11}\,a_{22}\,a_{33} + a_{12}\,a_{23}\,a_{31} + a_{13}\,a_{21}\,a_{32} - a_{13}\,a_{22}\,a_{31} - a_{11}\,a_{23}\,a_{32} - a_{12}\,a_{21}\,a_{33}$$

Вычислим главный детерминант

$$\det = \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} = x_1y_2 + y_1x_3 + x_2y_3 - y_2x_3 - x_1y_3 - y_1x_2.$$

Преобразуем выражение так, чтобы уменьшить количество умножений:

$$\det = x_1\,(y_2 - y_3) + x_2\,(y_3 - y_1) + x_3\,(y_1 - y_2).$$

Аналогично вычисляются *detA* и detB. Детерминант detC - самый сложный. Но его вычислять не обязательно. Запишем решение системы в следующем виде:

$$A = \det A / \det = (X_1\,(y_2 - y_3) + X_2\,(y_3 - y_1) + X_3\,(y_1 - y_2)) / \det,$$
$$B = \det B / \det = (x_1\,(X_2 - X_3) + x_2\,(X_3 - X_1) + x_3\,(X_1 - X_2)) / \det,$$
$$C = X_1 - A\,x_1 - B\,y_1.$$

Таким же способом решаем систему уравнений для *D, E и F*.

$$D = (Y_1\,(y_2 - y_3) + Y_2\,(y_3 - y_1) + Y_3\,(y_1 - y_2)) / \det,$$
$$E = (x_1\,(Y_2 - Y_3) + x_2\,(Y_3 - Y_1) + x_3\,(Y_1 - Y_2)) / \det,$$
$$F = Y_1 - D\,x_1 - E\,y_1.$$

Заметьте, что здесь главный детерминант *det* совпадает с детерминантом первой системы уравнений — для *A, B и C*.

Наложение текстур в центральной (перспективной) проекции осуществляется сложнее, чем в параллельной проекции. Рассмотрим рис. 3.45, на котором изображен текстурированный прямоугольник.

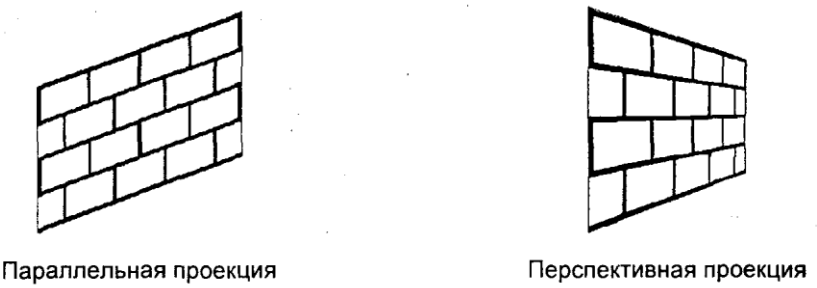


Рис. 3.45. Прямоугольник в разных проекциях

Прямоугольник в параллельной проекции всегда выглядит как параллелограмм, поскольку для этой проекции сохраняется параллельность прямых и отношение длин. В перспективной (центральной) проекции это уже не параллелограмм и не трапеция (в косоугольной — трапеция), поскольку параллельность и отношение длин здесь не сохраняются. А что сохраняется? Как изображать плоские грани?

В предидущих главах были рассматрены проекции на плоскость. Для таких проекций прямые линии остаются прямыми линиями, поэтому грани можно выводить как полигоны.

Здесь уместно вспомнить, как формируется изображение в определенной проекции средствами компьютерной графики. Последовательность преобразований координат выглядит так:

Мировые координаты $x\ y\ z$ → Видовые координаты $X\ Y\ Z$ → Координаты проекции $X_n\ Y_n\ Z_n$ → Экранные координаты $X_э\ Y_э\ Z_э$

Если считать, что точки текстуры должны соответствовать точкам на объекте, то координаты текстуры должны связываться с мировыми координатами. Поскольку для параллельной проекции в цепочке от мировых координат к экранным все преобразования *линейные,* то целиком допустимо связать координаты текстуры с экранными координатами одним аффинным преобразованиям.

Для перспективной проекции так делать нельзя. Преобразование координат из видовых — в координаты плоскости проецирования — *нелинейное.* Поэтому экранные координаты сначала следует преобразовать в такие, которые линейно связаны с мировыми — это могут быть, например, видовые. А потом видовые координаты (или непосредственно мировые) связать с координатами текстуры аффинным преобразованием, используя, например, метод трех точек.

Рассмотрим, как можно выводить в перспективной проекции полигон с текстурой. Будем использовать алгоритм заполнения полигона горизонтальными линиями, раньше уже рассмотренный нами. На рис. 3.46 изображена одна из горизонталей *(AB).* Вершины полигона (1-2-3-4) здесь заданы экранными двумерными координатами. Для краткости изложения будем считать, что экранные координаты совпадают с координатами в плоскости проецирования. В ходе вывода полигона для связи с текстурой будем вычислять видовые координаты произвольной точки *(P)* этого полигона. Для этого будем использовать как базовую такую операцию: по известным видовым координатам концов отрезка находим видовые координаты точки отрезка, заданной координатами в плоскости проецирования.

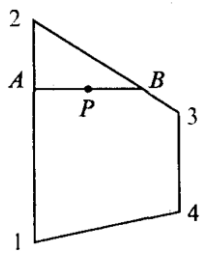


Рис. 3.46. Полигон

Для определения видовых координат *X, Y, Z* точки *A* должны быть известны видовые координаты концов отрезка (1-2). Для видовых координат справедливы соотношения:

$$\frac{X-X_1}{X_2-X_1}=\frac{Y-Y_1}{Y_2-Y_1}=\frac{Z-Z_1}{Z_2-Z_1}.$$

Выберем пропорцию, которая связывает, например, координаты *X* и *Z*. Тогда

$$X = a + Zb,$$

где $a = X_1 - Z_1 (X_2 - X_1)/(Z_2 - Z_1)$, $b = (X_2 - X_1)/(Z_2 - Z_1)$.

Теперь запишем для перспективной проекции соотношения между видовыми координатами произвольной точки и координатой X_n в плоскости проецирования:

$$X_n = X\frac{Z_к - Z_{пл}}{Z_к - Z},$$

где Z_k — это координата камеры (точки схода лучей проецирования), $Z_{пл}$ — координата плоскости проецирования. Перепишем это уравнение так:

$$X = c + Zd,$$

где $c = X_n Z_к / (Z_к - Z_{пл})$, $d = -X_n / (Z_к - Z_{пл})$. Теперь решим систему уравнений:

$$X = a + Zb,$$
$$X = c + Zd.$$

Решением системы будет:

$$Z = \frac{c-a}{b-d},$$

после чего вычисляется *X*, например, по формуле $X = a + Zb$. Для определения координаты *Y* достаточно заменить всюду в формулах *X* на *Y*. Здесь можно заметить, что вычисление видовых координат *(X. Y,* Z) по известным координатам проекции соответствует *обратному проективному преобразованию.*

Найдя видовые координаты точки *A,* мы можем так же вычислить видовые координаты и для точки *B,* лежащей на отрезке (3-4). Аналогично определяются видовые координаты точки *P*.

Следует отметить, что для преобразования координат необходимо вычислять дробно-линейные выражения, что может существенно замедлить цикл визуализации. Поэтому часто используют упрощенные вычисления координат. Также используется комбинирование методов — для небольших треугольных граней (которые являются небольшими в текущем ракурсе обзора) используют только аффинные преобразование, а для больших полигонов выполняют вычисления с учетом перспективы.

Для наложения текстур на поверхности объектов используются и другие преобразования координат.

Если в ходе анимации изменять координаты привязывания текстуры, то можно получить интересные видеоэффекты.

Одна из проблем наложения текстур состоит в том, что преобразования растровых образцов (повороты, изменение размеров и т. п.) приводят к ухудшению качества изображения. Повороты растра прибавляют ступенчатости *{aliasing);* увеличение размеров укрупняет пикселы, а уменьшение размеров растра приводит к потере многих пикселов образца текстуры, появляется *муар.*

Одним из методов улучшения визуализации текстурированных объектов является использование нескольких образцов текстур с разной детализацией — *MIPmaps (MIP* от лат. *Multum In Parvo* — много в одном). Компьютерная система в течение цикла визуализации сама выбирает подходящий образец текстуры с нужной разрешающей способностью.

Для улучшения текстурированных изображений также используют методы фильтрации растров текстур, например, *билинейную* фильтрацию (рис. 3.47).

Билинейная фильтрация выполняется так. Пусть текущий тексел имеет координаты *x, y.* Сначала интерполируются соответственно координате x цвета пар ближайших пикселов *A-B* и *C-D.* Потом выполняется интерполяция полученных значений соответственно координате y.

Также используются другие, более сложные способы фильтрации, например, *трилинейная* и *анизотропная* фильтрация. Трилинейная фильтрация объединяет билинейную фильтрацию и интерполяцию разных уровней текстур MIPmaps. Анизотропная фильтрация — самая совершенная, она учитывает также текущую пространственную ориентацию текстурированных граней.

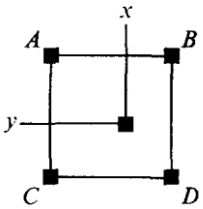


Рис. 3.47. Билинейная фильтрация

При использовании текстур необходим достаточный объем памяти компьютера — количество растровых образцов может достигать десятков, сотен в зависимости от количества типов объектов и многообразия пространственных сцен. Чтобы как можно быстрее создавать изображение, необходимо сохранять текстуры в оперативной памяти, а если это позволяет видеоадаптер — то в памяти видеоадаптера.

Для экономии памяти, которая выделяется для текстур, можно использовать блочное текстурирование. Текстура здесь уже не представляет всю грань целиком, а лишь отдельный фрагмент, который циклически повторяется в грани. Это делается так же, как размножение рисунка кисти при закрашивании полигонов, которые уже рассмотрены нами в этом параграфе. Разумеется, не для всех объектов можно использовать такой способ отображения, однако, например, для образцов массовой "коробочной" архитектуры в этом плане есть практически неограниченные возможности (рис. 3.48).

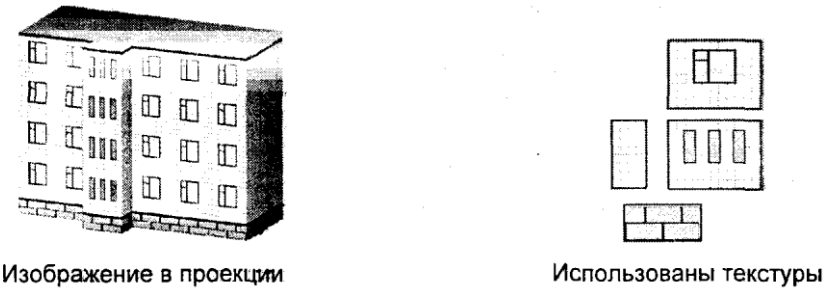


Рис. 3.48. Блочное текстурирование

При формировании растровых изображений объектов главное — для любого пиксела объекта установить нужный цвет. Иногда не очень важно, какой именно метод используется для расчетов цветов. Например, можно моделировать законы оптики и на основе определенных моделей делать расчеты освещения объектов в ходе цикла визуализации. Тем не менее, например, для компьютерной игры это можно осуществить намного более легким способом — просто наложить предварительно изготовленную текстуру. В сцене, которая изображена на рис. 3.49, расположение двух источников света не меняется, поэтому текстура для газона вокруг домика может использоваться для любого ракурса показа.

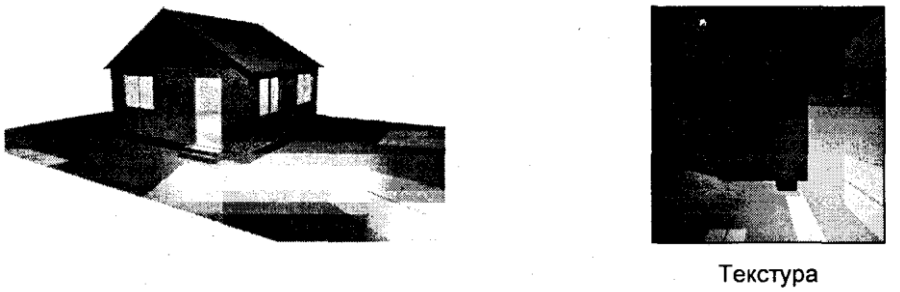


Рис. 3.49. Текстура как карта освещения и теней

Поскольку обычно наложение текстуры осуществляется быстрее создания теней путем геометрических расчетов или трассировки лучей, то это позволяет достичь высокой скорости визуализации. Однако, на этом пути возникают проблемы обеспечения надлежащей разрешающей способности текстуры.

Приведем еще один пример использования текстуры — в качестве карты прозрачности. Цвет любого пиксела текстуры здесь определяет прозрачность соответствующей точки поверхности объекта (рис. 3.50).

Рис. 3.50. Текстура - карта прозрачности для поверхности тора

Можно также использовать текстуру для имитации зеркального отражения, однако, ее корректное использование возможно только для одного ракурса обзора объектов.

Известны также текстуры для имитации рельефа *{Bump Mapping).* Иллюзия искривления поверхности средствами bump-текстурирования достигается изменением направления нормали.

Нормаль к поверхности для каждой точки изменяется соответственно значению тексела. Таким образом можно имитировать, например, волны. Поскольку иллюзия рельефности здесь создается не искривлением поверхности, а только соответствующей локальной подсветкой или затемнением, то возможности отображения рельефа методом *bump*-текстурирования существенно ограничены, в том числе и ракурсом показа. Если смотреть на плоскую грань с торца, то она — плоская, как ее не закрась.

Таким образом, для задания многих свойств поверхности объекта можно использовать одну или несколько текстур. В настоящее время употребляется термин *multitexturing* — многослойное наложение текстур.

Наложение двух и больше текстур предусматривает задание определенной функции комбинирования (смешивания) текселов.

Безусловно, можно предложить и более сложные способы наложения многослойных текстур - надо разрабатывать соответствующие функции комбинирования. Можно считать, что здесь мы снова возвращаемся к аналитическому, процедурному, описанию стиля заполнения - только теперь мы используем в формулах также и текстуры.

Шейдеры

Развитие методов и технологий текстурирования, стремление использовать гибкие процедурные методы закрашивания привели к появлению понятия *шейдер.* Шейдер (от англ. *shade* — затемнять, здесь не путать *shade* с *shadow* — тень, шейдер *затемняет,* а не *затеняет* — для получения изображения теней от объектов в графике используются пока что не шейдеры, а другие средства) — это небольшая программа, которая содержит набор инструкции для расчета цветов пикселов объектов в ходе цикла визуализации. Фактически, шейдеры, немного в другом виде, использовались уже давно в графических

пакетах для создания спецэффектов в кино (например, в пакете Maya). При создании спецэффектов для записи кадров кинофильма не очень нужен режим реального времени, каждый кадр может генерироваться минуту и дольше, главное здесь — это качество изображения.

В 2001-2002 гг. для массового употребления, в первую очередь для компьютерных игр, начали производиться видеоадаптеры (nVidia GeForce3, ATI Radeon 8500), которые аппа-ратно поддерживают базовые операции шейдеров. Разработчики графических программ заинтересовались гибкими возможностями шейдеров, поскольку ощущалась ограниченность фиксированного набора процедур графического конвейера. Возможность программирова-ния графических процессоров дает новый толчок развития средств компьютерной графики.

Современные графические процессоры способны выполнять для любого пиксела десятки и сотни шейдерных команд в реальном времени.

Сейчас существуют два типа шейдеров — вершинные (*vertex shaders)* и пиксельные *(pixel shaders).* Вершинные шейдеры предназначены для вычисления координат вершин полигонов и выполнения расчетов освещения, например, методом Гуро. Их можно считать альтернативой для стандартных возможностей аппаратных средств T&L (*transform and lighting).* Вершинный шейдер может обрабатывать все данные, относящиеся к одной вершине — координаты вершины, координаты нормали, текстурные координаты, цвет, освещение и т. п.

Пиксельные шейдеры появились сначала как способ описания наложения пикселов многослойных текстур. Теперь они содержат разнообразные операции обработки цветов отдельных пикселов.

Шейдеры оперируют такими типами данных — целые числа, числа с плавающей тонкой, тройки RGB, векторы, матрицы. Приведем формат одной команды шейдера для DirectX Graphics

```
op dest, src0, src1, src2
```

где: op — имя команды,

dest — регистр, в который записывается результат,

src0, src1, src2 — регистры входных данных.

Как видим, язык команд шейдеров фактически является ассемблером. Для разработки шейдерных программ можно использовать API DirectX (8. 0 и следующих версий), OpenGL 2. 0 (шейдерные операции частично поддерживаются в расширениях OpenGL версий 1. 2 и 1. 3). Также разработаны специальные языки программирования для шейдеров, например, Cg от nVidia .

Общие сведения о шейдерных возможностях некоторых современных видеоадаптеров предоставлены в таблице 3. 2.

Таблица 3. 2. Характеристики шейдерных возможностей видеоадаптеров

Характеристика	**nVidia NV30**	**ATI Radeon 9700**
Пиксельные шейдеры		
Наложение текстур	До 16	До 16
Текстурные инструкции	До 1024	До 160
Цветовые инструкции	До 1024	До 160
Тип данных	Плавающая точка	Плавающая точка
Максимальная точность	128 бит	128 бит
Вершинные шейдеры		
Инструкции	До 65536	До 1024
Статические инструкции	До 256	До 1024
Регистры	16	16
Циклы	256	Нет

3.5 Инструменты растровых графических пакетов

К фундаментальным инструментам растровой графики относятся такие инструменты обработки изображений, как:

- инструменты выделения;
- каналы и маски;
- инструменты ретуширования;
- гистограммы;
- кривые;
- инструменты для цветовой (цветовой баланс) и тоновой коррекции (уровни);
- фильтры (спецэффекты);
- слои

Инструменты выделения. Каналы и маски

Растровое изображение в отличие от векторного не содержит объектов, которые можно легко «расцепить для выполнения их индивидуального редактирования. Поэтому для создания, например, коллажей (фотомонтажей) из отдельных фрагментов нескольких изображений каждый из них предварительно необходимо выделить. Такая работа, напоминающая вырезание кусков изображений из бумаги ножницами, называется процессом *выделения* (или *обтравки)* изображений. *Выделение* (Selection) — это область, ограниченная замкнутой рамкой выделения в виде движущейся пунктирной линии (контура), которая отмечает часть изображения, доступную для копирования, редактирования и выполнения различных типов преобразований. На жаргоне программистов эта пунктирная линия получила название марширующие муравьи». Она отделяет выделенную область от защищенной области.

Маски — это один из базовых инструментов профессиональных растровых редакторов. В подтверждение этому можно напомнить, что в простейшем растровом редакторе MS Paint, поставляемом в составе ОС Windows, возможность работы с масками отсутствует. В то же время в нем поддерживаются выделения.

Хотя концепции *маски* и *выделения* тесно связаны, понятие маски шире. Всякая маска включает в себя два типа областей: непрозрачные и прозрачные. Первые используются для защиты закрываемых ими частей изображений или объектов от нежелательных изменений. Они, собственно, и выполняют функцию *маскирования.* Прозрачные области можно рассматривать как отверстия в маске. Их используют для выделения фрагментов изображения или объекта, которые собираются модифицировать. Эти области называются *выделенной областью,* или *выделением (обтравкой)*.

Таким образом, маска не есть нечто противоположное выделению. Противоположными свойствами обладают части маски, а именно *защищенные* и *выбранные* (выделенные) области (рис. 6.4). Соотношение между этими частями не является постоянным. В процессе работы над изображением оно может изменяться за счет увеличения доли одной из них и соответственно уменьшения доли другой. Для этой цели в растровых редакторах имеется специальный набор инструментов выделения.

Терминологическая эквилибристика

Свой вклад в неоднозначность терминов «маска» и «выделение» вносят названия родственных инструментов в различных программах растровой графики. Так, в Photoshop эти инструменты называются инструментами выделения, а в Corel PHOTO-PAINT в их названии присутствует слово «маска». Хотя ин том и другом случае они используются для нанесения на изображение выделенных областей. Аналогично команды по работе с масками и выделениями в PHOTO-PAINT представлены в меню Mask, а в Photoshop в меню Select.

Выделение

Под термином выделение (или выделенная область) будем понимать области изображений и объектов, доступные для перемещения, копирования, редактирования и выполнения любых других преобразований. И наоборот, термин маска используется для обозначения областей изображений и объектов, защищенных от применения перечисленных операций.

Понятие маски возникло не на пустом месте. По смыслу и назначению оно близко к понятию трафарета. Представьте себе художника, вырезающего из ватмана трафарет какого-либо слова. Затем он набивает по этому трафарету текст поролоновой губкой, смоченной в краске. При этом часть краски попадает в прорезанные отверстия, а часть остается на трафарете, который и выполняет в данном случае роль защитной маски.

Количество цветовых каналов определяется количеством базовых цветов в используемой цветовой модели. Так, изображение в формате Grayscale имеет один канал, в цветовых моделях RGB и L*a*b — три канала, а в модели CMYK — четыре канала. В растровых редакторах цветовые каналы генерируются автоматически при создании или открытииизображения.

Наряду с цветовыми каналами, число которых жестко определено типом используемой цветовой модели, в растровых редакторах возможно использование дополнительных каналов (альфа-каналов), количество которых ограничено только возможностями вашего компьютера. Эта разновидность каналов широко используется для ретуширования, компоновки и локальной коррекции изображений.

Назначение этого типа каналов тесно связано с понятием маски. Более того, фактически каждый такой канал представляет собой маску. Поэтому создание маски приводит к одновременному созданию альфа-канала, в который помещается «серое» изображениемаски.

Чтобы более четко понять связь этих двух понятий, давайте остановимся на физической природе маски. Как уже отмечалась, внешне маска напоминает трафарет. Если же говорить техническим языком, то маска сама является изображением. Это изображение помещается поверх другого изображения, над фрагментами которого мы собираемся выполнить определенные операции. Для любого пикксела маски значение оттенка серого цвета можно изменять в пределах 256 градаций серого (от 0 до 255). Область маски со значением цвета пикселов, равного 0 (черный), полностью защищает изображение от изменений (собственно, и служит маской). Область, пикселы которой имеют значение 255 (белый), полностью открыта для проведения изменений. Как уже отмечалось, такая область называется выбранной (выделенной).

Частично защищенные пикселы также входят в выбранную область и изображаются оттенками серого. Степень изменений, примененных к выбранной области или ее части, можно задать назначением прозрачности выделения.

Инструменты выделения и маскирования

Современные графические редакторы располагают разнообразными инструментами выделения. По принципу формирования выделенных областей их можно разделить на четыре группы.

Обычные (геометрические), использующие для построения выделений разнообразные геометрические формы: прямоугольную, квадратную, круглую и эллиптическую.

Инструменты выделения от «руки». Типичным примером таких инструментов являются: Лассо в Photoshop и Freehand Mask в Corel PHOTO-PAINT. Они используются для выделения объектов сложной формы путем их обводки.

Инструменты выделения контуров (path tools) похожи на инструменты предыдущей группы. Однако в данном случае выделенные области представляют собой векторные объекты. Благодаря этому такие выделения имеют ряд преимуществ перед обычными (растровыми) выделениями:

- требуют для своего хранения меньший объем памяти;
- предоставляют возможность импорта в векторные программы, такие как Adobe Illustrator, CorelDRAW и Freehand;
- дают возможность масштабирования без потери качества;
- более просты для прецизионного редактирования формы выделения, поскольку состоят из управляющих точек (узлов), которые можно перемещать для настройки нужной формы контура выделения.

Цветочувствительные, в которых выделенная область изображения определяется цветом изображения. В основе работы этих инструментов лежит назначение двух параметров:

- базового цвета, выбираемого щелчком мыши на соответствующей точке изображения;
- диапазона цветов, близких к базовому.

Такое количество инструментов выделения обусловлено разнообразием задач, решаемых при редактировании изображения. В одном случае вам могут понадобиться точные геометрические формы выделения, в другом — прецизионные нерегулярные формы объектов и, наконец, в третьем — области изображения, включающие в себя определенный диапазон цветов, например цвета неба.

С помощью инструментов выделения вы можете создавать два типа выделений.

• *Простые,* реализация которых требует выполнения одной операции.

• *Сложные* выделения строятся на базе двух или более простых выделений.

Ретушь

Традиционно инструменты ретуширования изображений предназначены для восстановления поврежденных изображений, например, для ретуши фотографий.

Ретушь (retouch) — коррекция изображения с целью устранения мелких дефектов, исправления тонального и цветового балансов.

С другой стороны, для дизайнеров в области рекламы и маркетинга основной целью ретуширования является украшение изображения, придание ему большей убедительности. Для реализации этого может потребоваться выполнение двух групп операций:

• устранить детали, мешающие созданию нужного эффекта. Обычно это морщины на лице, блики и мелкие посторонние предметы;

• добавить некоторые детали, чтобы подчеркнуть (усилить) нужный эффект.

Изображения могут иметь царапины, пятна и другие дефекты локального характера. В этом случае процесс ретуширования можно выполнить без применения выделений или масок, используя лишь входящие в состав используемого вами пакета инструментальные средства локального улучшения.

Отметим наиболее часто используемые средства ретуширования: *инструмент клонирования, инструменты размытия, инструменты* Палец *и* Губка, *инструменты* Осветлитель *и* Затемнитель. Они выполняют несколько функций. *Инструменты* клонирования (C1oning Tools) предназначены для копирования деталей из одного места изображения (неповрежденного) в другое (поврежденное). Типичным примером такого инструмента является Штамп. Клонирование рекомендуется применять для удаления дефектов сканирования, следов пыли, царапин пятен путем замены на тона и детали того же или другого изображения, сходного по цвету или более совершенного.

Инструменты размытия (Blur) и повышения резкости (Sharpen) позволяют соответственно локально снижать или усиливать контраст между пикселами изображения. Так, локальное ослабление нежелательных подробностей (морщин, нездорового цвета кожи и т. д.) позволит акцентировать внимание на главных деталях изображения, маскируя второстепенные детали. В то же время локальное увеличение резкости может привлечь внимание к каким-то особенностям изображения (например, блеск драгоценностей), что составляет основу рекламы производимых изделий или имиджа человека, использующего эти изделия.

Инструменты Палец (Smudge) и Губка (Sponge) сглаживают различия между соседними оттенками в тех местах, где проходит кисть. Они применяются для удаления морщин, складок на одежде, случайного шума, наложенного на изображение при сканировании, а также для сглаживания границ между исходными и клонированными с помощью инструмента Штамп (Stamp) участками изображения.

Инструменты Осветлитель (Dodge) и Затемнитель (Burn) делают объекты более светлыми или тусклыми. Эти средства предназначены для коррекции освещенности или изменения значения яркости, чтобы выделить или скрыть отдельные детали.

Хотя большинство фильтров предназначено для применения к изображению специальных эффектов (об этом подробнее будет сказано ниже в соответствующем разделе), некоторые из них могут быть полезными для ретуширования изображений. В большинстве случаев для получения нужного эффекта их следует использовать в совокупности с масками и выделениями.

К наиболее полезным типам фильтров для решения задач ретуширования можно отнести следующие.

• Нерезкое маскирование (Unsharp mask) и группа Фильтры усиления краев (Edge-Sharpening filters). С их помощью можно повышать контраст и подчеркивать детали изображения. Локальное использование их для целей ретуширования позволяет усилить одни детали изображения по сравнению с другими.

• Размытия (Blur) и Смягчения (Soften). Эти группы фильтров позволяют удалять дефекты сканирования и сглаживать второстепенные детали.

• добавление шума (Noise). За счет добавления шума в небольшую выделенную область можно скрыть некоторые дефекты изображения или замаскировать нарушающие гармонию детали изображения.

Гистограммы

Инструмент Гистограмма (Histogram) позволяет оценить разброс между минимальной и максимальной яркостью изображения (динамический диапазон). С его помощью можно получить также наглядное представление о распределении всех тонов в изображении. Поэтому неудивительно, что гистограмма является одним из основных средств, используемых для контроля за тональными и цветовыми настройками изображения. Гистограммой называется график, отображающий распределение пикселов изображения по яркости.

При построении этого графика по оси X откладываются значения яркостей в диапазоне от 0 (черный) до 255 (белый), а по оси Y — количество пикселов, имеющих соответствующее значение яркости.

Человек, впервые сталкивающийся с этим понятием, может усомниться в его важности. Однако после приобретения определенного практического опыта одного взгляда на гистограмму бывает достаточно, чтобы понять, какие тоновые области изображения нуждаются в коррекции и в какой именно.

В том случае, если вы хотите улучшить плохое изображение (или плохо отсканированное изображение), анализ его гистограммы позволит установить, стоит этим заниматься или лучше отсканировать его заново. С помощью гистограммы вы можете также определить пути и способы дальнейшей коррекции изображения.

Если основная масса пикселов изображения смещена к одному из краев графика, вы вряд ли вы сможете спасти изображение с помощью коррекции. И наоборот, достаточно равномерное распределение пикселов вдоль всей оси X позволяете уверенностью сказать, что изображение может быть улучшено.

С помощью гистограммы (рис. 3.51) вы также можете оценить тоновый диапазон изображения, то есть определить, какие тоновые области доминируют: тени (темные области), света (светлые области) или средние тона. Так, изображение, залитое серым однородным тоном, будет характеризоваться гистограммой, содержащей один пик в области средних тонов.

Термины *тени* (shadows), *средние тона* (midtones) и *света* (highlights) используются в графических редакторах для обозначения соответственно темных, средних и светлых тонов изображения.

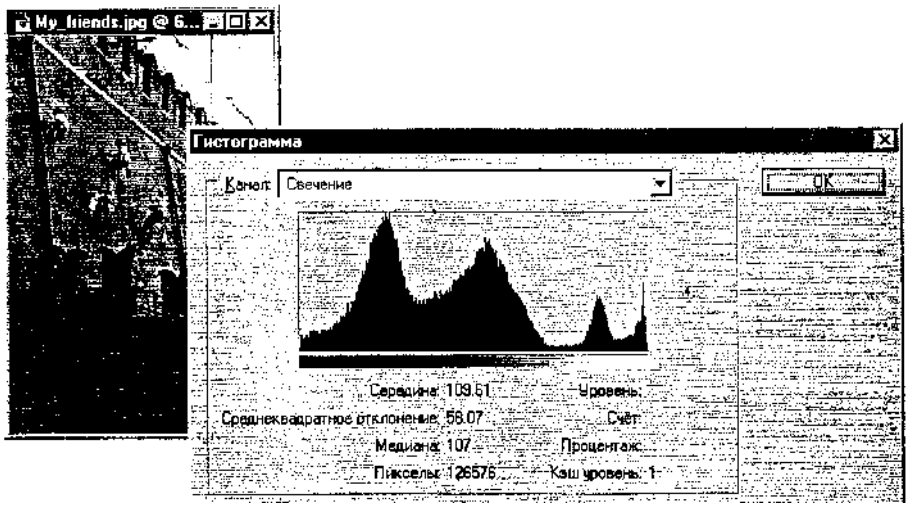


Рис. 3.51. Гистограммы в наглядной форме отображают распределение уровней яркости как для всего изображения, таки в отдельных его каналах. Поэтому они служат отличным средством определения методик тоновой и цветовой коррекции

Можно просматривать гистограммы распределения цветов по отдельным цветовым каналам, данное средство расширяет возможности использования этого инструмента и позволяет выполнять тонкую настройку.

• «Гребенка» с большим количеством пиков и провалов говорит о том, что изображение сильно зашумлено или же уже было подвергнуто компьютерной обработке.

• Пики в непосредственной близости к левому/правому краю гистограммы говорят о том, что на изображении присутствуют большие области черного/белого с потерей деталей в них. Изображений, в которых присутствие таких областей оправдано, не так уж много — например, силуэты. В остальных же случаях присутствие таких областей на изображении является серьезным недостатком и может быть результатом, например, неправильной экспозиции (коррекции не подлежит) или же отсечения светов/теней при сканировании.

Тоновая коррекция изображения

Смысл тоновой коррекции состоит в придании изображению максимального динамического диапазона.

Тон - уровень (градация, оттенок) серого цвета. Тоновое изображение имеет непрерывную шкалу градаций серого от белого до черного. Для одного канала число таких градаций равно 256.

В свою очередь, это напрямую связано с настройкой яркости изображения. Для оценки и коррекции яркости и контрастности изображения (его тоновой коррекции)

профессиональные растровые редакторы предоставляют широкий набор средств, среди которых можно отметить:

• два мощных универсальных инструмента — Уровни (Levels) и Кривые (Curves);

• более простые инструменты, например Яркость/Контраст (Brightness/Contrast), предназначенные для устранения наиболее грубых дефектов типа недостаточной яркости или повышенной контрастности.

Уровни (Levels)

В основе работы данного инструмента лежит использование гистограмм. Однако в отличие от рассмотренной в предыдущем разделе команды Histogram (Гистограмма) здесь этот инструмент выполняет активную функцию, позволяя изменять динамический диапазон изображения.

Прежде чем воспользоваться данным инструментом для тоновой коррекции изображения, познакомимся с предоставляемым им набором элементов настройки. Наряду с гистограммой, выполняющей функции основного индикатора настройки изображения, здесь имеются поля для ввода численных значений параметров и кнопки инструментов.

Остановимся на назначении основных групп параметров. Параметры раздела Входные уровни (Input Levels) используются для установки новых значений черной и белой точек изображения, что позволяет сократить диапазон яркостей изображения и повысить его контрастность. Для этих целей вы можете воспользоваться перемещением находящихся над гистограммой треугольников либо ввести численные значения в соответствующие поля ввода. Например, установка в левом поле значения 30 приведет к тому, что все цвета, имеющие значение яркости меньше этой величины, станут черными, и соответственно ввод значения 220 в правом поле приведет к обращению в максимум всех яркостей в диапазоне 220—255. В результате диапазон яркостей исходного изображения понизится с 255 до 195, а контрастность возрастет.

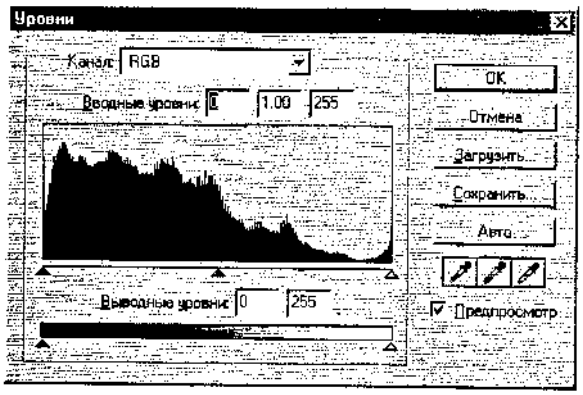


Рис. 3.52. Окно диалога Уровни (Levels)

Следует помнить, что при выполнении такого преобразования информация, содержащаяся в тоновых диапазонах 0 — 30 (светах) и 220 — 255 (тенях), будет потеряна. Однако при неудовлетворительном результате вы всегда можете воспользоваться кнопкой Отмена (Cancel) или экспериментировать с копией исходного изображения.

Между крайними треугольниками, характеризующими значение светов и теней изображения, расположен третий треугольник, который предназначен для управления яркостью в области средних тонов изображения. Этот элемент управления в растровой графике имеет специальное название — *коэффициент гамма, (гамма* — коэффициент контраста в средних тонах изображения), а действия, выполняемые путем перемещения среднего треугольника, называют настройкой гаммы. Установка значения этого параметра меньше 1 (это значение задается по умолчанию) приводит к затемнению изображения, и

наоборот, больше 1 — к осветлению изображения в области средних тонов. В обоих случаях происходит изменение контрастности изображения.

Параметрами раздела Выходные уровни (Output Levels) можно управлять точно так же, как и входными параметрами. Однако в отличие от них, здесь перемещение левого треугольника приводит к осветлению более темных пикселов (теней), и наоборот, перемещение правого треугольника затемняет более светлые пикселы (света). Например, задан в левом поле значение, равное 40, вы настраиваете на эту величину яркость самого темного пиксела, что приводит к повышению уровня освещенности изображения. Аналогичным образом с помощью правого поля ввода вы можете установить новое, более низкое, значение самого светлого пиксела. В итоге это приводит к снижению контрастности изображения.

Нажатие кнопки Авто (Auto) является *альтернативой* выполнения специальной команды Автоуровни (Auto Levels). Оно приводит к запуску процедуры *автоматической тоновой коррекции,* сущность которой состоит в отбрасывании заранее установленного количества самых светлых *и* самых темных пикселов изображения. По умолчанию эти значения равны 5%. Обычно использование этого средства приводит к неудовлетворительным результатам. Но все же попробуйте и будьте готовы нажать кнопкуОтмена.

Сущность белой и черной точек

Белой точкой (White point) называется то место изображения, где оно выглядит очень светлым, но при этом в нем еще можно различить какие-то детали изображения. Белую точку в изображении можно задать искусственно более темной. В этом случае все элементы изображения, более светлые, чем указанные данным инструментом, будут полностью белыми без видимых деталей. Установка нового значения белой точки позволяет для некоторых изображений расширить тональный диапазон без потери деталей в светах и повысить четкость средних тонов. Однако необходимо учесть, что нельзя указывать в качестве белой точки отблески от блестящих предметов (например, от металлических никелированных или хромированных деталей), поскольку в этих частях изображения нет вообще никаких видимых деталей. *Черной точкой* (Black point) называется то место изображения, где оно выглядит очень темным, но при этом в нем еще можно различить какие-то детали изображения. Черную точку в изображении можно задать искусственно более светлой, указан в изображении на более светлое место. В этом случае все элементы изображения, более темные, чем указанные данным инструментом, будут полностью черными без видимых деталей. Установка нового значения черной точки позволяет для некоторых изображений расширить тональный диапазон без потери деталей в тенях и повысить четкость средних тонов.

Кривые

По принципу действия команда Кривые близка к команде Уровни. Только здесь для настройки яркости изображения в окне диалога Кривые (Curves) (рис. 3.53) вместо гистограммы используется инструментальное средство, известное под именем *кривая* (в локализованных версиях растровых редакторов встречаются и другие термины — *настроечная кривая* и *градационная кривая).*

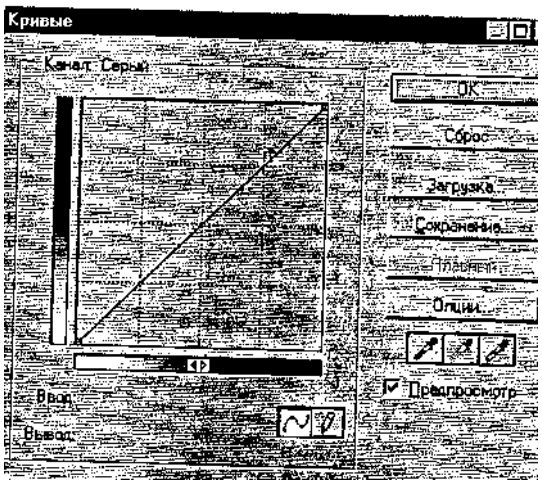


Рис. 3.53. В момент открытия окна диалога Кривые (Curves) его основное инструментальное средство — настроечная кривая — предстает в виде прямой линии с наклоном 45. Это говорит о том, что все входные и выходные пикселы имеют идентичные значения яркости

Кривые уже давно используются в лучших сканерах и графических системах высокого класса для подготовки изображений к печати. В последнее время они активно применяются в профессиональных графических пакетах, являясь одним из самых мощных и тонких средств регулирования тона и цвета изображения.

Кривая (curves) — это график, с помощью которого осуществляется преобразование спектрального диапазона исходного изображения (входные данные) к спектральному диапазону скорректированного изображения (выходные данные). В некоторых источниках это инструментальное средство называют также *яркостная кривая, настроечная кривая и градационная кривая.*

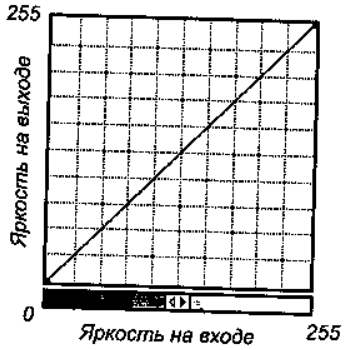


Рис. 3.54. Яркость на входе и яркость на выходе

Иными словами, *кривая* — это инструмент для одновременного изменения контраста во многих яркостных диапазонах изображения. Этим она отличается от гистограмм (и соответственно инструмента Уровни), в которой для настройки яркости используются только три области (света, тени и средние тона). По умолчанию в исходном виде *кривая* представляет собой прямую линию с наклоном в 45°, что соответствует линейному возрастанию уровней яркости от темного тона к светлому. В процессе редактирования кривой вы изменяете конечные (выходные) уровни яркости относительно исходной (входной) прямой линии.

Для RGB-изображений на кривой отображаются значения яркости в диапазоне от 0 до 255 градаций, причем в левой части графика расположены тени (0). В случае использования CMYK-изображений в качестве единиц измерения используются проценты от 0 до 100. В этом случае, наоборот, в левой части графика отображаются света (0). Для

инвертирования расположения на графике областей теней и светов нажмите на кнопку в форме двойной стрелки под кривой.

Устанавливая наклон кривой более 45° (выпуклая кривая), вы расширяете диапазон тонов или цветов, входящих в соответствующие области изображения, делая его контрастнее и детальнее. Наоборот, установка вогнутой кривой приводит к сужению диапазона тонов и, как следствие, — к уменьшению контраста.

Цветовая коррекция и цветовой баланс

В современных настольных издательских системах для получения качественных изображений (таких, как рекламные объявления и обложки журналов) используется технологическая цепочка, включающая сканирование изображения с последующей его цветокоррекцией. Для выполнения операции цветокоррекции может быть использовано программное обеспечение, поставляемое в комплекте со сканером, или растровые графические редакторы.

Цветокоррекция — изменение цветовых параметров пикселов (яркости, контрастности, цветового тона, насыщенности) с целью достижения оптимальных результатов.

К наиболее распространенным средствам, используемым для повышения качества цветных изображений, можно отнести следующие команды:

Цветовой баланс — соотношение цветов в изображении. Регулировка цветового баланса позволяет усилить или ослабить один цвет за счет другого дополнительного (комплементарного ему).

Для того чтобы понять сущность цветового баланса, вернемся к понятию цветового круга. Если вы забыли порядок следования цветов, обратитесь к рис. 3.55.

На цветовом круге каждый цвет имеет противоположный (комплементарный) ему цвет. Перемещаясь по прямой, соединяющей точку с заданным цветом и центр круга, вы попадете в точку с комплиментарным ему цветом. Красный цвет комплементарен голубому, зеленый — пурпурному, желтый — синему. В основе коррекции цвета с помощью команды Баланс цветов (Co1or balance) лежит уменьшение величины избыточной цветовой составляющей за счет усиления ее комплементарного цвета. Увеличение красного цвета приводит к уменьшению голубого, и, наоборот, снижение красного увеличивает содержание в изображении голубого цвета.

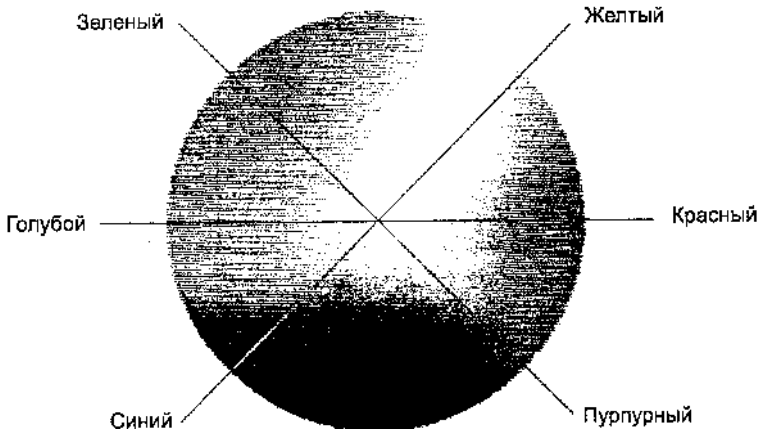


Рис. 3.55. Цветовой круг трудно изобразить с помощью опенков серого. Призовите на помощь ваше воображение

Фильтры (Plug-ins) и спецэффекты (Effects)

Большинство фильтров (filters или plug-ins) предназначено для создания специальных эффектов, например имитации мозаики или живописного стиля Ван-Гога. С

помощью трехмерных спецэффектов двухмерные графические программы способны трансформировать плоское двухмерное изображение в объемное. Группа художественных эффектов позволяет за считанные минуты превратить обычную фотографию в произведение живописи. При этом вы можете имитировать самые разные виды живописи (масло, акварель и т. п.) и стили любых художников.

Фильтры и спецэффекты представляют собой небольшие программы, выполняющие заранее установленную последовательность команд. Они автоматически вычисляют значения и характеристики каждого пиксела изображения и затем модифицируют их в соответствии с новыми значениями. Например, при применении к изображению фильтра Размывка движением (Motion Blur) идет анализ значений всех входящих в изображение пикселов и сдвиг этих значений в определенном направлении для создания иллюзии движения.

Большинство современных графических программ поддерживает возможность применения фильтров, разработанных третьими фирмами под стандарт Adobe Photoshop. Например, Corel PHOTO-PAINT также поддерживает подключаемые модули других фирм и содержит в своем составе фильтры, разработанные для Adobe Photoshop. Эти модули называются подключаемыми (Plug-ins). Их использование расширяет функциональные возможности программы.

Имея под «рукой» подключаемые фильтры, вы можете творить чудеса, диапазон которых ограничен только вашей фантазией и ресурсами используемой вами системы. Для иллюстрации возможностей подключаемых фильтров мы коснемся в этом разделе четырех эффектов, часто применяемых для обработки изображений:

• размывание;

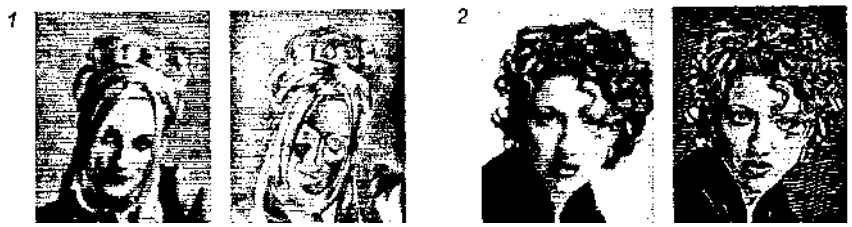


Рис. 3.56. Применеие фильтров

• увеличение резкости;

• тиснение;

• акварельный эффект.

Процесс *размывания* сводится к перераспределению цветов изображения и смягчению резких границ. При увеличении *резкости,* наоборот, подчеркиваются различия между цветами смежных пикселов и выделяются незаметные детали. *Тиснение* преобразует изображение так, что фигуры внутри изображения смотрятся, будто выдавленные на металлической поверхности, как профиль Ленина на лицевой стороне юбилейных рублей. *Акварельный эффект* превращает фотографическое изображение в картинку, как будто бы написанную акварелью.

С алгоритмической точки зрения получение специальных эффектов не представляет особого труда. Секрет каждого из них кроется в крошечной матрице чисел, которую называют *ядром свертки.* Матрица размером 3 х 3 содержит три строки по три числа в каждой. Для преобразования каждого пиксела изображения необходимо выполнить следующие действия:

• Шаг 1. Значение цвета пиксела умножается на число в центре ядра (a22).

• Шаг 2. На следующем шаге выполняется умножение восьми значений цветов пикселов, окружающих центральный пиксел, на соответствующие им коэффициенты ядра с последующим суммированием всех девяти значений. В результате получается новое значение цвета преобразуемого пиксела.

• Шаг 3. для каждого пиксела изображения повторяется процесс, включающий выполнение шагов 1 и 2. данную процедуру принято называть *фильтрацией изображения.*

Коэффициенты ядра свертки определяют результат процесса фильтрации. Их формирование зависит от типа эффекта. Например, *ядро размывания* состоит из совокупности коэффициентов, каждый из которых меньше 1, а их сумма составляет 1. Это означает, что каждый пиксел поглощает что-то из цветов соседей, но полная яркость изображения остается неизменной (если сумма коэффициентов больше 1, яркость увеличится; если меньше 1, яркость уменьшится). В *ядре резкости* центральный коэффициент больше 1, а окружающие его значения являются отрицательными числами, сумма которых на единицу меньше центрального коэффициента. Таким образом достигается увеличение существующего контраста между цветом пиксела и цветами его соседей. Это звучит немного мистически, но цифровое изображение, в конце концов, всего лишь связка чисел. Изменяя эти числа правильным способом, можно прийти к удивительным и, возможно, очень полезным спецэффектам.

Пример. Алгоритм работы фильтра Размывание

1. При подготовке к размыванию цифровое изображение считывается в память компьютера в виде красного, зеленого и синего компонентов цвета каждого пиксела.

2. Ядро размывания размером 3 x 3 применяется к красному, зеленому и синему компонентам цвета каждого пиксела изображения. Значение цвета пиксела (который, собственно, находится в центре ядра) вычисляется умножением соответствующего весового коэффициента на соответствующее ему значение цвета в изображении с последующим суммированием результатов. Итоговое изображение получается размытым по сравнению с оригиналом, потому что цвет каждого пиксела выровнялся (усреднился) благодаря влиянию соседей.

3. Степень размывания можно увеличить одним из трех способов:

• использованием большего размера ядра для распределения цвета среди большего числа соседей (в окне диалога этого фильтра значение размера ядра задается установкой параметра Радиус (Radius) в пикселах);

• подбором коэффициентов ядра и уменьшением влияния центрального коэффициента,

• повторной фильтрацией изображения с тем же ядром размывания.

Пример. Алгоритм работы фильтра Тиснение

В изображении, к которому применен эффект Тиснение, контуры кажутся выдавленными над поверхностью. Тиснение осуществляется почти так же, как размывание и увеличение резкости. Процесс производится над обычным цветным изображением.

1. Каждый пиксел в изображении обрабатывается ядром тиснения размером 3 x 3. В отличие от ядер размывания и резкости, в которых сумма коэффициентов равна 1, сумма весов в ядре тиснения равна 0. Это означает, что «фоновым» пикселам (пикселам, которые не находятся на границах перехода от одного цвета к другому) присваиваются нулевые значения, а не фоновым пикселам — значения, отличные от нуля.

2. После того как значение пиксела обработано ядром тиснения, к нему прибавляется значение 128. В результате фоновые пикселы окрасятся в средний серый цвет: красный — 128, зеленый — 128, синий — 128. Суммы, превышающие *255,* можно

округлить до *255* или взять остаток по модулю 255, чтобы значение оказалось между 0 и *255*.

3. Направление подсветки изображения можно изменять, меняя позиции 1 и — 1 в ядре. Если, например, поменять местами значения 1 и — 1, то направление подсветки инвертируется.

Слои

Слои — один из основных инструментов растровой графики. Что же такое слои? Представьте себе, что у вас на столе несколько кусков оконного стекла (прозрачных полиэтиленовых пленок, бумажных калек), наложенных друг на друга. На каждом стекле вы что-то нарисовали специальным фломастером и теперь смотрите на все это сверху. Считайте, что стекла — это и есть слои.

Слой (layer) — дополнительный уровень (холст) для рисования, метафора прозрачной кальки. Каждый слой сохраняет (повторяет) все параметры основного изображения (размеры, разрешение, цветовую модель, число каналов). Соответственно пропорционально количеству используемых слоев возрастает размер изображения. Так, добавление к фоновому слою нового слоя увеличивает размер файла изображения в два раза, двух слоев — в три раза и т. д.

Слой можно сделать невидимым, то есть вытащить стекло из стопки и убрать. Слои можно поменять местами, и тогда рисунки будут перекрывать друг друга иначе. Можно рисовать только на одном слое, совершенно не затрагивал другие.

Естественно, что если вы закрасите какой-то слой сплошным рисунком без дырок (или плотным сплошным цветом), то не увидите, что нарисовано на нижних слоях. Правда, компьютерная живопись позволяет сделать слой полупрозрачным. Если рисунки состоят в основном из линий и не закрашенных областей, то у вас будет просвечивать стол (фон), на котором лежат ваши стекла (или скатерть на этом столе).

3.6 Преимущества и недостатки растровой графики

Достоинства

Одним из достоинств растровой графики является простота и, как следствие, техническая реализуемость (автоматизация) ввода (оцифровки) изобразительной информации. Существует развитая система внешних устройств ввода изображений (к ним относятся сканеры, видеокамеры, цифровые фотокамеры, графические планшеты).

Растровое изображение имеет преимущества при работе с фотореалистичными объектами, например сценами природы или фотографиями людей. Дело в том, что наш мир создан как растровый. И его объекты трудно представить в векторном, то есть математическом, представлении. Фотореалистичность подразумевает, что в растровой программе можно получать живописные эффекты, например туман или дымку, добиваться тончайшей нюансировки цвета, создавать перспективную глубину и нерезкость, размытость и т. д.

Форматы файлов, предназначенные для сохранения точечных изображений, являются стандартными, поэтому не имеет решающего значения, в каком графическом редакторе создано то или иное изображение.

Недостатки

При первой же вашей попытке что-нибудь нарисовать в программе точечной графики — например, в Photoshop, — она потребует от вас принципиального решения о разрешении (количестве точек на единицу длины) и о глубине цвета (количестве цветовых битов на пиксел). Ничего этого знать в векторной программе не нужно. Объем файла точечной графики однозначно определяется произведением площади изображения на разрешение и на глубину цвета (если они приведены к единой размерности). При этом совершенно не важно, что отображено на фотографии: деревянный одноцветный столб

или коллекция бабочек с обилием цвета и форм. Если три параметра одинаковы, размер файла будет практически одинаковым.

Как только вы попытаетесь отсканировать не очень большую фотографию с максимальными разрешением и глубиной цвета, эта картинка потребует для сохранения столько дискового пространства, что вы схватитесь за голову обеими руками. При попытке слегка повернуть на небольшой угол изображение, например, с четкими тонкими вертикальными линиями, четкие линии превращаются в четкие «ступеньки». Любые трансформации (повороты, масштабирование, наклоны) в точечной графике не бывают без искажений.

Невозможно увеличить изображение для рассмотрения деталей. Поскольку изображение состоит из точек, то увеличение изображения приводит только к тому, что эти точки становятся крупнее. Никаких дополнительных деталей при увеличении растрового изображения рассмотреть не удается. Более того, увеличение точек растра визуально искажает иллюстрацию и делает ее грубой (пикселизация). Текст в растровой графике оказывается проблемой. В большинстве растровых программ вы, как правило, редактируете текст во время его создания, но когда вы щелкаете мышью в каком-либо еще месте на экране, печатный символ закрепляется там, где он был бы нанесен на холст. Если вы хотите отредактировать уже набранный ранее текст, то не можете просто поместить курсор между двумя буквами, удалить одну и начать снова набирать. С этой проблемой вы сразу столкнетесь, например, когда начнете работать в растровом графическом редакторе MS Paint. Кроме того, при большом разрешении файл растрового текста будет огромного размера.

Глава 4. Векторная графика

Изображение, созданное в векторных программах, основывается на математических формулах, а не на координатах пикселов. Поэтому векторные файлы содержат наборы инструкций для построения геометрических объектов — линий, эллипсов, прямоугольников, многоугольников и дуг (рис. 4.1). В соответствии с этим основу векторных изображений составляют разнообразные линии или кривые, называемые *векторами,* или, по-другому, *контурами.* Каждый *контур* представляет собой независимый объект, который можно редактировать: перемещать, масштабировать, изменять. В соответствии с этим векторную графику часто называют также *объектно-ориентированной* графикой.

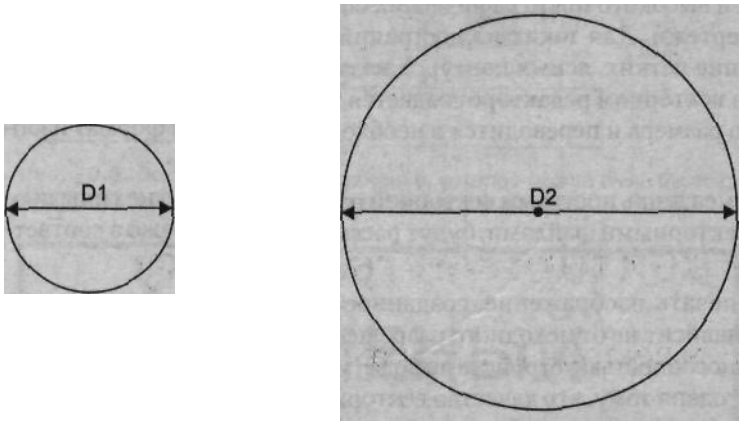


Рис. 4. 1.Примитив векторного файла

Особенности терминологии

Каждый тип компьютерной графики имеет свою терминологию, математический аппарат и характерный для него набор инструментальных средств. Поэтому, несмотря на большое количество представленных на рынке векторных программ, все они в той или иной мере включают в свой состав общий базовый набор инструментальных средств.

В этой главе приведеы с базовые термины и понятия векторной графики, составляющие ядро любой современной векторной программы. Основная сложность в реализации этой задачи - разнообразие терминов, используемых и различных программах для обозначения одних и тех же понятий. Ситуация еще более осложняется при переходе к локализованным версиям оригинальных продуктов. Даже в случае локализации различных версий одной и той же программы ни переводчики, ни редакторы не заботятся о сохранении преемственности в терминологии. Поэтому для профессиональной работы с графикой важно составить представление об особенностях используемой терминологии и базовых примитивах векторной графики.

4.1 Средства создания векторных изображений

Векторные изображения могут быть созданы несколькими видами программ.

- *Программами векторной графики.*
- *Программами САПР,* типичным представителем которых является программа AutoCAD. Ее векторный формат — DXF (Dynamic Exchange Format) понимается многими современными программами.
- Специализированными программами конвертирования растровых изображений в векторные. (CorelTrace 9 и Adobe Streamline).

К векторным объектам относятся также текст и PostScript-контуры вроде тех, которые можно найти также в файлах, созданных с помощью текстовых процессов типа MS Word или программ верстки типа PageMaker.

На платформе Windows наибольшее распространение из программ векторной графики получил редактор

Векторные редакторы и программы САПР являются наилучшим средством для построения шрифтовых и высокоточных графических объектов, таких как, например, конструкторские чертежи. Для таких иллюстраций принципиально важное значение имеет сохранение четких, ясных контуров независимо от размера изображения. Как правило, в векторном редакторе создается заготовка, затем она масштабируется до нужного размера и переводится в необходимый нам формат изображения.

Когда вы выводите на печать изображение, созданное в векторной программе (рис. 4.2), его качество зависит не от исходного разрешения изображения, а определяется разрешающей способностью устройств вывода (монитора, принтера, плоттера и т. п.). Именно благодаря тому, что качество векторного изображения не связано с разрешением, файлы векторных изображений имеют, как правило, меньший объем по сравнению с файлами растровых редакторов.

Рис. 4.2. Увеличение масштаба векторного изображения не приводит к ухудшению его качества (в отличие от растрового изображения)

На рис. 4.3 - 4.4 приведена еще пара примеров, демонстрирующих возможности векторной графики.

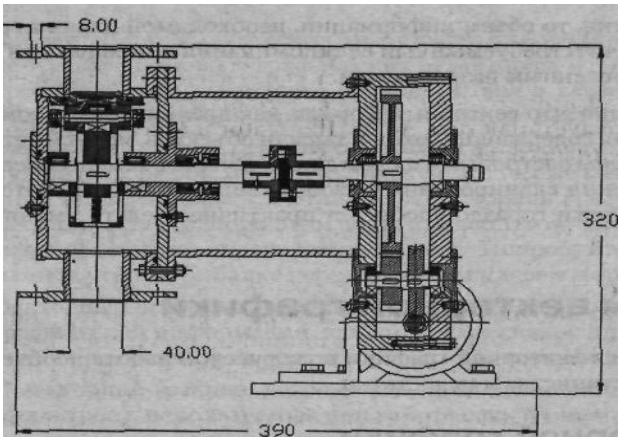


Рис. 4.3. Векторная графика в системах автоматизированного проектирования

Рис. 4.4. Представленные здесь различные гарнитуры векторных шрифтов TrueType (Unicode) имеют расширение TTF

4.2 Сравнение механизмов формирования изображений в растровой и векторной графике

Проиллюстрируем разницу в механизмах работы растровых и векторных редакторов на примере описания одного и того же отрезка прямой:

- в векторном формате — задаются координаты начала и конца прямой, цвет и толщина линии. Для сохранения такой информации на диске потребуется все го несколько байтов памяти;
- в растровом формате — задаются координаты и цвет каждой точки (пиксела), входящей в этот отрезок прямой. А поскольку количество входящих в нее пикселов зависит от разрешения, то объем информации, необходимой для описания отрезка прямой (а значит, требуемый для ее запоминания объем памяти), будет определяться установленным разрешением.

Из приведенного примера видно, что векторный формат, как правило, более компактен (хотя сложные рисунки, содержащие сотни и тысячи объектов, могут иметь размеры, превышающие размеры растровых изображений). Вместе с тем он совершенно не пригоден для хранения сканированных изображений, например фотографий. А вот рисунки и чертежи гораздо удобнее и практичнее делать именно в векторном виде.

4.3 Структура векторной иллюстрации

Структуру любой векторной иллюстрации можно представить в виде иерархического дерева. В такой схеме сама иллюстрация занимает верхний уровень, а ее составные части занимают более низкие уровни иерархии. Для знакомства с основными элементами векторного изображения давайте откроем в одном из векторных редакторов (например, CorelDRAW) любую векторную иллюстрацию и выделим ее составные части, последовательно спускаясь с вершины дерева на его более нижние ветви. Самый верхний иерархический уровень занимает сама иллюстрация, объединяющая в своем составе *объекты* + *узлы* + *линии* + *заливки* (рис. 4.5*)*.

1. Следующий уровень иерархии — *объекты,* представляющие собой разнообразные векторные формы. *В* большинстве редакторов для их отображения необходимо выбрать режим просмотра в виде каркаса.

2. *Объекты* иллюстрации состоят из одного или нескольких *контуров.* На рис. 4.5 приведено отображение основных семи объектов данной иллюстрации. Два из них —цветки — представляют собой объекты, состоящие из нескольких контуров: оболочка цветка *(замкнутый* контур) и тычинка цветка (из четырех линейных отрезков, являющихся *открытыми* контурами). Обычно все объекты в иллюстрации сгруппированы, поэтому для получения доступа к редактированию

отдельных объектов иллюстрации их нужно сначала разгруппировать.

Контуром называется любая геометрическая фигура, созданная с помощью рисующих инструментов векторной программы и представляющая собой очертания того или иного графического объекта.

Типичными примерами контуров могут служить окружность, прямоугольник или другие графические элементы сложного изображения (в том числе и сегмент кривой линии), как, например, фрагменты ветви цветущей сакуры на рис. 4.5.

Замкнутый контур — это замкнутая кривая, у которой начальная и конечная точки совпадают. Примером замкнутого контура является окружность. В некоторых редакторах замкнутый контур называют *фигурой.*

Открытый контур имеет четко обозначенные концевые точки. Синусоидальная линия, например, является открытым контуром.

4. Следующий уровень иерархии составляют *сегменты,* которые выполняют функции кирпичиков, используемых для построения контуров (каждый контур может состоять из одного или нескольких *сегментов).* Начало и конец каждого сегмента называют *узлами,* или *опорными точками,* поскольку они фиксируют положение сегмента, «привязывая» его к определенной позиции в контуре. Перемещение узловых точек приводит к модификации сегментов контура и к изменению его формы. Наряду с узлами в состав сегмента входят также соединяющие узлы линии (прямые или кривые).

Закрытые контуры (формы) имеют свойство заполнения цветом, текстурой или растровым изображением (картой). На рис. 4.5 приведен пример одноцветной заливки измененного контура.

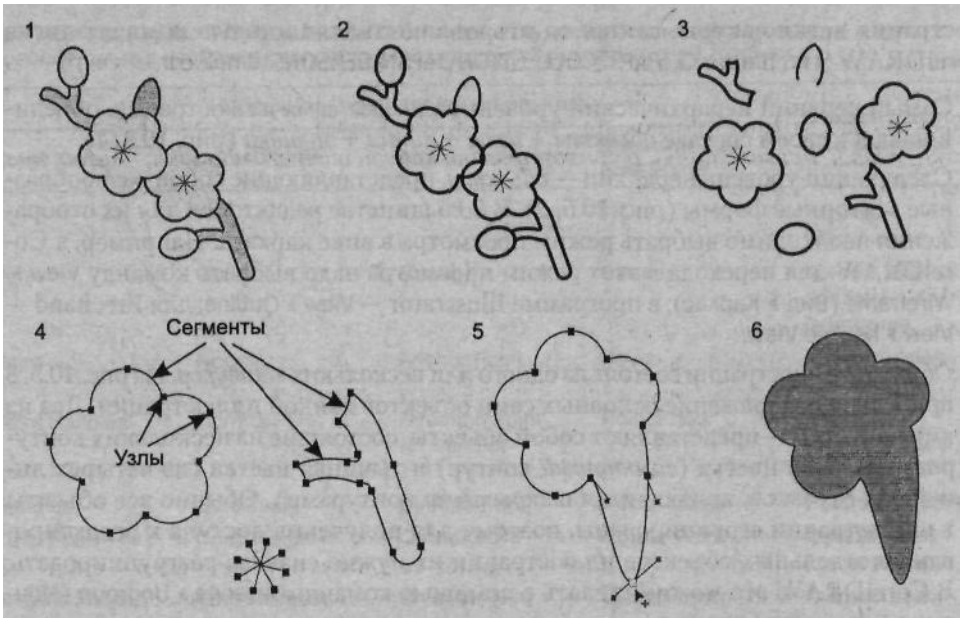


Рис. 4.5. Анатомия векторного рисунка; *1)* исходное векторное изображение (объекты + узлы + линии + заливки); 2) рисунок как совокупность контуров (набор кривых Безье); разгруппированный рисунок в виде набора отдельных контуров (объектов); представление некоторых контуров рисунка а виде сегментов, состоящих из узлов и отрезков кривых; 5) модификация формы контура путем редактирования положения узловых точек и управляющих точек с помощью инструмента Shape (Форма); 6*)* одноцветная заливка измененной оболочки (контура) цветка

Заливка — это цвет или узор, выводимый в замкнутой области, ограниченной кривой.

5. На самом нижнем уровне иерархии расположены *узлы* и *отрезки линий,* соединяющих между собой соседние узлы. Линии наряду с узлами выполняют функции основных

элементов векторного изображения. Существует несколько типов линий и разновидностей узлов. Их названия и функциональное назначение будут рассмотрены ниже.

Простейшая незамкнутая линия имеет две вершины, называемые *узлами* (или *концевыми точками)*. В двухмерной графике узел (точка) задается двумя числами (x, y),

В широком смысле любой из перечисленных здесь элементов иллюстрации, начиная от самой иллюстрации и кончая узлами и линиями, можно трактовать как объект. Поэтому в дальнейшем изложении мы иногда будем использовать термин «объект» в смысле элемента векторного рисунка.

4.4 Математические основы векторной графики

Если основным элементом растровой графики является пиксел (точка), то в случае векторной графики в роли базового элемента выступает линия. Это связано с тем, что в векторной графике любой объект состоит из набора линий, соединенных между собой узлами. Как уже отмечалось в предыдущем разделе, отдельная линия, соединяющая соседние узлы, называется *сегментом* (в геометрии ей соответствует отрезок). Сегмент может быть задан с помощью уравнения прямой или уравнения кривой линии, требующих для своего описания разного количества параметров. Для более полного понимания механизма формирования векторных объектов рассмотрим способы представления основных элементов векторной графики: *точки, прямой линии, отрезка прямой, кривой второго порядка, кривой третьего порядка, кривых Безъе.*

В векторной графике *тачке* соответствует *узел.* На плоскости этот объект представляется двумя числами (*X, Y*), задающими его положение относительно начала координат.

Для описания *прямой линии* используется уравнение $Y = aX + b$. Поэтому для построения данного объекта требуется задание всего двух параметров: *a* и *b*. Результатом будет построение бесконечной прямой в декартовых координатах. В отличие от прямой, *отрезок прямой* требует для своего описания двух дополнительных параметров, соответствующих началу и концу отрезка (например, X_1 и X_2).

К классу *кривых второго порядка* относятся параболы, гиперболы, эллипсы и окружности, то есть все линии, уравнения которых содержат переменные в степени не выше второй. В векторной графике эти кривые используется для построения базовых форм (примитивов) в виде эллипсов и окружностей. Кривые второго порядка не имеют точек перегиба. Используемое для описания этих кривых каноническое уравнение требует для своего задания пяти параметров:

$$x^2 + a1y^2 + a_2xy + a_3x + a_4y + a_5 = 0.$$

Для построения отрезка кривой требуется задать два дополнительных параметра.

В отличие от кривых второго порядка *кривые третьего порядка* могут иметь точку перегиба. Например, график функции $Y \text{ ™ } X^3$ (рис. 10.6) имеет точку перегиба в начале координат (0, 0). Именно эта особенность данного класса функций позволяет использовать их в качестве основных кривых для моделирования различных природных объектов в векторной графике. Следует отметить, что упомянутые ранее прямые и кривые второго порядка являются частным случаем кривых третьего порядка.

Каноническое уравнение, используемое для описания уравнения третьего порядка, требует для своего задания девяти параметров:

$$x^3 + a_1y^3 + a_2\,x^2\,y + a_3xy^2 + a_4x^2 + a_5y^2 + a_6xy + a_7x + a_8y + a_9 = 0.$$

Для описания отрезка кривой третьего порядка требуется на два параметра больше. *Кривые Безъе* — это частный вид кривых третьего порядка, требующий для своего описания меньшего количества параметров — восьми вместо одиннадцати. В основе построения кривых Безье лежит использование двух касательных, проведенных к крайним

точкам отрезка линии (рис. 4.6, справа). На кривизну (форму) линии влияет угол наклона и длина отрезка касательной, значениями которых можно управлять в интерактивном режиме путем перетаскивания их *концевых точек.* Таким образом, касательные выполняют функции виртуальных рычагов, позволяющих управлять формой кривой. Более подробно об этом будет сказано далее в разделе «Кривые Безье».

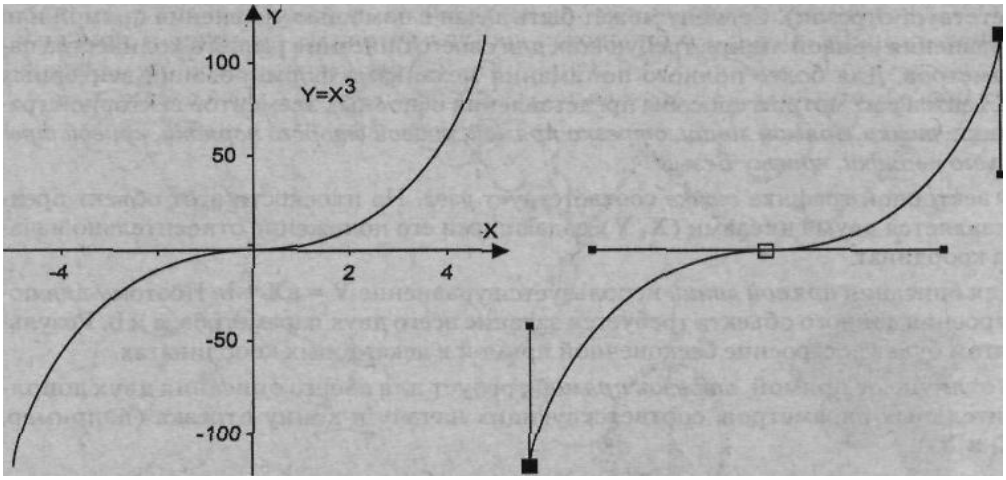


Рис. 4.6. Представление кривой линии с помощью кривых третьего порядка: слева — классический вариант; справа — кривая Беэье

4.5. Элементы (объекты) векторной графики

Теперь давайте познакомимся более детально с основными элементами векторного рисунка, рассмотренными нами ранее в разделе «Структура векторной иллюстрации».

Линии

Как уже отмечалось, в основе векторной графики лежит использование математических представлений о свойствах контуров, основу которых составляет элементарный объект векторной графики *линия.* С ее помощью можно легко построить любой более сложный объект. Например, объект четырехугольник можно создать с помощью четырех линий, а куб — с помощью 12 линий или 6 четырехугольников. Таким образом, иллюстрация составляется из простых объектов, как из кубиков.

Благодаря этому процесс рисования а векторных редакторах фактически сводится к созданию контуров (объектов) нужной формы и приданию им определенных заливок и обводок. Этот принцип лежит в основе всех программ векторной графики. Различаются лишь приемы работы и некоторые специальные эффекты.

В то же время построение линии наряду с использованием для ее описания математического аппарата предполагает задание ряда дополнительных атрибутов, определяющих ее основные свойства: *форму, толщину, цвет, стиль* (сплошная, пунктирная и т. п.). Количество перечисленных атрибутов зависит от вида линии. Открытые линии, например, в отличие от замкнутых не имеют атрибута заливки (рис. 4.7*).* Замкнутые контуры кроме обводки могут иметь определенную пользователем *заливку*).

По умолчанию контуры объектов обычно не имеют толщины. Чтобы контур был виден на экране, ему придают *обводку (абрис)* определенной толщины, стиля (например, сплошная или пунктирная) и цвета. В большинстве редакторов выбор перечисленных атрибутов линии выполняется путем использования специальных библиотек, доступ к которым реализуется с помощью соответствующих окон диалога. На рис. 4.8 - 4,9 приведены подобные окна диалога программы Corel XARA, называемые Галерея линии и Галерея цветов.

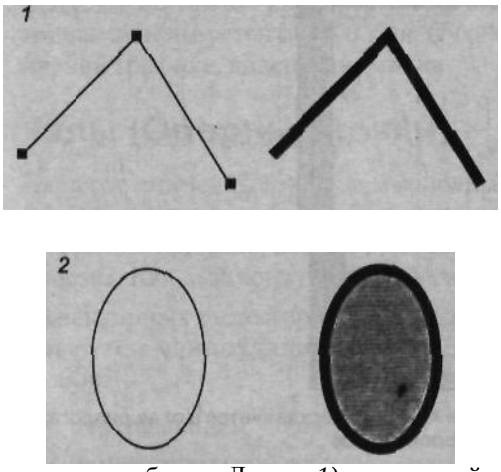


Рис. 4.7. Иллюстрация свойств векторного объекта Линия: *1)* прямолинейный незамкнутый контур (линия), нарисованный в программе Corel XARA 2 инструментом Рисунок от руки (Freehand) при нажатой клавише Alt без атрибута обводки (слева) и с добавлением обводки (справа) толщиной 4 пункта (4pt); *2)* замкнутая линия в виде эллипса без заливки (слева) и с заливкой(справа)

Более подробно цветовые и другие параметры объектов будут рассмотрены далее в разделе «Атрибуты объекта — заливка и обводка».

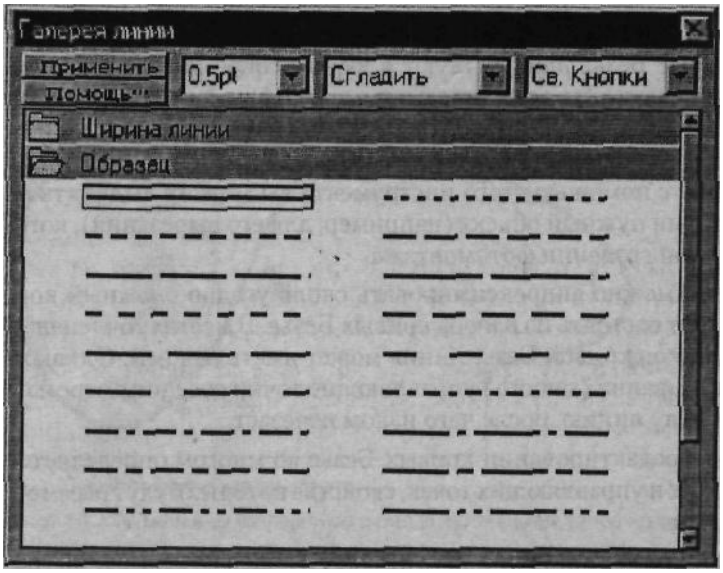


Рис. 4.8. Часть окна диалога Галерей линии Corel XARA с открытой папкой Образец, где можно выбрать любой из имеющихся в ней стилей линии — сплошная, пунктирная и т. п.

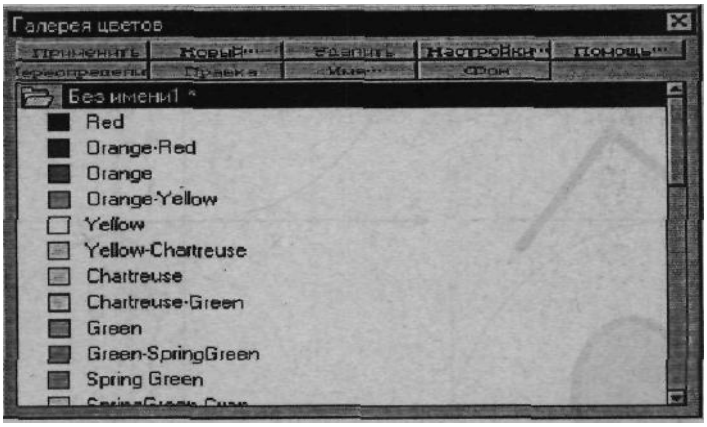


Рис. 4.9. Доступ к цветовым атрибутам линии в Corel XARA предоставляется в окне диалога Галерея цветов

Кривые Безье

В начале 70-х годов профессор Пьер Безье, проектируя на компьютере корпуса автомобилей «Рено», впервые применил для этой цели особый вид кривых, описываемых уравнением третьего порядка, которые впоследствии стали известными под названием *кривые Безье* (функция Bezier).

Поскольку эти линии имеют особое значение как для векторной, так и растровой графики, имеет смысл рассмотреть их более подробно.

В настоящее время кривые Безье присутствуют в любом современном графическом пакете. Достаточно сказать, что все компьютерные шрифты состоят из кривых Безье. Кривые Безье находят также широкое применение и в растровой графике. Так, в программе Photoshop используется термин *контур* (path), базирующийся на кривых Безье. Именно с помощью этого инструмента вы можете выделить на сканированной фотографии нужный объект (например, для его вырезания), который будет использован при создании фотомонтажа.

Отрезками такой кривой можно аппроксимировать сколь угодно сложный контур. В этом случае он будет состоять из набора кривых Безье. В местах сочленений сформированная из отрезков кривой Безье линия может иметь изломы. Однако с помощью функции сглаживания (smooth) управляющие точки соседних отрезков легко выстраиваются в одну линию, после чего излом исчезает. Гибкость в построении и редактировании кривых Безье во многом определяется характеристиками узловых и управляющих точек, свойства которых будут рассмотрены *в* следующем разделе.

Появление кривых Безье вызвало настоящий переворот в видео и трехмерной графике. Это связано с тем, что до появления формул Безье контуры компьютерных персонажей были ломаными, поверхности — гранеными, а движение — прерывистым, скачкообразным, неестественным. Использование кривых Безье позволило реализовать наиболее общий и интуитивно понятный способ управления движением. В соответствии с ним параметрам кривой можно поставить в соответствие параметры движения компьютерного персонажа. В результате движение будет происходить по тем же рассмотренным нами правилам. Таким образом, знаменитая кривая используется не только в двухмерной компьютерной графике, но и в трехмерной графике, видео и анимаций.

Узлы (Опорные точки)

Наряду с линией (line) другим основным элементом векторной графики является *узел* (опорная точка). Как уже отмечалось, линии и узлы используются для построения контуров, которые могут быть представлены в виде прямой, кривой или формы. Каждый контур имеет несколько узлов.

В векторных редакторах (как, впрочем, и в растровых) форму контура изменяют путем манипуляции узлами. Это можно сделать одним из следующих способов:

- перемещением узлов;
- изменением свойств узлов (в том числе атрибутов связанных с ними касательных линий и управляющих точек, рис. 4.10);
- добавлением или удалением узлов.

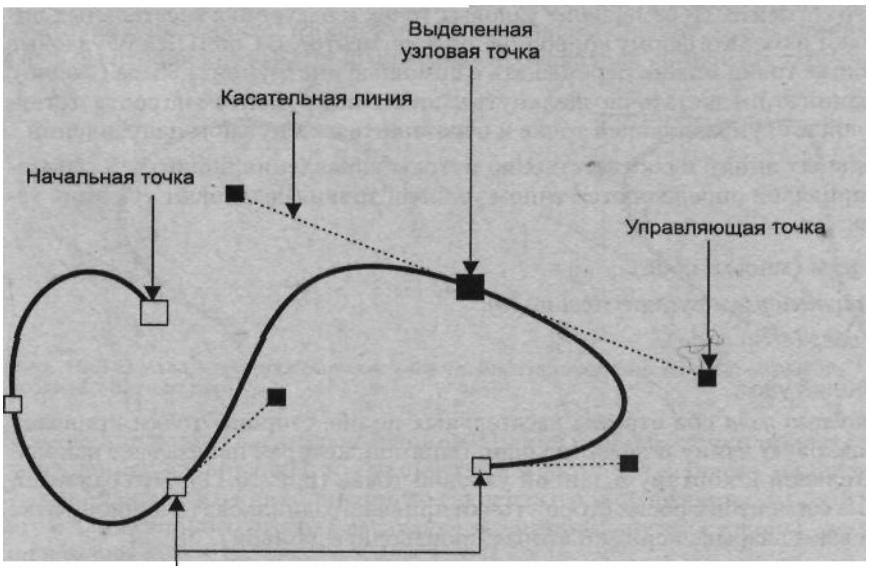


Рис. 4.10. Невыделенные узловые точки

Таким образом, в основе всех процедур, связанных с редактированием (отчасти и созданием) любого типа контуров, лежит работа с узлами. Поэтому прежде, чем приступить к изучению процедур редактирования, давайте сначала познакомимся с типами узловых точек.

Касательные линии и управляющие точки

При выделении узловой точки криволинейного сегмента у нее появляются одна или две *управляющие точки,* соединенные с узловой точкой *касательными линиями, Управляющие точки* изображаются черными закрашенными точками. Расположение касательных линий и управляющих точек определяет длину и форму (кривизну) криволинейного сегмента, а их перемещение приводит к изменению формы контура.

Б локализованных версиях векторных программ наряду с термином *касательные* используются и другие термины: *рычаги управления, направляющие* и *«.усы».* Форма и цвет управляющих точек также зависят от используемого редактора. Если в CorelDRAW они, как и выделенные узлы, обозначаются черными квадратиками, но меньшего размера, то в Corel XARA они закрашены красным цветом, а выделенный узел отображается незакрашенным квадратом.

Типы узловых точек

Касательная линия всегда является касательной к сегменту кривой в узловой точке. Ее наклон и длина определяют наклон и радиус изгиба соответствующего криволинейного сегмента. Перемещение узловых точек и настройка касательных линий позволяют изменять форму криволинейных сегментов. Различают три типа узловых точек:

- *гладкий узел* (smooth node);
- *симметричный узел* (symmetrical node);
- *острый узел* (cusp node).

Симметричный узел

У *симметричного узла* оба отрезка касательных по обе стороны точки привязки имеют одинаковую длину и лежат на одной прямой, которая показывает направление касательной к контуру в данной узловой точке (рис. 4.11). Это означает, что кривизна сегментов с обеих сторон точки привязки одинакова (в данной точке не претерпевают разрыва первая и вторая производные кривой).

Изменение положения управляющей точки приводит к соответствующему изменению угла наклона касательной к кривой. Изменение длины касательной линии с одной стороны точки привязки путем перемещения управляющей точки приводит к

соответствующему изменению и второй касательной линии, что изменяет радиус кривизны линии в точке привязки.

В CorelDRAW симметричные узлы создаются автоматически при рисовании кривых инструментом Bezier (Кривая Безье) методом перетаскивания. Поскольку этот тип узлов является частным случаем гладких узлов в большинстве программ векторной графики (например, в Corel Хата), он не выделен в виде самостоятельного типа узла.

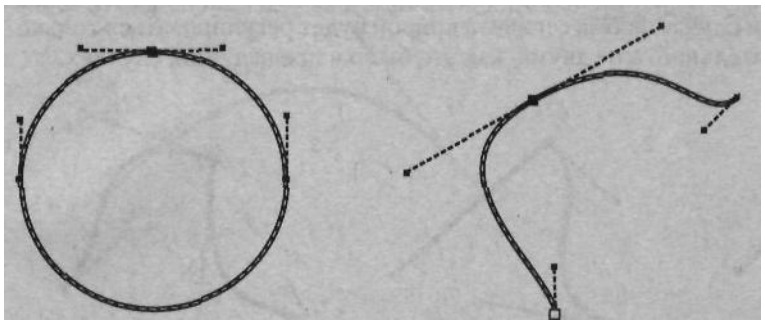


Рис. 4.11. У симметричной узловой точки длина обоих отрезков касательных одинакова, и они лежат на одной прямой

Гладкий узел

У гадкой узловой точки оба отрезка касательных линий по обе стороны точки привязки лежат на одной прямой, которая показывает направление касательной к кривой в данной точке, но длина управляющих линий разная (рис. 4.12). Это говорит о том, что кривизна криволинейных участков, прилегающих к этой опорной точке, различна с разных ее сторон. Математически это значит, что в данной точке нет разрыва первой производной, но вторая производная кривой претерпевает разрыв.

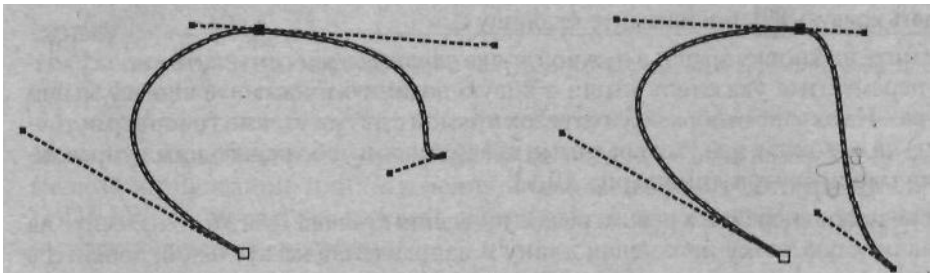


Рис. 4.12. У гладкой узловой точки касательные линии лежат на одной прямой. но имеют разную длину

Изменение длины касательной линии с одной стороны точки привязки путем перемещения управляющей точки приводит к соответствующему изменению радиуса кривизны этого криволинейного сегмента с одной стороны узловой точки. При этом длина второго отрезка касательной линии (с другой стороны узловой точки) не изменяется.

Острый узел

У *острого узла* касательные линии с разных сторон этой точки не лежат на одной прямой. Поэтому два криволинейных сегмента, прилегающих к опорной точке, имеют различную кривизну с разных сторон узловой точки и контур в этой точке образует резкий излом (рис. 4.13). Здесь радиус кривизны и угол наклона касательной для каждого криволинейного сегмента можно регулировать независимо друг от друга соответствующим изменением длины и угла наклона касательной линии для каждого прилегающего к опорной точке криволинейного сегмента в отдельности. В частности, один из отрезков касательных может быть равен нулю (рис. 4.14). В этом случае форма сегмента кривой будет регулироваться только одним отрезком касательной, а не двумя, как это было в предыдущих случаях.

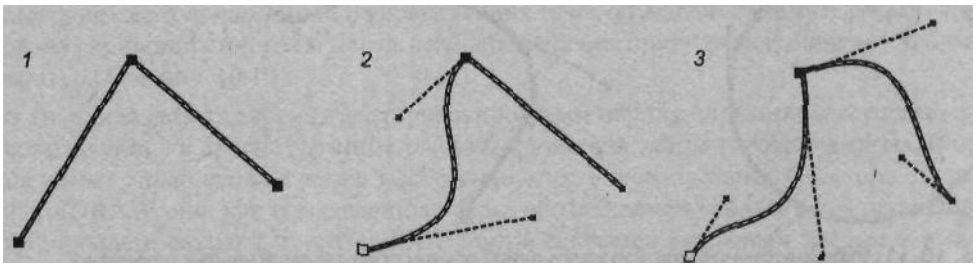


Рис. 4.13. Три варианта острых узлов: без управляющих точек (1), с одной управляющей точкой *(2}* и двумя (3). В последнем случае кривизну сегментов контура в острой узловой точке можно изменять независимо для каждого сегмента

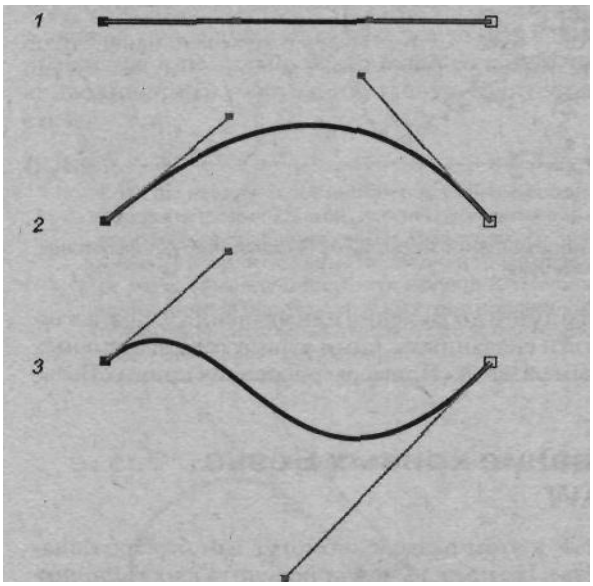


Рис. 4.14. Иллюстрация рисования (?) и редактирования *(2, 3)* кривой Безье в программе Corel KARA 2 с помощью инструмента Shape Editor (Редактирования фигур)

Примитивы (Формы)

Наряду с разнообразными кривыми, основу которых составляют кривые Безье, векторные редакторы имеют в своем составе специальные инструментальные средства для создания простых форм *(графических примитивов),* что упрощает построение сложных объектов. В качестве примера такого примитива можно указать на эллипсовидные формы, используемые при рисовании почек на ветке сакуры (рис. 4.5*)*.

Часто наряду со своим прямым назначением простые формы используются в качестве исходных заготовок для создания на их базе более сложных объектов. В этом случае для последующего редактирования созданных заготовок необходимо привлечение рассмотренной нами ранее технологии редактирования кривых Безье *с* помощью перемещения узлов и управляющих точек. Однако все не так однозначно. Для осуществления этой процедуры в некоторых редакторах требуется выполнить специальное преобразование примитивов в кривые Безье, поскольку их математическое описание в некоторых редакторах отличается от формул, заложенных в построение кривых Безье.

Таблица 10.5 (*продолжение*)

Типы углов (стыков) *сегментов*	Название				Назначение
	CorelDRAW 9	Corel Xara 2	Adobe Illustrator 9	Free Hand 9	
	Beveled (Срезанные)	Bevel join (Срезанные стыки)	Bevel join (Срезанный стык)	Bevel (Срез)	Срезает углы в точках соединения сегментов контуров. Обычно этот вариант принимается по умолчанию
	Rounded (Скругленные)	Round join (Круглый стык)	Round join (Скругленный стык)	Round (Круглый)	Создает стыки соединений сегментов округлой формы

Рис. 4.15. Палитра Swatches (Каталог) предназначена для выбора цвета обводки в Adobe Illustrator 9

Комбинированные объекты

Как вы могли уже убедиться при знакомстве с разделом «Структура векторного рисунка», векторное изображение может состоять из десятков и сотен объектов (контуров). Все они сначала создаются как простые объекты, из которых затем формируется сложный объект. Достигнутый в результате этих действий результат необходимо зафиксировать, чтобы избежать при выполнении последующих операций искажения рисунка из-за возможного изменения соотношения пропорций между объектами или их взаимного расположения. Для этих целей в векторных редакторах предусмотрена группа базовых операций, включающих:

- группировку объектов;
- объединение объектов;
- использование составных контуров.

Группировка объектов

Операция группировки состоит в объединении двух или более объектов (контуров) в одну группу. С полученным таким образом сгруппированным объектом можно обращаться как с единым объектом. Его можно перемешать, поворачивать, растягивать и выполнять многие другие операции без искажения взаимного расположения и пропорций входящих в него объектов.

При реализации операции группировки можно использовать несколько уровней группировки. В этом случае разгруппировка объектов происходит в обратном порядке с сохранением иерархии группировки.

Объединение объектов

Объединенный объект (контур) создается путем использования одной или нескольких операций по объединению двух или нескольких контуров. В результате такой операции из нескольких объектов получается новый объект, обладающий свойствами самого верхнего из исходных объектов, участвующих в операции. Поэтому в отличие от рассмотренной ранее операции группировки здесь свойства составляющих объектов теряются.

В современных векторных редакторах предусмотрены различные варианты слияния объектов. Наиболее распространенными из них являются три процедуры, принцип действия которых основан на использовании базовых логических операций ИЛИ, И, И-НЕ.

4.6. Достоинства и недостатки векторной графики

Для эффективного применения векторной графики в творческой работе необходимо представлять себе ее достоинства и недостатки.

Достоинства

Одним из главных достоинств этого вида графики является возможность неограниченного масштабирования изображения без потери качества и практически без увеличения размеров исходного файла. Это связано с тем, что векторная графика содержит только описания объектов, формирующих изображения, а компьютер или устройство печати интерпретирует их необходимым образом.

Векторную графику значительно легче редактировать, поскольку готовое изображение не является «плоской» картинкой из пикселов, а составлено из объектов, которые могут накладываться друг на друга, перекрываться, оставаясь в то же время совершенно независимыми друг от друга.

Векторным программам свойственна высокая точность рисования (до сотой доли микрона).

Векторная графика экономна в плане объемов дискового пространства, необходимого для хранения изображений. Это связано с тем, что сохраняется не само изображение, а только некоторые основные данные (математическая формула объекта), используя которые программа всякий раз воссоздает изображение заново. Описание цветовых характеристик почти не увеличивает размер векторного файла.

Векторные изображения, как правило, занимают меньший объем памяти компьютера по сравнению с растровыми. Гораздо проще описать окружность радиусом 10 и центром в точке x = 20, y = 30, чем помнить все пикселы массива, соответствующего этой окружности.

Для векторных редакторов характерно прекрасное качество печати рисунков и отсутствие проблем с экспортом векторного изображения в растровое.

Недостатки

Практически невозможно осуществить экспорт изображения из растрового формата в векторный. Попробуйте, например, отсканировать герб России, а затем вырезать его на плоттере. И наоборот, обратное преобразование (то есть превращение векторного изображения в растровое) выполняется практически автоматически не только с помощью графических редакторов, но и буфера обмена Windows.

Векторная графика ограничена в чисто живописных средствах и не позволяет получать фотореалистичные изображения с тем же качеством, что и растровая. Причина в том, что здесь, в отличие от растровой графики, минимальной областью, закрашиваемой однородным цветом, является не один пиксел, а один объект. А размеры объекта по определению больше.

Векторный принцип описания изображения не позволяет автоматизировать ввод графической информации, как это делает сканер для растровой графики. К сожалению, не существует, например, векторных мониторов или векторных сканеров.

В векторной графике невозможно применение обширной библиотеки эффектов (фильтров), используемых при работе с растровыми изображениями.

Строго говоря, ни один современный профессиональный графический пакет не является чисто векторным или чисто растровым, а совмещает в себе элементы как того, так и другого вида графики. Например, векторный редактор CorelDRAW имеет как собственные, так и подключаемые (plug-ins) инструменты для редактирования растровых изображений, а в шестой версии растрового редактора Photoshop расширены инструментальные возможности для работы с векторными объектами.

Глава 5. Фрактальная графика

Понятия фракталы, фрактальная геометрия и фрактальная графика, появившиеся в конце 70-х, сегодня прочно вошли в обиход математиков и компьютерных художников. Слово фрактал образовано от латинского fractus и в переводе означает “состояние из фрагментов”. Оно было предложено математиком Бенуа Мандельбромом в 1975 году для обозначения нерегулярных, но самоподобных структур, которыми он занимался. Рождение фрактальной геометрии принято связывать с выходом в 1977 году книги Мандельброта “The Fractal Geometry of Nature”. В его работе использованы научные результаты других ученых, работавших в 1875-1925 годах в той же области (Пуанкаре, Фату, Жюлиа, Кантор, Хаусдорф). Но только в наше время удалось объединить их работы в единую систему.

Из всех типов фракталов наиболее наглядными являются геометрические фракталы. В двухмерном случае их получают с помощью некоторой ломаной (или поверхности в трехмерном случае), называется генератором. За один шаг алгоритма каждый из отрезков, составляющих ломанную, заменяется на ломаную-генератор в соответствующем масштабе. В результате бесконечного повторения этой процедуры получается геометрический фрактал.

Одним из основных свойств фракталов является самоподобие. Объект называют самоподобным, когда увеличенные части объекта походят на сам объект и друг на друга.

Перефразируя это определение, можно сказать, что в простейшем случае небольшая часть фрактала содержит информацию обо всем фрактале. Например, снежинка несет информацию о снежном сугробе, а горный камень имеет те же самые очертания, и что и горный хребет. Благодаря тому свойству мы можем использовать фракталы для генерирования поверхности местности, которая походит на саму себя, независимо от масштаба, в котором она отображена. Эта идея нашла использование в компьютерной графике благодаря компактности математического аппарата, необходимого для ее реализации. Так, с помощью некоторых математических коэффициентов можно задать линии и поверхности очень сложной формы.

Сегодня разработаны алгоритмы синтеза коэффициентов фрактала, использующего произвести копию любой картинки сколь угодно близкой к исходному оригиналу. С точки зрения машинной графики фрактальная геометрия незаменима при генерации искусственных облаков, гор, поверхности моря. Фактически благодаря фрактальной геометрии найден способ эффективной реализации сложных неевклидовых объектов, образы которых весьма похожи на природные.

Геометрические фракталы на экране компьютера – это узоры, построенные самим компьютером по заданной программе. Они очень красивы, необычны и интересны. Многие художники на Западе (например, Мелиса, Бинде) рассматривают фракталы как новый вид компьютерного искусства. Помимо фрактальной живописи существуют фрактальная анимация и фрактальная музыка.

Создатель фракталов – это художник, скульптор, фотограф, изобретатель и ученый в одном лице. Вы сами задаете форму рисунка математической формулой, исследуете сходимость процесса, варьируя его параметры, выбираете вид изображения и палитру цветов, то есть творите рисунок «с нуля». В этом одно из отличий фрактальных графических редакторов (и в частности — Painter) от прочих графических программ. Например, в Adobe Photoshop изображение, как правило, «с нуля» не создается, а только обрабатывается. Другой самобытной особенностью фрактального графического редактора Painter (как и прочих фрактальных программ, например Art Dabbler) является то, что реальный художник, работающий без компьютера, никогда не достигнет с помощью кисти, карандаша и пера тех возможностей, которые заложены в Painter программистами.

5.1 Математика фракталов. Алгоритмы фрактального сжатия изображений

У фрактальной математики возникают все новые и новые сферы применения. Коснемся лишь одного перспективного направления — создания алгоритма фрактального сжатия графической информации. В 1991 году такой алгоритм был найден. Он имеет ряд уникальных возможностей. Фрактальный архиватор позволяет, например, при распаковке произвольно менять разрешение изображения без появления эффекта зернистости. Более того, он распаковывает гораздо быстрее, чем ближайший конкурент, JPEG, и не только статическую графику, но и видео. В 1992 году компания Microsoft использовала фрактальный архиватор и выпустила компакт-диск Microsoft Encarta мультимедиа-энциклопедия, содержащий информацию о животных, цветах, деревьях и живописных местах. На диск было записано 7 часов звука, 100 анимационных роликов, примерно 800 масштабируемых карт, а также 7000 качественных фотографий. И все это — на одном диске! Напомним, что обычный компакт-диск в 650 Мбайт без использования компрессии может содержать либо 56 минут качественного звука, либо 1 час видео с разрешением 320 x 200 в формате MPEG-1, либо 700 полноцветных изображений размером 640 x 480.

В настоящее время алгоритмы, используемые для генерации изображений фрактальной графики, находят применение и в традиционных видах компьютерной графики: растровой и векторной. Например, в CorelDRAW эти алгоритмы используются для создания текстурных заливок. В недавно появившейся на рынке ПО растровой программе PhotoDraw 2000 фирмы Microsoft кроме стандартных градиентных заливок контуров можно сгенерировать фрактальный узор и воспользоваться им в качестве заливки.

Фрактал можно определить как объект довольно сложной формы, которая получена в результате выполнения простого итерационного цикла. Итерационность, рекурсивность процедуры создания обуславливают такие свойства фракталов, как *самоподобие* — отдельные части похожи по форме на весь фрактал в целом. Латинское *fractus* означает "составлен из фрагментов". В 1975 году французский математик *Бенуа Мандельброт* издал книгу "The fractal Geometry of Nature" . Слово "фрактал" стало модным, и остается таковым и поныне.

Фракталом Мандельброта названа фигура, которая порождается очень простым циклом. Для создания этого фрактала необходимо для каждой точки изображения выполнить цикл итераций в соответствии с формулой:

$$z_{k+1} = z_k^2 + z_0,$$

где k = 0, 1,..., *n.* Величины z_k — это комплексные числа, $z_k = x_k + 1y_k$, причем стартовые значения x_0 и y_0 — это координаты точки изображения. Для каждой точки изображения итерации выполняются ограниченное количество раз *(n)* или до тех пор, пока модуль числа z_k не превышает 2. Модуль комплексного числа равняется корню квадратному из $x^2 + y^2$. Для вычисления квадрата величины z_k можно воспользоваться формулой $z^2 = (x + iy)\ (x + iy) = x^2 - y^2 + i2xy$, поскольку $i^2 = -1$. Цикл итераций для фрактала Мандельброта можно выполнять в диапазоне x = (от -2. 2, до 1), y = (от -1. 2 до 1. 2). Для того чтобы получить изображение в растре, необходимо пересчитывать координаты этого диапазона в пиксельные (рис. 5.1).

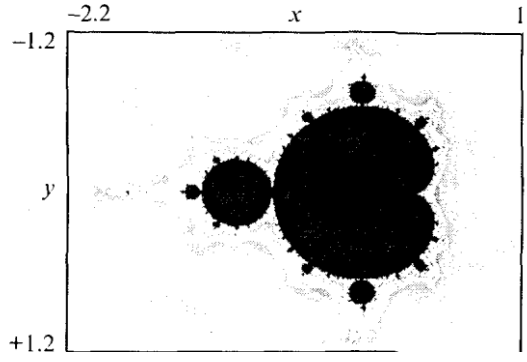


Рис.5.1. Фрактал Мандельброта

Фрактал Жулиа совсем не похож на фрактал Мандельброта, однако, он определяется итерационным циклом, почти полностью тождественным циклу генерации Мандельброта. Формула итераций для фрактала Жулиа такова:

$$z_{k+1} = z_k^2 + c,$$

где c — комплексная константа.

Условием завершения итераций является $| z_k | > 2$ — так же, как для фрактала Мандельброта.

Как видим, фрактал самоподобный — при любом увеличении отдельные части напоминают формы целого. Самоподобие считается важным свойством фракталов. Это отличает их от других типов объектов сложной формы.

Рассмотрим следующий пример фрактала — ***фрактал Ньютон.*** Для него итерационная формула имеет такой вид:

$$z_{k+1} = \frac{3z_k^4 + 1}{4z_k^3},$$

где z — также комплексные числа, причем $z_0 = x + iy$ соответствует координатам точки изображения.

Условием прекращения цикла итераций для фрактала Ньютон есть приближение значений $|x^4-1|$ к нулю.

Рассмотрим еще одну разновидность фракталов, названных **геометрическими,** поскольку их форма может быть описана как последовательность простых геометрических операций. Например, *кривая Кох* становится фракталом в результате бесконечного количества итераций, в ходе которых выполняется деление каждого отрезка прямой на три части. На рис. 5.2 показаны три итерации — постепенно линия становится похожей на снежинку.

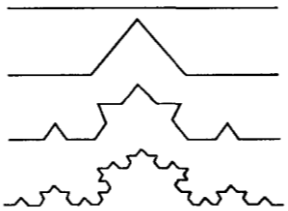


Рис. 5.2. Геометрические интерпретации для кривой Кох

Следующую группу составляют фракталы, которые генерируются согласно методу "систем итеративных функций" — IFS (Iterated Functions Systems). Этот метод может быть описан, как последовательный итеративный расчет координат новых точек в пространстве:

$$x_{k+1} = F_x(x_k, y_k),$$
$$y_{k+1} = F_y(x_k, y_k),$$

где F_x и F_y — функции преобразования координат, например, аффинного преобразования. Эти функции и обуславливают форму фрактала. В случае аффинного преобразования необходимо найти соответствующие числовые значения коэффициентов.

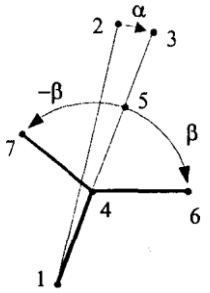


Рис. 5.3. Опорные точки элементов фрактала

Давайте попробуем разработать фрактал, который выглядел бы, как растение. Вообразим ствол, на котором много веточек. На каждой веточке много меньших веточек и так далее. Наименьшие ветви можно считать листвой или колючками. Все элементы будем рисовать отрезками прямой. Каждый отрезок будет определяться двумя конечными точками.

Для начала итераций необходимо задать стартовые координаты концов отрезка. Это будут точки 1, 2. На каждом шаге итераций будем рассчитывать координаты других точек.

Сначала находим точку 3. Это повернутая на угол α точка 2, центр поворота— в точке 1 (рис. 5.3):

$$x3 = (x2 - x1)\cos\alpha - (y2 - y1)\sin\alpha + x1,$$
$$y3 = (x2 - x1)\sin\alpha + (y2 - y1)\cos\alpha + y1.$$

Если α = О, то ствол и все ветви прямые. Потом находим точку 4. От нее будут распространяться ветви. Пусть соотношение длин отрезков 1-4 и 1-3 равняется *к,* причем О < А: < 1. Тогда для вычисления координат точки 4 можно воспользоваться такими формулами:

$$x4 = x1\ (1 - k) + x3\ k,$$
$$y4 = y1\ (1 - k) + y3\ k.$$

Теперь зададим длину и угол наклона ветвей, которые растут из точки 4. Сначала найдем координаты точки 5. Введем еще один параметр — *к1,* который будет определять соотношение длин отрезков 4-5 и 4-3, причем 0 < *kl* < 1. Координаты точки 5 равняются

$$x5 = x4\ (1 - k1) + x3\ k1,$$
$$y5 = y4\ (1 - k1) + y3\ k1.$$

Точки 6 и 7 — это точка 5, но повернутая относительно точки 4 на углы β и -β соответственно:

$$x6 = (x5 - x4)\cos\beta - (y5 - y4)\sin\beta + x4,$$
$$y6 = (x5 - x4)\sin\beta + (y5 - y4)\cos\beta + y4,$$
$$x7 = (x5 - x4)\cos\beta + (y5 - y4)\sin\beta + y4.$$
$$y7 = -(x5 - x4)\sin\beta + (y5 - y4)\cos\beta + y4,$$

Кроме расчета опорных точек, на каждом шаге будем рисовать один отрезок 1 -4. В зависимости от номера итераций можно изменять цвет отрезка. Также можно устанавливать его толщину, например, пропорционально длине.

Таким образом, фрактал мы определили как последовательность аффинных преобразований координат точек. Величины а, β, *к, k1* — это параметры, которые описывают вид фрактала в целом. Они являются константами на протяжении всего итеративного процесса. Это дает возможность в итерациях использовать только операции сложения, вычитания и умножения, если вычислить значения sin(), cos(), *(1 - к) и (1 - k1)* только один раз перед началом итераций как коэффициенты-константы.

Для того чтобы нарисовать фрактал, необходимо вызвать процедуру ШАГ, установив соответствующие значения ее аргументов: **ШАГ(x1, y1, x2, y2, 0).** Обратите внимание на один из аргументов этой процедуры — *num,* который сначала имеет значение 0. В теле процедуры есть три рекурсивных вызова с разными значениями этого аргумента:

- **ШАГ(x4, y4, x3, y3, num)**—продолжаем СТВОЛ;
- **ШАГ(x4, y4, x6, y6, num+l)**—правая ветвь;
- **ШАГ(x4, y4, x7, y7, num+1)** —левая ветвь.

Значение *num* показывает степень детализации расчета дерева. Один цикл итераций содержит много шагов, которые соответствуют одному значению величины *num.* Числовое значение *num* можно использовать для прекращения итеративного процесса, а также для определения текущего цвета элементов "растения".

Завершение циклов итераций в нашем алгоритме происходит тогда, когда длина ветви становится меньше некоторой величины l_{min}, например, $l_{min} = 1$.

Этот фрактал при $\alpha = 2°$, $\beta = 86°$, $\kappa = 0.14$, $k1 = 0.3$ похож на папоротник (рис. 5.4).

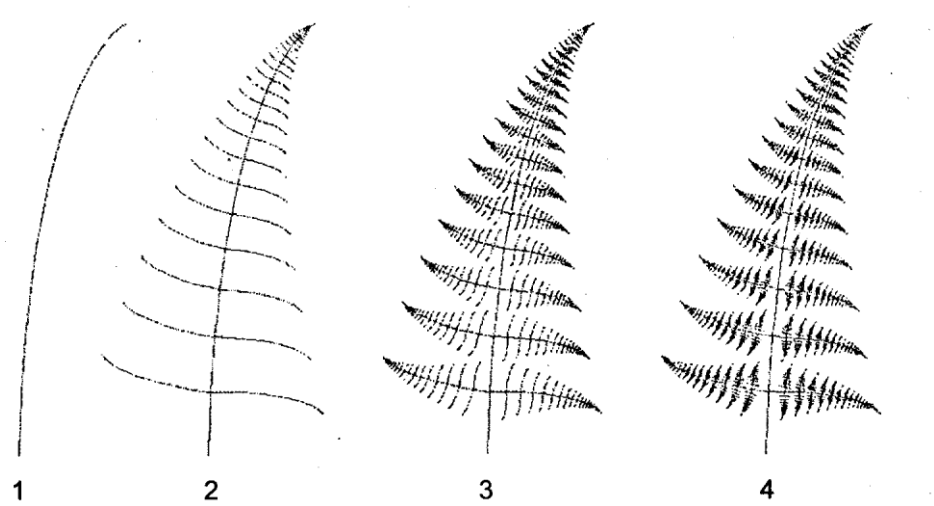


Рис. 5.4 Вид фрактала для разного количества циклов итераций

Метод IFS используется не только для создания изображений. Его использовали *Барнсли* и *Слоан* для эффективного сжатия графических изображений при записи в файл. Основная идея такая: поскольку фракталы могут представлять очень сложные изображения с помощью простых итераций, то описание этих итераций требует значительно меньшего объема информации, чем соответствующие растровые изображения. Для кодирования изображений необходимо решать обратную задачу — для изображения (или его фрагмента) подобрать соответствующие коэффициенты аффинного преобразования. Этот метод используется для записи цветных фотографий в файлы со сжатием в десятки и сотни раз без заметного ухудшения изображения. Формат таких графических файлов был назван FIF (Fractal Image Format) и запатентован фирмой Iterated Systems.

5.2 Обзор основных фрактальных программ

В 1997 году на рынке компьютерной графики произошло знаменательное событие. Среди известных производителей профессионального ПО для графики (Adobe, Macromedia, Autodeck, Corel, Microsoft) объявился новичок, способный захватить часть рынка графического ПО. Речь идет о компании MetaCreations Inc, которая была образована путем слияния нескольких коллективов разработчиков, специализировавшихся в областях двухмерной (2D) и трехмерной графики (3D). Остановимся на наиболее значимых из них.

Фирма MetaTools знаменита своим основателем — Каем Краузе, а также его детищами - наборами фильтров (plug-ins) для пакетов растровой (Kai's Power Tools) **и** векторной (Vector Effects)графики, модулями для программ обработки цифрового видео (Studio Effects) и генератором трехмерных ландшафтов КРГ Bryce. Компания Fractal Design впервые ввела в компьютерный обиход понятие Natural Media, представляющее возможность имитации “естественных” инструментов художника. Представьте, что вы рисуете на виртуальном холсте (то есть водите мышью по коврику **и** видите результат на экране монитора) инструментом программы Painter в режиме Oil (**Кисть с масляной краской)** лини. Зеленого цвета. Затем рядом тем же инструментом линию красного цвета. В результате вы видите, что в местах наложения цветов происходит смешивание электронных красок точно так же, но как они **бы смешивались** на бумаге или на холсте. Не удивительно, что программы Fractal Designer Painter, Expression и Kay Dream Studio дали толчок развитию компьютерной графики как искусства.

Итак, лидером на рынке фрактальной графики до недавнего времени (то есть до продажи своих программных продуктов другим фирмам) являлась компания Meta Creations. Спектр ее продуктов охватывает многие области компьютерной графики. Fractal Design Painter - программа для создания и обработки высокохудожественных растровых иллюстраций. Поддерживает многослойность изображений и возможность использования фильтров от программы Adobe Photoshop. Данная программа позволяет эмулировать большое число художественных инструментов: карандаши, кисти, пастели, разнообразные типы красок. На сегодняшний день Fractal Desing Painter -программа «номер один» для художников, использующих в своем творчестве компьютер. Для максимально удобства работы рекомендуется использовать графический планшет, поскольку в отличие от мыши он позволяет более точно передавать путь движения кисти.

Fractal Design Expression комбинирует в себе растровую и векторную технику компьютерной графики. То есть вы рисуете векторные объекты, как в CorelDRAW или Adobe Illustrator, редактируете их по опорным узлам и выполняете все прочие векторные операции. Но каждой линии, фигуре вы можете назначить любой растровый тип кисти. Кистей великое множество, ведь эта программа родилась во Fractal Design, фирме, знаменитой своей имитацией реальных инструментов художника. Здесь эмулируется практически все реальные растровое художественные инструменты и краски, а результатом работы является векторное изображение.

Глава 6. Цветовые модели компьютерной графики

Для изучения способов представления цвета в компьютерных системах сначала рассмотрим некоторые общие аспекты.

6.1 Элементы цвета

Представьте себе, что перед вами лежит лист белой бумаги с нарисованным на нем зеленым квадратом. Вы не задавали себе вопроса, «Почему этот цвет зеленый?» Ответ на него кроется в физических и биологических представлениях о природе

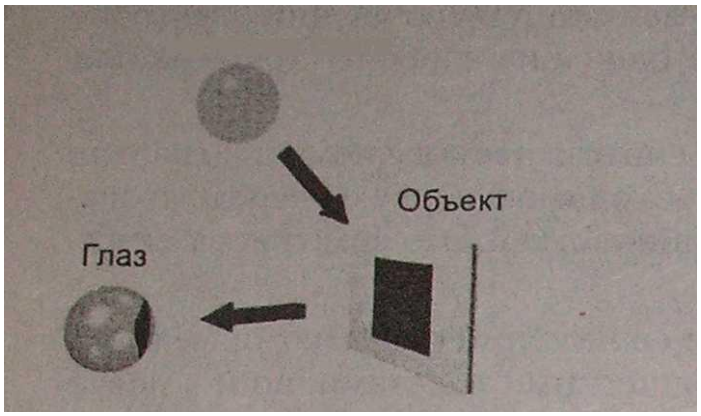


Рис. 6.1. Основные участники процесса восприятия цвета

Для того чтобы «увидеть» цвет, нужны три вещи (рис. 6.1):

- источник света;
- объект;
- ваш глаз (приемник излучения).

Теперь можно перейти к оценке роли физических и биологических аспектов процесса восприятия цвета.

Первый аспект — физика. Свет попадает на квадрат и отражается.

Второй аспект — биология. Отраженный свет попадает в глаз человека и воздействует на светочувствительные клетки глаза, которые содержат два типа рецепторов: палочки (cones) и колбочки (staves). Колбочки активны только в темноте или в сумерках. При нормальном освещении мы воспринимаем цвет исключительно с помощью палочек трех разновидностей, каждая из которых чувствительна к определенному диапазону видимого спектра. В данном случае отраженный от объекта свет воздействует на палочки, чувствительные к зеленому цвету. Они передают соответствующие импульсы в мозг, который после их обработки и последующей интерпретации выдает сообщение: квадратный, зеленый.

Но вопросы по-прежнему остаются.

Что в действительности стимулирует колбочки?

Почему в данном случае происходит возбуждение только одного типа палочек, который чувствителен к зеленому цвету?

Ответы на них будут даны ниже.

6.1.1 Свет и цвет

Как уже было отмечено в рассмотренном выше примере, наличие света является непременным условием визуального восприятия всего цветового богатства окружающего нас мира. В то же время из курса элементарной физики большинству из вас известно, что белый свет вне зависимости от его источника — солнце, лампочка или экран монитора — в действительности представляет собой смесь цветов. Если пропустить луч белого света через простую призму, он разложится на цветной спектр. Цвета этого спектра, называемого видимым спектром света, условно

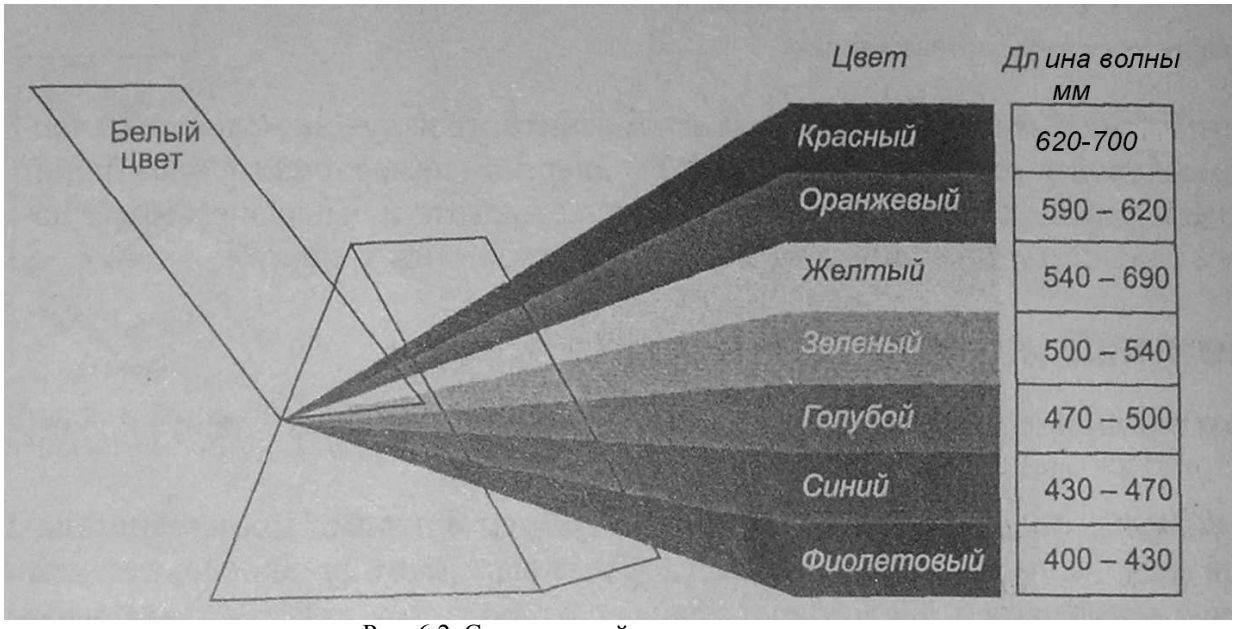


Рис. 6.2. Спектральный состав видимого цвета

классифицируют как красный, оранжевый, желтый, зеленый, голубой, синий и фиолетовый. Любой из них, в свою очередь, представляет собой электромагнитное излучение, перекрывающее достаточно широкий диапазон длин волн видимого спектра (рис. 6.2). Для нашего глаза каждый кусочек этого видимого спектра обладает своими уникальными характеристиками, которые и называются цветом. Поскольку в видимом спектре содержатся миллионы цветов, то различие между двумя соседними цветами практически неощутимо.

Спектральный состав цвета можно представить в виде графика распределения энергии излучения по разным длинам волн. Та длина волны, на которую приходится максимальная интенсивность излучения, называется доминирующей. Именно она в значительной степени определяет окраску цвета, хотя основные параметры воспринимаемого нашим глазом цвета определяются результатом воздействия на него всего спектрального состава цвета.

6.1.2 Физическая природа света и цвета

Напомним, что свет представляет собой электромагнитное излучение, связанное с флуктуацией электрического и магнитного полей. Иными словами, свет представляет собой энергию, а цвет есть продукт взаимодействия этой энергии с веществом. Однако для понимания природы цвета необходимо совершить небольшой экскурс в физику световых явлений и коснуться природы источников цвета.

Свет имеет двойственную природу, обладая свойствами волны и частицы. Корпускулы света, называемые фотонами, излучаются источником света в виде волн, распространяющихся с постоянной скоростью порядка 300000 км/с. Аналогично морским волнам световые волны имеют гребни и впадины. Поэтому в качестве характеристики световых волн используют длину волны, представляющую собой расстояние между двумя гребнями (единица измерения — метры или ангстремы, равные 10^8м), и амплитуду, определяемую как расстояние между гребнем и впадиной.

Разные длины волны воспринимаются нами как разные цвета: свет с большой длиной волны будет красным, а с маленькой — синим или фиолетовым. В случае если свет состоит из волн разной длины (например, белый цвет содержит все длины волн, то наш глаз смешивает разные длины воли в одну, получаем таким образом один результирующий цвет.

Рис. 6.3. Характеристики световой волны

Альтернативными характеристиками электромагнитного излучения являются частота (измеряемая в герцах или циклах/с) и энергия (измеряемая в электроно-вольтах). Чем короче длина волны, тем больше ее частота и выше энергия. И наоборот, чем больше длина волны, тем меньше частота и ниже энергия.

6.1.3 Излученный и отраженный свет

Все, что мы видим в окружающем нас пространстве, либо излучает свет, либо его отражает.

Излученный цвет — это свет, испускаемый активным источником. Примерами таких источников могут служить солнце, лампочка или экран монитора. В основе их действия обычно лежит нагревание металлических тел либо химические или термоядерные реакции. Цвет любого излучателя зависит от спектрального состава излучения. Если источник излучает световые волны во всем видимом диапазоне, то его цвет будет восприниматься нашим глазом как белый. Преобладание в его спектральном составе длин волн определенного диапазона (например, 400 - 450 нм) даст нам ощущение доминирующего в нем цвета (в данном случае сине-фиолетового). И наконец, присутствие в излучаемом свете световых компонент из разных областей видимого спектра (например, красной и зеленой) дает восприятие нами результирующего цвета (в данном случае желтого). Но при этом в любом случае попадающий в наш глаз излучаемый цвет сохраняет в себе все цвета, из которых он был создан.

Отраженный свет возникает при отражении некоторым предметом (вернее, его поверхностью) световых волн, падающих на него от источника света. Механизм отражения цвета зависит от цветового типа поверхности, которые можно условно разделить на две группы:

- ахроматические;
- хроматические.

Первую группу составляют ахроматические (иначе бесцветные) цвета: черный, белый и все серые (от самого темного до самого светлого). Их часто называют нейтральными. В предельном случае такие поверхности либо отражают все падающие на них лучи, ничего не поглощая (идеально белая поверхность), либо полностью лучи поглощают, ничего не отражая (идеальная черная поверхность). Все остальные варианты (серые поверхности) равномерно поглощают световые волны разной длины. Отраженный от них цвет не меняет своего спектрального состава, изменяется только его интенсивность.

Вторую группу образуют поверхности, окрашенные в хроматические цвета, которые по-разному отражают свет с разной длиной волны. Так, если вы осветите белым цветом листок зеленой бумаги, то бумага будет выглядеть зеленой, потому что ее поверхность поглощает все световые волны, кроме зеленой составляющей белого цвета. Что же произойдет, если осветить зеленую бумагу красным или синим цветом? Бумага будет восприниматься черной, потому что падающие на нее красный и синий цвета она не отражает. Если же осветить зеленый предмет зеленым светом, это позволит выделить его на фоне окружающих его предметов другого цвета.

Процесс отражения света сопровождается не только связанным с ним процессом поглощения в приповерхностном слое. При наличии полупрозрачных предметов часть падающего света проходит через них (рис. 6.4). На этом свойстве основано действие

фильтров фотоаппаратов, вырезающих из области видимого спектра нужный цветовой диапазон (иначе — отсекающих нежелательный цветовой спектр). Чтобы лучше понять этот эффект, прижмите к поверхности лампочки пластину цветного оргстекла. В результате наш глаз «увидит» цвет, не поглощенный пластиком.

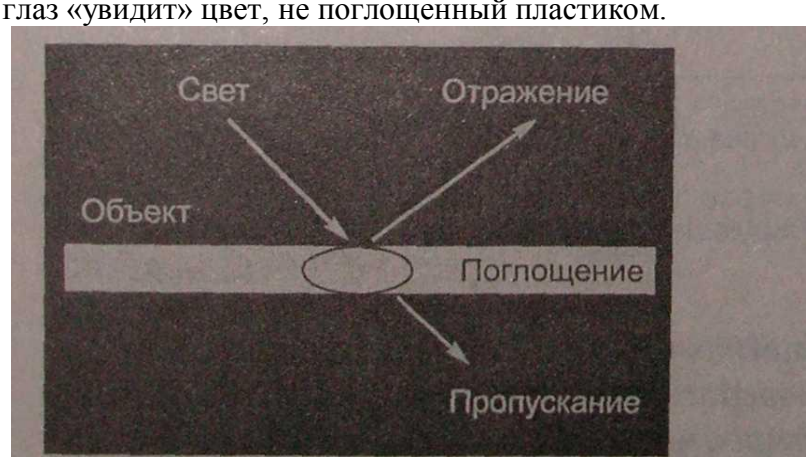


Рис. 6.4. Процессы отражения, поглощения и пропускания света объектом.

Каждый объект имеет спектральные характеристики отражения и пропускания. Эти характеристики определяют, как объект отражает и пропускает свет с определенными длинами волн.

• Спектральная кривая отражения определяется путем измерения отраженного света при освещении объекта стандартным источником.

•Спектральная кривая пропускания определяется путем измерения света, ярошедшего сквозь объект.

Некоторые измерительные устройства позволяют даже вводить поправки, компенсирующие изменение условий внешнего освещения.

Спектральные характеристики отражения и пропускания связаны с явлением мета» метрии, суть которого состоит в том, что объекты с разными спектральными характеристиками могут выглядеть одинаково при одном освещении и по-разному — при другом. Такое различие обусловлено как составом объектов, так и спектральным составом внешнего освещения. Для определения спектральных характеристик объектов используют специальные приборы, спектрофотометры, со стандартными источниками света.

Указанные различия в механизмах формирования излученного и отраженного цвета важны для понимания восприятия цвета глазом человека.

6.1.4 Яркостная и цветовая информация

Как уже отмечалось, излучаемый источником цвет, как правило, представляет собой смесь световых волн различной длины (рис. 6.5). Единственным исключением являются так называемые монохроматические источники света, примерами которых могут служить различные типы лазеров и широко распространенные натриевые лампы. Последние излучают свет только одной длины волны в оранжевой области спектра.

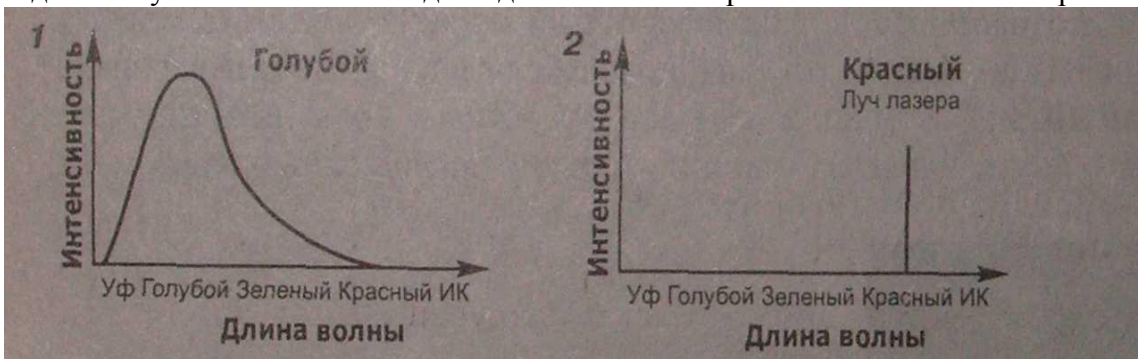


Рис. 6.5. Источники света: 1 - в виде смеси длин волн, воспринимаемой как голубой цвет в соответствии с цветом доминирующей длины волны; 2 - монохроматический красный цвет

Длина световых волн выражается в нанометрах (нм), представляющих собой миллиардные доли метра (10^{-9}). Наш глаз может воспринимать электромагнитные волны с длинами в диапазоне от 400 до 700 нм, что составляет ничтожно малую часть всего спектра электромагнитных волн (от 10^{4} до 10^{-14} м).

В действительности человеческий глаз может воспринимать цвет в более широком диапазоне длин волн — от 380 до 780 нм. Однако воздействие, оказываемое светом за пределами диапазона 400-700 нм, пренебрежимо мало.

Как уже отмечалось, энергия, переносимая электромагнитной волной, связана с длиной волны обратно пропорциональной зависимостью. Поэтому фиолетовая область видимого спектра, являясь коротковолновой, обладает более высокой энергией по сравнению с красной областью спектра.

С физической точки зрения свет можно охарактеризовать двумя параметрами: энергией (интенсивностью) и длиной волны. Однако в теории цвета, живописи, телевидении и компьютерной графике наибольшее распространение получили два производных от них параметра: яркость и цветность.

Яркость (или интенсивность) пропорциональна сумме энергий всех составляющих цветового спектра света.

Цветность, наоборот, связана с доминирующими длинами волн в этом спектре, Ахроматические цвета, то есть белые, серые и черные, характеризуются только яркостью. Это проявляется в том, что одни цвета темнее, а другие светлее. В отличие от них хроматические цвета для своего описания требуют задания и яркости, и цветности.

Распространенность указанных параметров обусловлена физиологическими особенностями нашего зрения, связанными с наличием в сетчатке глаза уже упоминавшихся ранее двух типов нервных клеток: палочек, реагирующих на яркостную составляющую света, и колбочек, воспринимающих цветовую информацию.

Яркость является количественной характеристикой цвета. С ее помощью мы можем сравнивать интенсивность излучения различных источников между собой. В отличие от нее цветность имеет качественный характер. Поэтому для того, чтобы сравнить два цвета по цветности, желательно было бы отделить их от яркости. Практически это невозможно, но теоретически вполне доступно с помощью имеющейся всех графических пакетах цветовой модели Lab. Присутствующие в ней абстрактные цветовые компоненты (собственно цветности) a и b обладают нулевой яркостью, а канал L содержит только яркостную информацию.

6.1.5 Цвет и окраска

Для правильной интерпретации восприятия цвета необходимо различать понятия цвета и окраски предмета.

Окраска — это способность предмета отражать излучение в том или ином диапазоне длин волн.

Цвет является более широким понятием, включающим окраску и условия освещения.

Чтобы представить имеющееся между ними различие, вспомните, как, например, выглядит снег при различных условиях освещения (зимний, мартовский или в сумерках) или сравните его изображения на картинах Пластова, Грабаря и Кустодиева. Несмотря на то что чистый снег всегда имеет белую окраску, его цвет в зависимости от освещения может не только быть белым, но иметь голубой, розовый и даже желтый оттенки. Эту разницу очень важно понимать при использовании цвета в прикладных целях, поскольку различия в освещении при настройке цветопередачи изображения разработчиком и последующем просмотре изображения потребителем дадут совершенно разные результаты.

Цвет - это один из факторов нашего восприятия светового излучения. Светом и цветом исследователи интересовались давно. Одним из первых выдающихся достижений

в этой области являются опыты *Исаака Ньютона* в 1666 г. по разложению белого света на составляющие. Ранее считалось, что белый цвет — простейшей. Ньютон опроверг это. Суть опытов Ньютона такова. Белый луч света (использовался солнечный свет) направлялся на стеклянную треугольную призму. Пройдя сквозь призму, луч преломлялся и, будучи направленный на экран, давал в результате цветную полосу — спектр. В спектре присутствовали цвета радуги, которые плавно переходили друг в друга. Эти цвета уже не раскладывались на составляющие. Ньютон разбил весь спектр на семь участков, соответствующих ярко выраженным различным цветам. Он считал эти семь цветов основными — красный, оранжевый, желтый, зеленый, голубой, синий и фиолетовый. Почему именно семь? Некоторые объясняют это убежденностью Ньютона в мистических свойствах семерки.

Вторая часть опытов Ньютона такова. Лучи, прошедшие сквозь призму, направлялись на вторую призму, с помощью которой удалось снова получить белый свет. Таким образом, *было* доказано, что белый цвет — это смесь множества разных цветов. Семь основных цветов Ньютон расположил по кругу (рис. 6.6).

Рис. 6.6. Цветовой круг Ньютона

Ньютон предположил, что определенный цвет получается путем смешивания основных цветов, взятых в определенной пропорции. Если в точках на границе цветового круга, которые соответствуют основным цветам, расположить грузы, пропорциональное количеству каждого цвета в смеси, то суммарный цвет будет соответствовать точке центра тяжести, Белый цвет соответствует центру цветового круга.

Последующие исследования цвета выполняли *Томас Юнг, Джемс Максвелл* и другие ученые. Исследования человеческого цветовосприятия являлись довольно важной задачей, но основные усилия были направлены на изучение объективных свойств света. В настоящее время физики полагают, что свет имеет двойственный характер. С одной стороны, свет представляется в виде потока частиц (еще Ньютон выдвинул так называемую корпускулярну теорию). С другой стороны, свету присущи волновые свойства. С помощью волновой теории, выдвинутой Христианом Гюйгенсом в 1678 году, были объяснены многие свойства света, в частности, законы отражения и преломления.

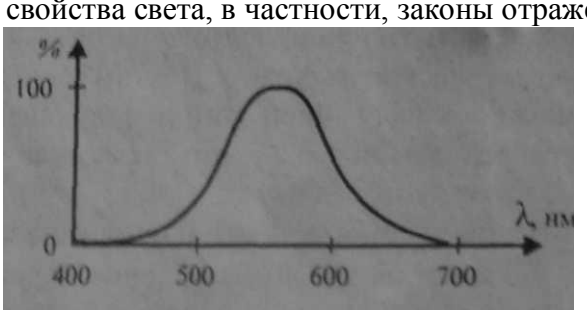


Рис. 6.7. Зависимость чувчтвительности человеческого зрения от длины волны светового излучения

С позиций волновых свойств цвет описывается следующим образом. Одна из волновых характеристик света - длина волны - расстояние, которое проходит волна в течение одного периода колебания. Монохромотическим называется излучение, спектр которого состоит из единственной линии, соответствующей единственной длине волны. Радуга, полученная Ньютоном, состоит из бесчисленного множества монохроматических излучений (равно как и радуга, наблюдаемая нами после дождя). Довольно качественным источником монохроматического излучения является лазер - именно поэтому его луч легко сфокусировать. Цвет монохроматического излучения определяется длиной волны.

Диапазон дайн волн для видимого света простирается от 380-400 нм (фиолетовый) до 700-780 нм (красный). В указанном диапазоне чувствительность человеческого зрения непостоянная. Наибольшая чувствительность наблюдается для длин волн, соответствующих зеленому цвету.

Как показал Ньютон, белый цвет можно представить смесью всех цветов радуги. Другими словами, спектр белого является бесконечным, сплошным - в нем присутствуют излучения всех длин волн видимого диапазона.

6.2 Характеристики источника света

6.2.1 Стандартные источники

Для имитации различного освещения измерительные устройства используют стандартизованные источники излучения - D50, D65, D93, A, B, C, а также F2 или F8 (флюоресцентные лампы). Эти источники имеют определенные стандартизованные спектральные характеристики, установленные в 1931 г. международной комиссией по освещению (CIE):

• источник А — норма среднего искусственного света эквивалентна цветовой температуре 2854°К, что соответствует излучению лампы накаливания;

• источник В — норма прямого солнечного света с цветовой температурой, близкой к 4800°К;

• источник С - норма рассеянного дневного света с температурой около 6500°К;

• источник D65 имеет температуру, почти равную 6500°К (применяется во всем мире, кроме Германии, где стандартным считается D50 с цветовой температурой 5000°К).

Источники В и С в действительности получают из источника А путем изменения спектральной характеристики последнего с помощью соответствующего фильтра.

6.2.2 Особенности восприятия цвета человеком

Световые волны, излучаемые или отражаемые объектом, собираются хрусталиком и через стекловидное тело проецируются на сетчатку (рис. 6.8). Там они возбуждают определенные нервные клетки, физиологическое назначение которых состоит в распознавании световых волн. В результате возбуждения в нервных клетках возникает электрический сигнал, который по зрительному нерву поступает в зрительный центр мозга, где с помощью пока еще до конца не понятных механизмов и возникает зрительное восприятие цвета.

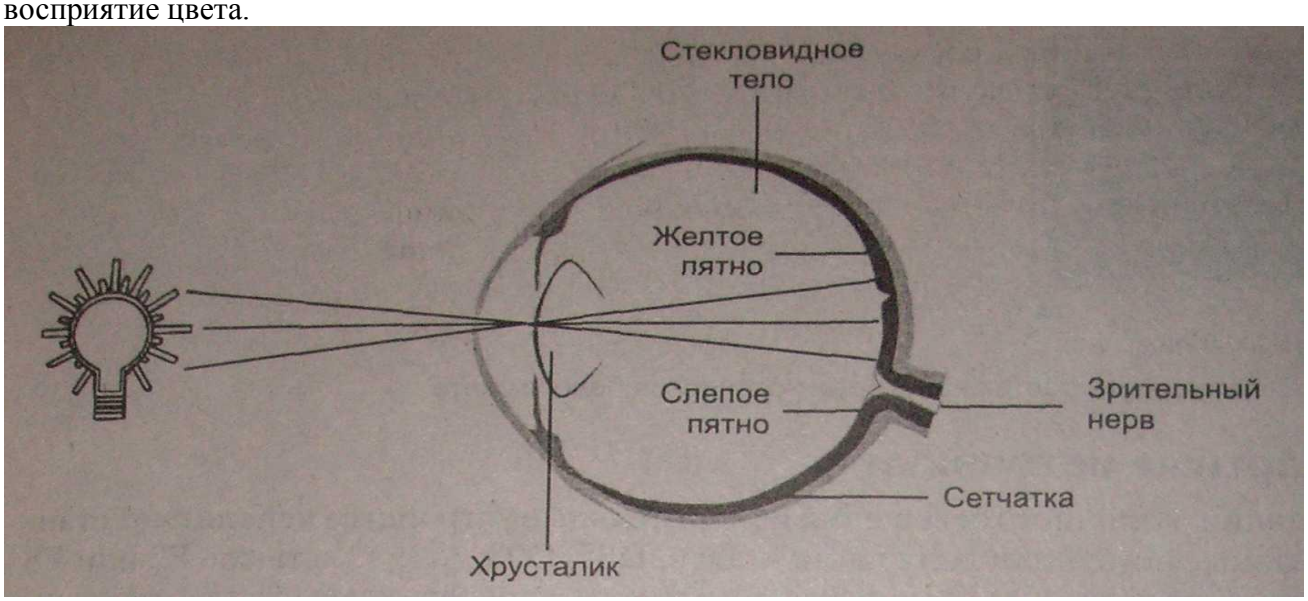


Рис. 6.8. Схема функционирования человеческого глаза

На самой сетчатке можно выделить две области, которые называют желтым пятном и слепым пятном. На слепом пятне нервные пути сетчатки переходят в зрительный нерв.

Поскольку в атом месте нервных клеток нет, то свет, попадающий на слепое пятно, не обнаруживается. На желтом пятне имеет место обратная картина. Оно расположено по центру зрительной оси и содержит много зрительных клеток, чувствительных к цвету (колбочек; см. ниже). При хорошем освещении глаз обычно фокусирует падающий свет на желтом пятне. Наоборот, ночью сильной фокусировки приходится избегать, поскольку из-за низкой чувствительности колбочек зрительное восприятие значительно ослабляется.

Колбочки и палочки

За цветовое и яркостное восприятие человеческого глаза отвечают два различных типа нервных клеток (рецепторов), называемых соответственно колбочками и палочками.

Процесс функционирования палочек и колбочек не имеет принципиальных отличий. В обоих случаях происходит поглощение световых волн и по достижении определенного порога вырабатывается нервный импульс. При этом оба вида нервных клеток реагируют на интенсивность падающего света. В чем же тогда проявляется их различие?

Палочки «отвечают» за черно-белое зрение, поскольку способны регистрировать только суммарную энергию света. Этот тип рецепторов равномерно распределен по сетчатке глаза и обладает очень высокой чувствительностью, примерно в 1000 раз превышающей чувствительность колбочек. Именно благодаря им обеспечивается возможность распознавания предметов в условиях плохой освещенности («ночью все кошки серы»).

Колбочки предназначены для распознавания цветовой информации. В отличие от палочек имеются три сорта колбочек, каждая из которых реагирует на определенный диапазон длин волн. Из экспериментальных данных видно, что первый тип воспринимает световые волны с длинами волн в диапазоне 400-500 нм («синяя» составляющая спектра), второй — от 500 до 600 нм («зеленая» составляющая спектра) и третий — от 600 до 700 («красная» составляющая спектра). В зависимости от того, световые волны какой длины и интенсивности присутствуют в спектре, те или иные группы колбочек возбуждаются сильнее или слабее. Полученная с помощью зрительных рецепторов информация поступает в виде сигналов в мозг, который определяет, в каких соотношения возбуждены три вида колбочек, создавая на базе этого цветовое восприятие.

Таким образом, исходя из особенностей строения человеческого глаза можно сделать вывод, что цвет трехмерен по своей природе.

Рис. 6.9. Спектральная чувствительность различных типов колбочек

Принцип действия большинства технических устройств, предназначенных для обработки содержащейся в свете цветовой информации, также базируется на раздельном распознавании красной, зеленой и синей составляющих света.

Настало время разобраться с тем, как свойства палочек и колбочек влияют на чувствительность зрения к яркости света.

Спектральная чувствительность глаза к яркости

Как можно увидеть из рис. 6.9, области чувствительности различных типов колбочек значительно перекрываются. Поэтому, как правило, в процессе восприятия глазом падающего на него света возбуждаются все три сорта колбочек. А поскольку чувствительности разных типов колбочек отличаются очень сильно, то глаз человека имеет неодинаковую чувствительность к разным длинам волн. Особенно хорошо воспринимается зеленый цвет, красный — несколько слабее, а чувствительность к синему цвету чрезвычайно низка. В результате отдельные цветовые составляющие цветного изображения вносят разный вклад в ощущение яркости.

На практике в качестве яркостной характеристики чувствительности глаза обычно используют кривую спектральной чувствительности (рис. 6.10). Для дневного освещения ее можно получить путем суммирования приведенных на рис.6.10 спектральных составляющих трех типов колбочек с последующим нормированием полученной кривой (путем деления всех ее составляющих на максимальное значение яркости). По существу этот график представляет собой не что иное, как КПД человеческого глаза. По графику можно легко оценить, какая часть попавшего в глаз света вносит наибольший вклад в формирование ощущения цвета. Так, для получения с помощью синего цвета такого же ощущения яркости, как от зеленого цвета, его реальная энергия должна быть в несколько раз выше.

Рис. 6.10. Кривые спектральной чувствительности глаза при различных условиях внешнего освещения: 1- в сумерках, 2 - при дневном освещении

При получении кривой спектральной чувствительности в качестве нормирующего коэффициента используется принятая за единицу спектральная эффективность желто-зеленого излучения с длиной волны 555 нм. Для отображения интенсивности составляющей спектра используется обозначение V_λ.

При оценке яркостной чувствительности цвета следует учитывать, что свой вклад в ощущение яркости вносят как колбочки, так и палочки (рис. 6.11). А поскольку максимальная чувствительность палочек по сравнению с колбочками лежит в более коротковолновой области спектра (соответственно 500 нм против 555 нм), то именно этим фактором объясняется зависимость спектральной чувствительности от внешней освещенности.

Вклад палочек и колбочек в результирующее значение яркости определяется условиями освещенности. Так, в темноте работают только палочки. В сумерках в формировании яркостного восприятия участвуют и палочки, и колбочки, а при повышении уровня освещенности начинают доминировать палочки.

Рис. 6.11. Спектральная чувствительность палочек и колбочек

Это легко проверить на практике. Если вам приходилось встречать рассвет за рыбалкой на реке или озере, то вы можете вспомнить, что поначалу серый окружающий фон с восходом солнца понемногу расцвечивается в цветовые тона.

В результате проведения многочисленных психологических тестов наряду с теоретическими исследованиями было установлено, что для большинства людей ощущение яркости при восприятии цветных изображений определяется на 71,5% зеленой составляющей, 21 % — красной и 7,2% — синей. Таким образом, если известны зеленая, синяя и красная составляющие источника цвета, то воспринимаемая нашим глазом результирующая яркость такого источника может быть определена по формуле:

$$Яркость = 0{,}715160 \times зеленый + 0{,}212671 \times красный + 0{,}072169 \times синий.$$

Существуют и другие выражения для определения яркости.

Для стандарта в области NTSC телевидения:

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

Спектральная чувствительность наблюдателя

Спектральная чувствительность определяет диапазон принимаемых наблюдателем или приемником цветов. На рис. 6.12 представлена спектральная чувствительность глаза. Левее синей области частот — ультрафиолетовые волны, правее красной — инфракрасные волны. Наилучшую чувствительность глаз имеет в районе 555 нм (зеленый цвет). На спектральную чувствительность влияют качество источника света, а также различия в цветах фона и в углах обзора.

Рис. 6.12. Спектральная чувчтвительность глаза

Роль фона легко понять, так как его цвет воспринимается теми же чувствительными органами, что и цвет объекта.

От углов обзора (углов, под которыми наблюдается данный объект) зависит интенсивность отражаемого света (для отражающих поверхностей), а также появление теней и различие в интенсивности излучения (для излучающих поверхностей). Прямое излучение является более сильным, чем излучение под углами.

Путем измерения спектральной чувствительности наблюдателя (или принимающего устройства) мы регистрируем его способность воспринимать свет с различными длинами волн.

6.3 Цветовой и динамический диапазоны

Для эффективной организации передачи информации между различными устройствами, входящими в состав издательских систем, важно понимать разницу между цветовым и динамическим диапазонами.

Цветовой диапазон — диапазон цветов, которые могут восприниматься или воспроизводиться наблюдателем или приемным устройством.

Динамический диапазон характеризует различие между наиболее светлым и наиболее темным элементами в изображении или в поле зрения.

Человеческое зрение имеет широчайший цветовой и динамический диапазон. Глаз человека способен различать градации миллионных долей яркости.

Компьютерные устройства имеют сравнительно узкие цветовой и динамический диапазоны. Кроме того, имеются различия в характеристиках разных устройств. Например, цветовые и динамические диапазоны сканеров и мониторов шире, чем соответствующие диапазоны принтеров.

В совокупности цветовой и динамический диапазоны определяют область воспринимаемых нами цветов и области цветов (цветовое пространство), в которых работают устройства ввода, вывода и обработки изображений. Для представления этих областей используются два способа:

1. В виде различных цветовых моделей.

2. С помощью набора цветов (палитр), доступных в системах соответствия цветов. Для каждой из таких систем - DIC, DuPont®, FOCOLTONE®, PANTO-NE®, TOYO и TRUMATCH®, — определены специальные цвета, которые можно выбирать по каталогам образцов. За исключением плашечных цветов палитры PANTONE®, эти системы подстановки цветов связываются с цветовыми моделями. Системы DIC и TOYO базируются на совместном использовании основных цветов и специальных красителей.

Современные графические пакеты оперируют большим количеством специфических терминов, включающих определение цветовые, цвет. Перечислим их:

- цветовые модели;
- цветовые палитры, которые в свою очередь подразделяются на плашечные цветовые палитры и основные цветовые палитры;
- системы соответствия цветов;
- системы управления цветами.

Их обилие и внешняя схожесть могут смутить не только новичка в области обработки компьютерных изображений. Далее будет дано последовательное разъяснения смысла и назначения этих терминов.

Для характеристики цвета используются следующие атрибуты.

1. Цветовой тон. Его можно определить преобладающей длиной волны в спектре излучения. Цветовой тон позволяет отличать один цвет от другого — например, зеленый от красного, желтого и других.

2. Яркость. Определяется энергией, интенсивностью светового излучения. Выражает количество воспринимаемого света.

3. Насыщенность или чистота тона. Выражается долей присутствия белого цвета. В идеально чистом цвете примесь белого отсутствует. Если, например, к чистому красному цвету добавить в определенной пропорции белый цвет (у художников это называется "разбелом"), то получится светлый бледно-красный цвет.

Укачанные три атрибута позволяют описать все цвета и оттенки. То, что атрибутов именно три, является одним из проявлений трехмерных свойств цвета. Далее мы увидим, что имеются и другие трехмерные системы описания цвета.

Мы попытались объяснить цвет с помощью длин волн и спектра. Как оказывается, это неполное представление о цвете, а вообще говоря, оно неправильное. Во-первых, глаз человека - это не спектрометр. Зрительная система человека, скорее всего, регистрирует на длину волны и спектр, а формирует ощущения другим способом. Во-вторых, без учета особенностей человеческого восприятия невозможно объяснить смешение цветов. Например, белый цвет действительно можно представить равномерным спектром смеси бесконечного множества монохроматических цветов. Однако тот же белый цвет можно создать смесью всего двух специально подобранных монохроматических цветов (такие цвета называются взаимно дополнительными), Во всяком случае, человек воспринимает эту смесь как белый цвет. А можно получить белый цвет, смешав три нам более монохроматических излучений. Излучения, различные по спектру, но дающие один и тот же цвет, называются метамерными.

Необходимо также уточнить, что понимается под цветовым тоном. Рассмотрим два примера спектра (рис. 6.13).

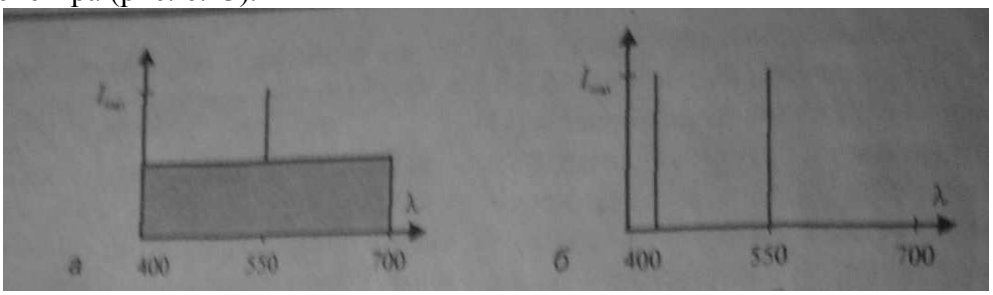


Рис. 6.13. Спектры: а - в сплошном спектре имеется явное преобладание одной составляющей: б - в дискретном спектре две составляющие с одинаковой интенсивностью

Анализ спектра, изображенного на рис. 6.13 (а), позволяет утверждать, что излучение имеет светло-зеленый цвет, поскольку четко выделяется одна спектральная линия на фоне равномерного спектра белого. А какой цвет (цветовой тон) соответствует спектру варианта (б)? Здесь нельзя в спектре преобладающую составляющую, поскольку присутствуют красная и зеленая линии одинаковой интенсивности. По законам смешения цветов, это может дать оттенок желтого цвета — однако в спектре нет соответствующей линии монохроматического желтого. Поэтому, под цветовым тоном следует понимать цвет монохроматического излучения, соответствующего суммарному цвету смеси. Впрочем, как именно соответствующего" — это также требует уточнения.

Наука, которая изучает цвет и его измерения, называется колориметрией. Она описывает общие закономерности цветового восприятия света человеком. Одними из основных законов колориметрии являются законы смешивания цветов. Эти законы в наиболее полном виде были сформулированы в 1853 году немецким математиком *Германом Грассманом.*

1. Цвет - трехмерен, для его описания необходимы три компонента. Любые четыре цвета находятся в линейной зависимости, хотя существует неограниченное число линейно-независимых совокупностей из трех цветов.

Другими словами, для любого заданного цвета (*Ц*) можно записать такое цветовое уравнение, которое выражает линейную зависимость цветов:

$$Ц = к_1 Ц_1 + к_2 Ц_2 + к_3 Ц_3,$$

где $Ц_1$, $Ц_2$,$Ц_3$ - некоторые базисные, линейно-независимые цвета, коэффициенты $к_1$, $к_2$ и $к_3$ указывают количество соответствующего смешиваемого цвета, Линейная независимость цветоа $Ц_1$, $Ц_2$, $Ц_3$ означает, что ни один из них не может быть выражен взвешен ной суммой (линейной комбинацией) двух других. Первый закон можно трактовать и в более широком смысле, а именно, в смысле трехмерности цвета. Необязательно для описания цвета использовать смесь других цветом можно применять и другие компоненты, но их обязательно должно быть три.

2. Если в смеси трех цветовых компонентов один меняется непрерывно, в то время как два других остаются постоянными, цвет смеси также изменяете непрерывно.

3. Цвет смеси зависит только от цветов смешиваемых компонентов и не зависит от их спектральных составов. Смысл третьего закона становится более понятным, если учесть, что один и тот же цвет (в том числе и цвет смешиваемых компонентов) может быть получен разными способами. Например, смешиваемый компонент может быть получен, в свою очередь, смешиванием других компонентов.

На теоретической базе этих законов существуют все современные цветовые модели.

6.4 Типы цветовых моделей

Большинство графических пакетов позволяют оперировать широким кругом цветовых моделей, часть из которых создана для специальных целей, а другая - для особых типов красок. Перечислим их:

- CMY;
- CMYK;
- RGB;
- HSB;
- HLS;
- Lab;
- YIQ;
- YCC.

По принципу действия перечисленные цветовые модели можно условно разбить на три класса:

- аддитивные (RGB), основанные на сложении цветов;
- субтрактивные (CMY, CMYK), основу которых составляет операция вычитания цветов (субтрактивный синтез);
- перцепционные (HSB, HLS, Lab, YCC), базирующиеся на восприятии.

Перед тем как перейти к непосредственному рассмотрению конкретных цветовых моделей, уделим немного внимания общим физическим закономерностям, свойственным природе цвета.

Эта модель используется для описания цветов, которые могут быть получены с помощью устройств, основанных на принципе излучения. В качестве основных цветов берется красный (Red), зеленый (Green) и синий (Blue). Другие цвета и оттенки могут быть получены смешиванием определенного количества любого из основных цветов.

6.4.1 Аддитивные цветовые модели

Аддитивный цвет получается на основе законов Грассмана путем соединения лучей света разных цветов. В основе этого явления лежит тот факт, что большинство цветов видимого спектра могут быть получены путем смешивания в различных пропорциях трех основных цветовых компонент. Этими компонентами, которые в теории цвета иногда называются первичными цветами, являются красный (Red), зеленый (Green) и синий (Blue) цвета. При попарном смешивании первичных цветов образуются вторичные цвета: голубой (Cyan), пурпурный (Magenta) и желтый (Yellow). Следует отметить, что первичные и вторичные цвета относятся к базовым цветам.

Базовыми цветами называют цвета, с помощью которых можно получить практически весь спектр видимых цветов.

Для получения новых цветов с помощью аддитивного синтеза можно использовать и различные комбинации из двух основных цветов, варьирование состава которых приводит к изменению результирующего цвета. На рис. 6.14 приведена схема получения новых цветов на базе двух первичных путем использования источников зеленого и красного цветов, интенсивностью каждого из которых можно управлять с помощью фильтра. Можно увидеть, что равные пропорции первичных цветов дают желтый цвет (1, 2); снижение в смеси интенсивности зеленого цвета при той же интенсивности красного позволяет синтезировать оранжевый цвет (3, 4); подобные колометрические схемы позволяют создать желтый и оранжевый цвета в виде геометрического места цветовых точек — локуса (2,4). Однако таким способом нельзя получить некоторые цвета, например голубой, для создания которого требуется наличие третьего первичного цвета — синего.

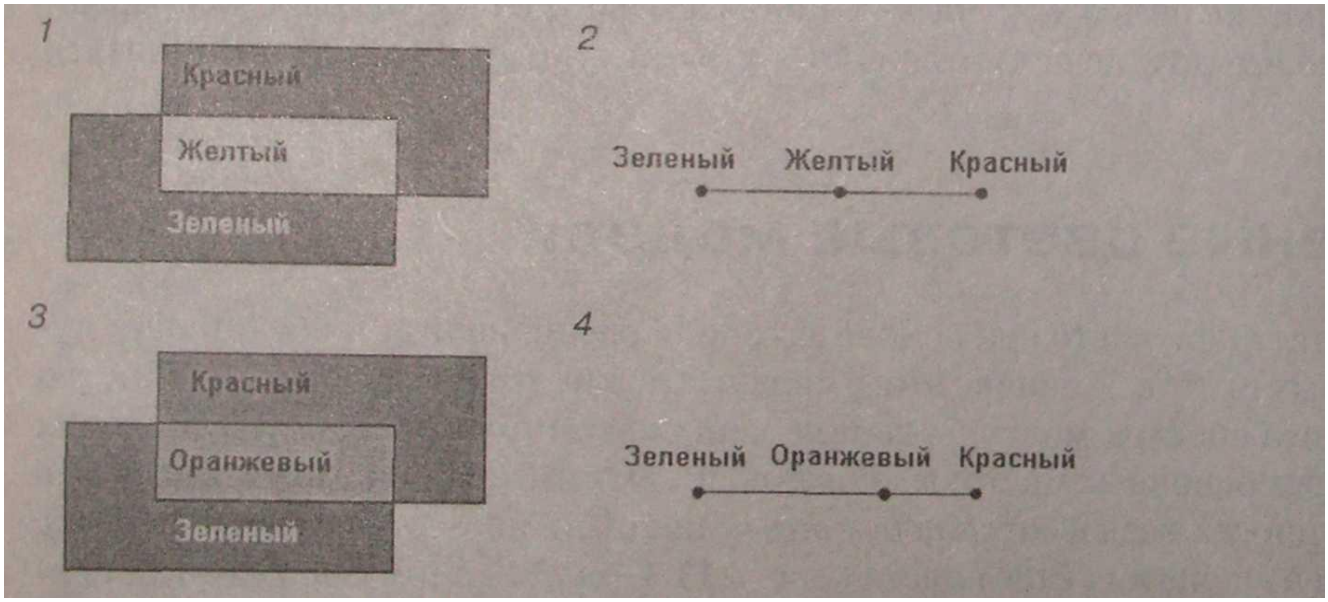


Рис. 6.14. Аддитивный синтез новых цветов на базе разного процентного соотношения двух первичных цветов - красного и зеленого.

Аддитивные цвета нашли широкое применение в системах освещения, видеосистемах, устройствах записи на фотопленку, мониторах, сканерах и цифровых камерах.

Используемые для построения RGB-модели первичные, или аддитивные, цвета имеют еще одно название. Иногда, чтобы подчеркнуть тот факт, что при добавлении света интенсивность цвета увеличивается, эту модель называют добавляющей. Такое обилие терминов, используемых для описания RGB-модели, связано с тем, что она возникла задолго до появления компьютера и каждая область ее применения внесла свой вклад в терминологию.

.

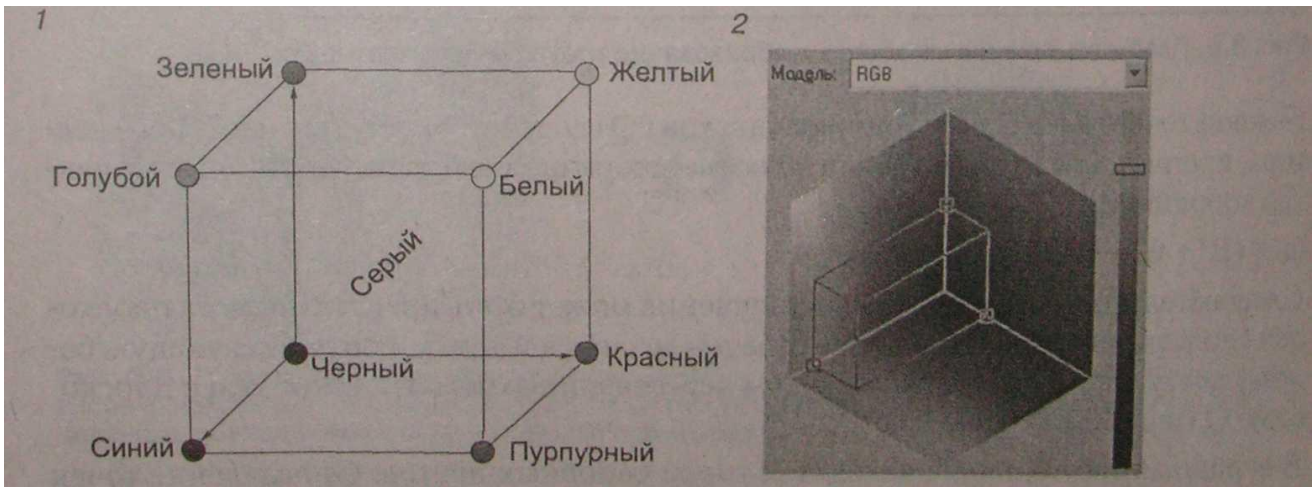


Рис. 6.15. Представление RGB-модели в виде куба: 1 - схема модели; 2 - практическая реализация модели в пакете Corel PHOTO-PAINT

Рис. 6.16. Модель Т. Юта

RGB - модель

Вкратце история модели RGB такова. *Томас Ют* (1773-1829) взял три фонаря и приспособил к ним красный, зеленый и синий светофильтры. Так были получены источники света соответствующих цветов. Направив на белый экран свет этих трех источников, ученый получил такое изображение (рис. 6.16). На экране свет от источников давал цветные круги. В местах пересечения кругов наблюдалось смешивание цветов. Желтый цвет давало смешивание красного и зеленого, голубой — смесь зеленого и синего, пурпурный — синего и красного, а белый цвет образовался смешением всех трех основных цветов. Некоторое время спустя, *Джемс Максвелл* (1831-1879) изготовил первый колориметр, с помощью которого человек мог сравнивать монохроматический цвет и цвет смешивания в заданной пропорции компонентов RGB. Регулируя яркость любого из смешиваемых компонентов, можно добиться уравнивания цветов смеси и монохроматического излучения. Это описывается следующим образом:

$$Ц = rR + gG + bB,$$

где *r, g* и *b* — количество соответствующих основных цветов. Эта модель используется для описания цветов, которые могут быть получены с помощью устройств, которые основаны на принципе излучения. В качестве основных цветов выбран красный (Red), зеленый (Green) и синий (Blue). Другие цвета й оттенки могут быть получены смешиванием определенного количества основных цветов.

Соотношение коэффициентов r, *g и b* Максвелл наглядно показал с помощью треугольника, со временем названного его именем. Треугольник Максвелла является равносторонним, в его вершинах располагаются основные цвета — *R, G* и *B* (рис. 6.17). Из заданной точки проводятся линии, перпендикулярные сторонам треугольника. Длина каждой линии и показывает соответствующую величину коэффициента *г, g* или *b.* Одинаковые значения *r=g=b* имеют место в центре треугольника и соответствуют белому цвету. Следует также указать, что некоторые цвета отображаются точками вне

треугольника RGB, — это означает отрицательное значение соответствующего цветового коэффициента. Сумма коэффициентов равняется высоте треугольника, а при высоте, равной единице, составляет $r + g + b = 1$.

В качестве основных цветов, Максвелл использовал излучения с такими длинами волн: 630, 528 и 457 им.

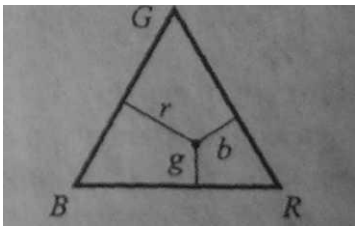


Рис. 6.17. Треугольник Максвелла

К настоящему времени система RGB — это официальный стандарт. Решением Международной Комиссии по Освещению — *МКО (CIE — Commision International de VEclairage)* в 1931 году были стандартизированы основные цвета, которые было рекомендовано использовать в качестве *R, G* и *B*. Это монохроматические цвета светового излучения с длинами волн соответственно:

R — 700 нм, *G* — 546.1 нм, B — 435.8 нм.

Красный цвет получается с помощью лампы накаливания с фильтром. Для получения чистых зеленых и синих цветов используется ртутная лампа. Также стандартизировано значение светового потока для каждого основного цвета.

Еще одним важным параметром для системы RGB является цвет, полученный после смешивания трех компонентов в равных количествах. Это белый цвет. Оказывается, для того, чтобы смешиванием компонентов R,G, и B получить белый цвет, яркости соответствующих источников не должно быть равным, и должны находиться в пропорции

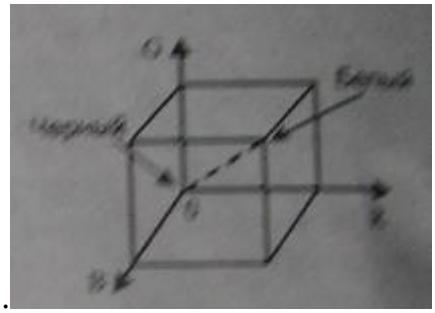


Рис. 6.18. Трехмерные координаты RGB

$L_R : L_G : L_B = 1 : 4{,}5907 : 0{,}0601$

Если расчеты цвета делаются для источников излучения с одинаковой яркостью, то указанное соотношение яркостей можно учесть соответствующими масштабными коэффициентами.

Теперь рассмотрим другие аспекты. Цвет, создаваемый смешиванием трех основных компонентов, можно представить вектором в трехмерной системе координат R, G и B, изображенной на рис. 6.18. Черному цвету соответствует центр координат - точка (0,0,0). Белый цвет выражен максимальным значением компонентов. Пусть это максимальное значение вдоль каждой оси равняется единице. Тогда белый цвет - это вектор (1,1,1). Точки, которые лежат на диагонали куба от черного к белому, имеют одинаковые значения координат: $R_1 = G_1 = B_1$. Это градации серого - их можно считать белым цветом разной яркости. Следовательно, если все компоненты вектора (*r ,g ,b*) умножить на одинаковый коэффциент (k=0..1...1), то цвет (*kr, kg, kb*) сохраняется, изменяется только яркость. Поэтому для анализа цвета важно соотношение компонентов. Если в цветовом уравнении

$$Ц = rR + gG + bB$$

разделить коэффициенты *r, g* и *b* на их сумму:

$$r' = \frac{r}{r+g+b}, r' = \frac{r}{r+g+b}, r' = \frac{r}{r+g+b},$$

то можно записать такое цветовое уравнение

$$Ц = r' R + g' G + b' B.$$

Это уравнение представляет векторы цвета (r', g', b'), которые лежат в единичной плоскости $r' + g' + b' = 1$. Иными словами, мы перешли от куба к треугольнику Максвелла.

В ходе колориметрических экспериментов были определены коэффициенты (r', g', b'), *это призма из белого гипса, грани которой освещаются источником света.* На левую грань направлен источник чистого монохромотического излучения, а правая грань освещается смесью трех источников RGB. Наблюдатель видит одновременно две грани, что позволяет фиксировать равенство цветов.

Результаты экспериментов можно изобразить графически (рис.6.19).

Как видим, коэффициенты r', g', b' могут быть и положительными, и отрицательными, суммой компонентов R,G,B. Но как отнять то, чего нет? Для уравнивания цвета пришлось прибавить к монохроматическому излучению один из компонентов R,G или B. Например, если монохромическое излучение для некоторого значения λ разбавлялось красным, то это можно выразить так:

$$Ц(\lambda) + r'(\lambda)R = g'(\lambda)G + b'(\lambda)B$$

Как оказалось, ни один цвет монохромотического излучения не может быть

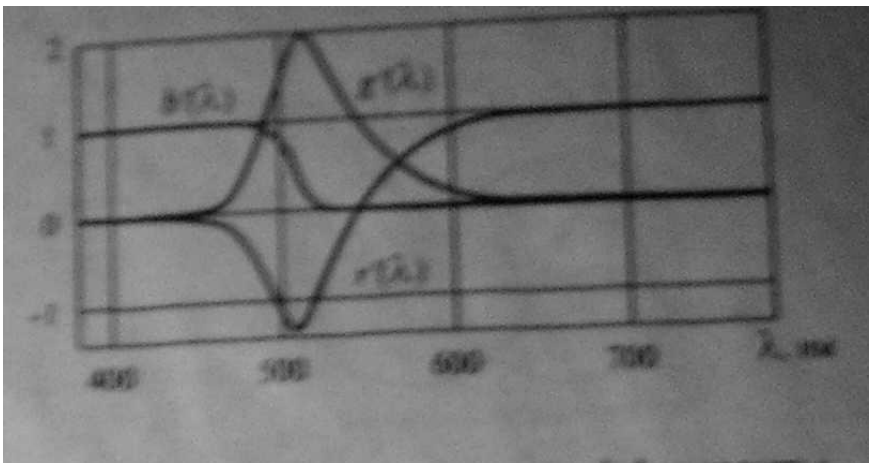


Рис. 6.19. Трехцветные коэффициенты смешивания RGB

представлен только положительными значениями коэффициентов смешивания. Это наглядно можно изобразить с помощью цветового графика, построенного на основе треугольника Максвелла (рис.6.20). Верхняя часть кривой линии соответствует чистым монохромотическим цветам, а нижняя линия - от 380 нм до 780 нм - представляет так называемые пурпурные цвета (смесь синего и красного), которые не являются монохромотическими. точки, которые лежат внутри контура кривой, соответствуют реальным цветам, а вне этого контура - нереальным цветам. Точки внутри треугольника соответствуют положительным значениям коэффициентов r', g', b' и представляют цвета, которые можно получить смешиванием компонентов RGB.

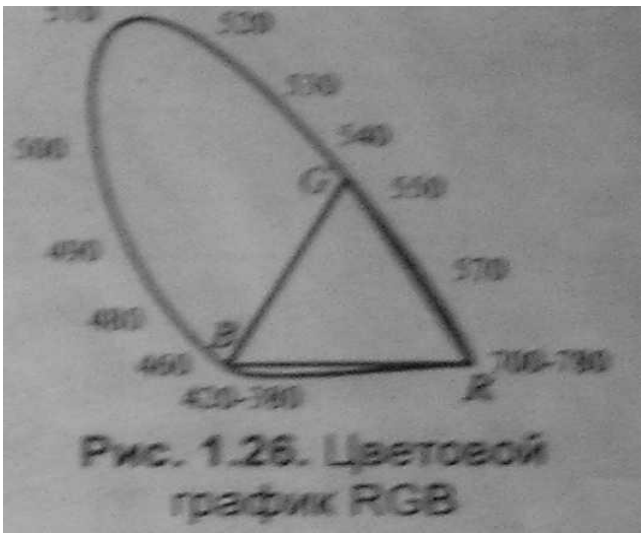


Рис. 1.26. Цветовой график RGB

Рис. 6.20. Цветовой график RGB

Таким образом, система RGB имеет неполный цветовой охват - некоторые насыщенные цвета не могут быть представлены смесью указанных трех компонентов. В первую очередь, это цвета от зеленого к синему, включая все оттенки голубого - они соответствуют левой части кривой цветового графика. Ещё раз подчеркнем, что речь здесь идет о насыщенных цветах, так как ненасыщенные голубые цвета получить можно смешиванием компонентов RGB. Несмотря на неполный охват, система RGB широко используется в данное время - в первую очередь, в цветных телевизорах и дисплеях компьютеров. Отсутствие некоторых оттенков цвета не слишком заметно.

Ещё одним фактором, способствующим популярности системы RGB, является ее наглядность - основные цвета находятся в трех четко различимых участках видимого спектра. Кроме того, одна гипотеза, объясняющих цветовое зрение человека - трехкомпонентная теория - утверждает, что в зрительной системе человека есть три типа светочувствительных элементов.

Один тип элементов реагирует на зеленый, другой тип - на красный, а третий тип - на синий цвет.

Такая гипотеза высказывалась ещё Ломоносовым, её обоснованием занимались многие ученый, начиная с Т.Юнга. Впрочем, трехкомпонентная теория не является единственной теорией цветового зрения человека.

Почему RGB-модель нравится компьютеру?

В графических пакетах цветовая модель RGB используется для создания цветов изображения на экране монитора, основными элементами которого являются три электронных прожектора и экран с нанесенными на него тремя разными люминофорами (рис. 3.6, /). Точно так же, как и зрительные пигменты трех типов колбочек, эти люминофоры имеют разные спектральные характеристики. Но в отличие от глаза они не поглощают, а излучают свет. Один люминофор под действием попадающего на него электронного луча излучает красный цвет, другой — зеленый и третий — синий.

Мельчайший элемент изображения, воспроизводимый компьютером, называется пикселом (pixel от pixture element). При работе с низким разрешением отдельные пикселы не видны. Однако если вы будете рассматривать белый экран включенного монитора через лупу, то увидите, что он состоит из множества отдельных точек красного, зеленого и синего цветов, объединенных в RGB-элементы в виде триад основных точек. Цвет каждого из воспроизводимых кинескопом пикселов (RGB-элементов изображения) получается в результате смешивания красного, синего и зеленого цветов входящих в него трех люминофорных точек. При просмотре изображения на экране с некоторого расстояния эти цветовые составляющие RGB-элементов.

Для назначения цвета и яркости точек, формирующих изображение монитора, нужно задать значения интенсивностей для каждой из составляющих RGB-элемента (пиксела). В этом процессе значения интенсивностей используются для управления мощностью трех электронных прожекторов, возбуждающих свечение соответствующего типа люминофора. В то же время число градаций интенсивности определяет цветовое разрешение, или, иначе, глубину цвета, которые характеризуют максимальное

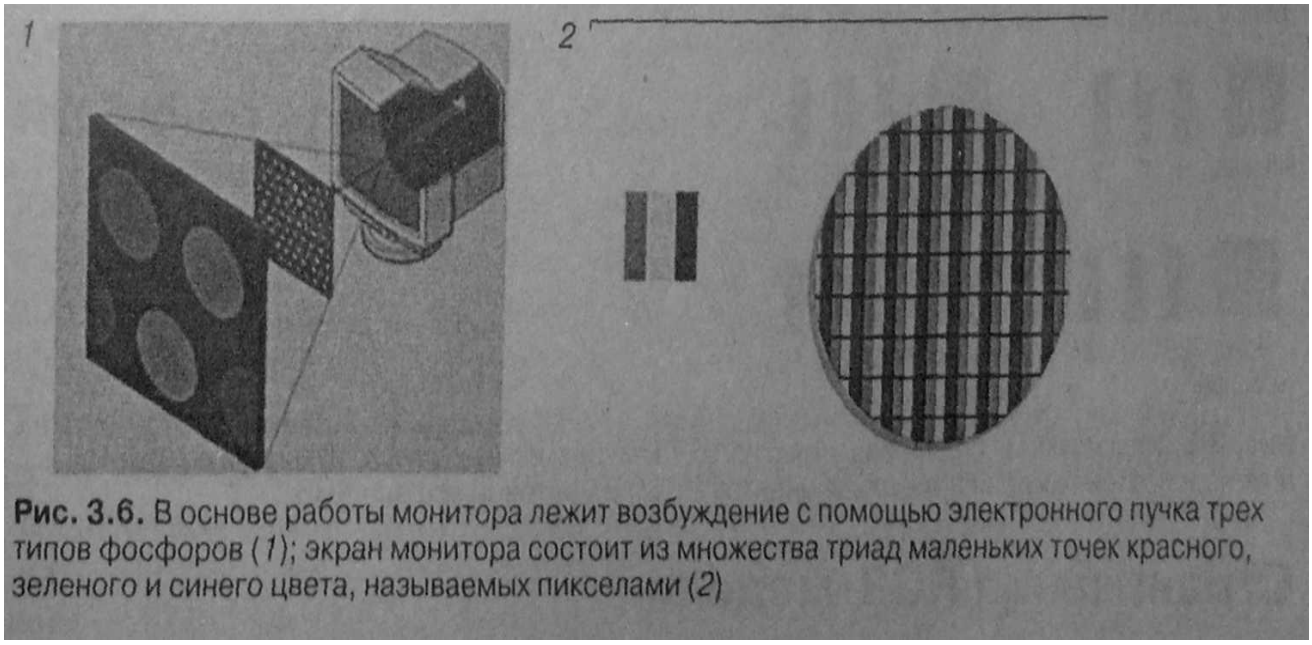


Рис. 3.6. В основе работы монитора лежит возбуждение с помощью электронного пучка трех типов фосфоров (*1*); экран монитора состоит из множества триад маленьких точек красного, зеленого и синего цвета, называемых пикселами (*2*)

Рис. 6.21. 1 - возбуждение поверхности монитора с помощью электронного пучка трех типов фосфоров; 2 - триады пикселов красного, зеленого и синего цветов.

количество воспроизводимых цветов. На рис. 6.22 приведена схема формирования 24-битового цвета, обеспечивающая возможность воспроизведения 256 × 256 × 256 = 16,7 млн. цветов.

Последние версии профессиональных графических редакторов (таких, как, например, CorelDRAW 9, Corel Photo-Paint 9, Photoshop 5.5) наряду со стандартной 8-битовой глубиной цвета поддерживают 16-битовую глубину цвета, которая позволяет воспроизводить 65 536 оттенков серого.

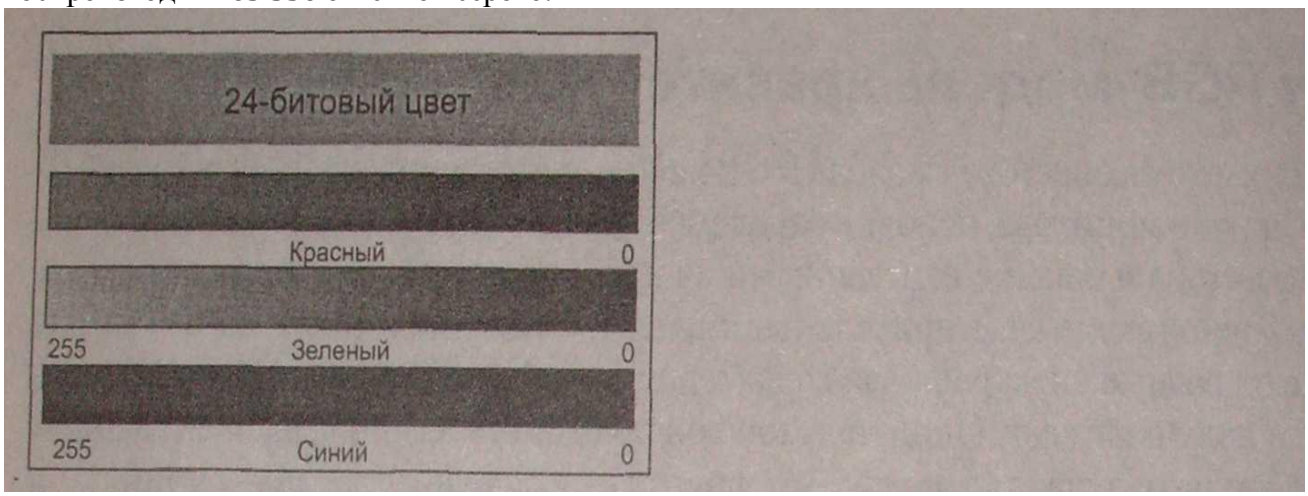


Рис. 6.22. Каждый из трех цветовых компонентов RGB - триады может принимать одно из 256 дискретных значений - от максимальной интенсивности (255) до нулевой, соответствующей черному цвету

На рис. 6.23 приведена иллюстрация получения с помощью аддитивного синтеза шести (из 16,7 млн.) цветов. Как уже упоминалось ранее, в случае, когда все три цветовые компоненты имеют максимальную интенсивность, результирующий цвет кажется белым. Если все компоненты имеют нулевую интенсивность, то результирующий цвет — чистый черный.

Рис. 6.23. Пример. Формирование 6 из 16,7 млн. возможных цветов путем вариации интенсивностей каждой из трех компонентов R, G и B цветовой модели rgb.

Ограничения RGB-модели

Несмотря на то что цветовая модель RGB достаточно проста и наглядна, при ее практическом применении возникают две серьезные проблемы:

• ограничение *цветового охвата*

Первая проблема связана с тем, что цвет, возникающий в результате смешения цветовых составляющих RGB элемента, зависит от типа люминофора. А поскольку в технологии производства современных кинескопов находят применение разные типы люминофоров, то установка одних и тех же интенсивностей электронных лучей в случае различных люминофоров приведет к синтезу разного цвета. Например, если на электронный блок монитора подать определенную тройку RGB-значений, скажем R = 98, G = 127 и B = 201, то нельзя однозначно сказать, каков будет результат смешивания. Эти значения всего лишь задают интенсивности возбуждения трех люминофоров одного элемента изображения. Какой получится при этом цвет, зависит от спектрального состава излучаемого люминофором света. Поэтому в случае аддитивного синтеза для однозначного определения цвета наряду с установкой триады значений интенсивностей необходимо знать спектральную характеристику люминофора.

Существуют и другие причины, приводящие к аппаратной зависимости RGB-модели даже для мониторов, выпускаемых одним и тем же производителем. Это связано, в частности, с тем, что в процессе эксплуатации происходит старение люминофора и изменение эмиссионных характеристик электронных прожекторов. Для устранения (или по крайней мере минимизации) зависимости RGB-модели от аппаратных средств используются различные устройства и программы градуировки. Цветовой охват (color gamut) — это диапазон цветов, который может различать человек или воспроизводить устройство независимо от механизма получения цвета (излучения или отражения).

Ограниченность цветового охвата объясняется тем, что с помощью аддитивного синтеза принципиально невозможно получить все цвета видимого спектра (это доказано теоретически!). В частности, некоторые цвета, такие как чистый голубой или чистый желтый, не могут быть точно воссозданы на экране. Но несмотря на то, что человеческий глаз способен различать цветов больше, чем монитор, RGB-модели вполне достаточно для

создания цветов и оттенков, необходимых для воспроизводства фотореалистических изображений на экране вашего компьютера.

sRGB — стандартизированный вариант RGB-цветового пространства

Как вы уже, очевидно, поняли, главный недостаток RGB-модели заключается в ее размытости. Это обусловлено тем, что на практике RGB-модель характеризует цветовое пространство конкретного устройства, например монитора или сканера.

Нужен какой-то общий знаменатель.

Тем не менее любое RGB-пространство можно сделать стандартным. Для этого надо всего лишь однозначно определить его. Например, в Photoshop 5 предлагается целых девять заранее определенных вариантов, важное место среди которых занимает стандартное цветовое пространство для Интернета — sRGB (так называемое standard RGB — стандартное RGB). По инициативе двух фирм — Microsoft и HP — оно стандартизировано и соответствует цветовому пространству типичного монитора VGA низшего класса. Сегодня это пространство является альтернативой системам управления цветом, использующим 1CC, предназначенные для описания цветового охвата устройств, которые входят в состав настольных издательских систем. В отличие от последних для пользователя Интернета важны простота и компактность файлов. Вряд ли вам понравится получать по сети двухмегабайтный (и даже двухкилобайтный) профиль с каждой картинкой (хотя спецификация ICC 1:1998-09 позволяет встраивать профили даже в изображения в формате GIF). Идея стандартного RGB-пространства настолько привлекательна, что даже Adobe Systems включила его в состав своих продуктов. Например, Photoshop 5.0 открывает RGB-файлы, не содержащие ICC-профиля, как sRGB.

6.4.2 Субтрактивные цветовые модели

В отличие от экрана монитора, воспроизведение цветов которого основано на излучении света, печатная страница может только отражать цвет. Поэтому RGB-модель в данном случае неприемлема. Вместо нее для описания печатных цветов используется модель CMY, базирующаяся на субтрактивных цветах (рис. 6.24).

Субтрактивные цвета в отличие от аддитивных цветов (той же RGB-модели) получаются вычитанием вторичных цветов из общего луча света. В этой системе белый цвет появляется как результат отсутствия всех цветов, тогда как их присутствие дает черный цвет.

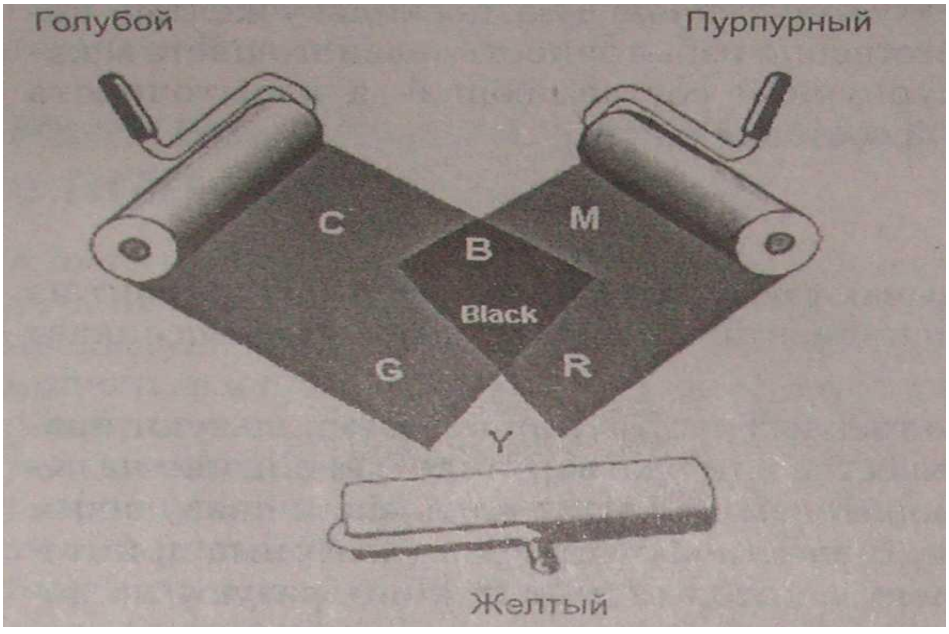


Рис. 6.24. Субтрактивная цветовая модель CMY

В последнее время в качестве синонима термина «субтрактивная» иногда используют термин «исключающая». Происхождение этого названия связано с явлением отражения света от покрытой красителем поверхности, а также с тем фактом, что при добавлении красителей интенсивность света уменьшается, поскольку свет поглощается тем больше, чем больше красителя нанесено на поверхность.

Нанесение на бумагу трех базовых цветов: голубого (Cyan), пурпурного (Magenta) и желтого (Yellow) позволяет создать множество субтрактивных цветов.

Физические процессы, лежащие в основе субтрактивного синтеза цвета, были рассмотрены ранее в разделе «Излученный и отраженный цвет» главы 2, «Основы работы с цветом». Поэтому здесь мы коснемся только некоторых деталей, позволяющих уточнить практические нюансы использования этой модели. Для этого нам потребуется записать соотношения, связывающие аддитивные (красный, зеленый, синий) и субтрактивные (голубой, желтый, пурпурный) цвета:

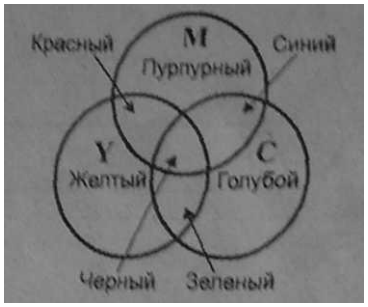
Зеленый + Синий = Голубой;
Зеленый + Красный = Желтый;
Красный + Синий = Пурпурный;
Зеленый + Синий + Красный = Белый;
Голубой + Желтый + Пурпурный = Черный.

Итак, что же происходит, когда на лист бумаги с нанесенным на него красителем падает белый свет? Если краситель голубой (сине-зеленый), то он поглощает из спектра красный цвет и отражает голубой. Соответственно пурпурный краситель поглощает комплиментарный ему зеленый цвет, а желтый краситель — синий цвет. Если при печати наложить друг на друга пурпурный и желтый цвета, то получится красный цвет, поскольку пурпурный краситель устранит зеленую составляющую, а желтый — синюю составляющую падающего цвета. Соответственно при печати с наложением всех трех субтрактивных цветов результирующий цвет будет черным.

На базе выполненных рассуждений можно сформулировать правило коррекции цветового разбаланса при цветной печати: если изображение имеет излишне синий оттенок, то следует увеличить желтую составляющую, поскольку желтый поглощает синие составляющие. Соответственно избыточность зеленого цвета можно скорректировать увеличением пурпурной составляющей, а избыточность красного цвета — увеличением голубой составляющей.

Цветовая модель CMY

Используется для описания цвета при получении изображений на устройствах, которые реализуют принцип ппоглощения цветов. В первую очередь, она используется в устройствах, которые печатают на бумаге. Название данной модели состоит из названий основных субтрактивных цветов:голубого (Cyan), пурпурного (Magenta) и желтого (Yellow) (рис.6.25).



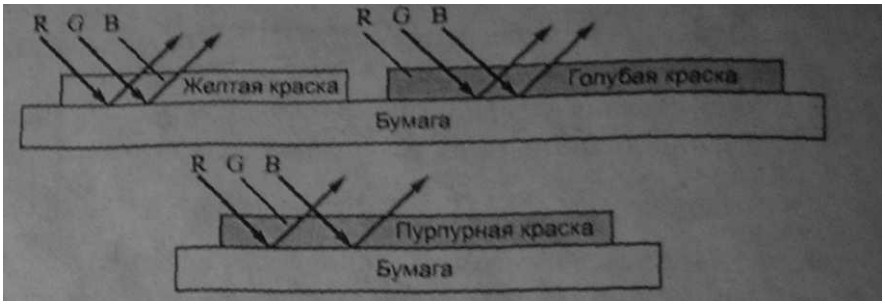


Рис. 6.25. Цветовая модель CMY - поглощение (вычитание) цветов)

Нанесение желтой краски на белую бумагу означает, что поглощается отраженный синий цвет. Голубая краска поглощает красный цвет. Пурпурная краска — зеленый. Комби-нированне красок позволяет получить цвета, которые остались — зеленый, красный, синий и черный. Черный соответствует поглощению всех цветов при отражении.

На практике добиться черного смешиванием сложно из-за нендеальности красок, поэтому в принтерах используют еще и краску черного цвета *(black).* Тогда модель называется *CMYK,* Необходимо также отметить, что не всякие краски обеспечивают указанное выше вычитание цветов CMY.

В таблице 6.1 для сравнения представим описание некоторых цветов в моделях RGB и CMY.

Таблица 6.1. Описание цветов в моделях RGB и CMY

Цвет	Модель RGB			Модель CMY		
	R	G	B	C	M	Y
Красный	1	0	0	0	1	1
Желтый	1	1	0	0	0	1
Ярко-Зеленый	0	1	0	1	0	1
Голубой	0	1	1	1	0	0
Синий	0	0	1	1	1	0
Пурпурный	1	0	1	0	1	0
Черный	0	0	0	1	1	1
Белый	1	1	1	0	0	0

Соотношение для перекодировки цвета из модели CMY в RBG:

$$\begin{bmatrix} r \\ g \\ b \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} c \\ m \\ y \end{bmatrix}$$

Здесь считается, что компоненты кодируются числами в диапозоне от 0 до 1. Для другого диапозона чисел можно записать соответствующие обратное соотношение.

CMY и CMYK

Существуют две наиболее распространенные версии субтрактивной модели: CMY и CMYK. Первая из них используется в том случае, если изображение или рисунок будут выводиться на черно-белом принтере, позволяющем заменять черный картридж на цветной (color upgrade). В ее основе лежит использование трех субтрактивных (вторичных) цветов: голубого (Cyan), пурпурного (Magenta) и желтого (Yellow). Теоретически при смешивании этих цветов на белой бумаге в равной пропорции получается черный цвет.

Однако в реальном технологическом процессе получение черного цвета путем смешивания трех основных цветов для бумаги неэффективно по трем причинам.

Невозможно произвести идеально чистые пурпурные, синие и желтые краски. Поэтому цвет получается не чисто черным, а грязно-коричневым.

На создание черного цвета с помощью модели CMY тратится в три раза больше краски.

В силу перечисленных факторов при печати чистого черного цвета используется добавка дополнительной черной компоненты цвета. Эта технология приводит также к улучшению качества теней и серых оттенков. Интенсивность каждой из четырех компонент цвета может изменяться в диапазоне от 0 до 100 %.

В аббревиатуре модели CMYK используется буква «К» (последняя буква слова Black) для того, чтобы избежать путаницы, поскольку в английском языке с буквы «В» начинается не только слово Black (черный), но и слово Blue (синий). Встречается еще один вариант трактовки использования этой буквы как аббревиатуры термина Key color (ключевой цвет).

Ограничения модели CMYK

CMYK-модель имеет те же два типа ограничений, что и RGB-модель: аппаратная зависимость; ограниченный цветовой диапазон.

В CMYK-модели также нельзя точно предсказать результирующий цвет только на базе численных значений ее отдельных компонентов. В этом смысле она является даже более аппаратно-зависимой моделью, чем RGB. Это связано с тем, что в ней имеется большее количество дестабилизирующих факторов, чем в RGB-модели. К ним в первую очередь можно отнести вариацию состава цветных красителей, используемых для создания печатных цветов. Цветовое ощущение определяется еще и типом применяемой бумаги, способом печати и, не в последнюю очередь, внешним освещением. Последнее неудивительно — ведь никакой объект не может отразить цвет, отсутствующий в источнике излучения.

В силу тот что цветные красители имеют худшие характеристики по сравнению с люминофорами, цветовая модель CMYK имеет более узкий цветовой диапазон по сравнению с RGB-моделью (рис. 6.26). В частности, она не может воспроизводить яркие насыщенные цвета, а также ряд специфических цветов, таких, например, как металлический или золотистый.

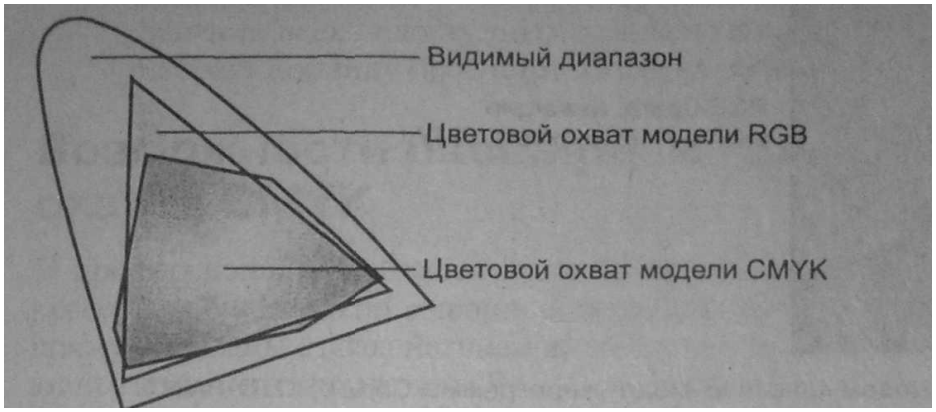


Рис. 6.26. Сопоставление цветовых охватов RGB и CMYK - моделей

Об экранных цветах, которые невозможно точно воссоздать при печати, говорят, что они лежат вне цветового охвата (gamut alarm) модели CMYK (рис. 6.26). В большинстве графических пакетов под такими цветами понимаются цвета, которые могут быть представлены в формате RGB или HSB, но при этом не имеют печатных аналогов в цветовом пространстве CMYK (рис. 6.27).

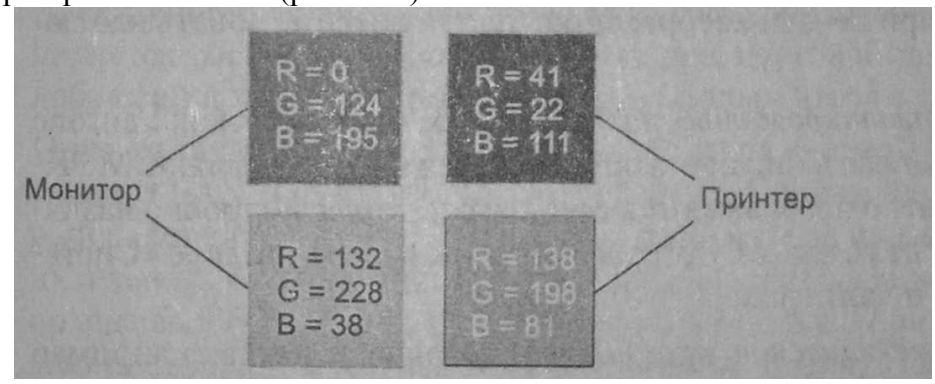


Рис. 6.27. Несовпадение цветов, отображанмых на экране монитора, с печатаемыми на принтере. При распечатке цвета становятся темными и приглушенными.

Несоответствие цветовых диапазонов RGB- и CMYK-моделей представляет серьезную проблему. Судите сами: полученная вами на экране монитора в результате напряженной работы прекрасная картинка при распечатке вдруг превращается в унылое и блеклое подобие оригинала. Для предотвращения подобной ситуации разработчиками графических программ предусмотрен комплекс специальных средств.

Наиболее простые основаны на выявлении и коррекции несоответствующих цветов непосредственно в процессе редактирования.

Более кардинальные предназначены для расширения цветового пространства CMYK-модели.

И наконец самый «продвинутый» — использование систем управления цветом — CMS (color management systems).

К первой группе средств, используемых при подготовке изображения для печати, можно отнести следующие.

Редактирование изображения в формате CMYK-модели. Хотя относительно целесообразности применения этого способа существуют прямо противоположные мнения, не вдаваясь в физические аспекты дискуссии, отметим, что полученное в этом случае при печати изображение будет соответствовать наблюдаемому на мониторе.

Возможности расширения цветового охвата CMYK

И профессионалы в области полиграфии, занимающиеся подготовкой и изданием красочных буклетов по живописи, и специалисты в области рекламы, чьи доходы напрямую связаны с воздействием цветных публикаций на покупателя, уже давно имеют претензии к стандартной CMYK-модели из-за относительно узкого диапазона воспроизводимых ею цветов. С помощью четырехцветной печати можно воспроизвести достаточно реалистичные красные цвета, но невозможно добиться ярких розовых, синих, фиолетовых и многих других цветов. Но даже те цвета, которые хорошо воспроизводятся с помощью этой модели, часто оказываются недостаточно насыщенными. По этой причине на базе CMYK-модели разработан ряд новых технологий.

Технология HiFi Color

К настоящему времени создано несколько вариантов HiFi Color. Их общей чертой является расширение используемых при цветовой печати гаммы цветов за счет добавления новых цветов к четырем базовым цветам CMYK.

Одна из таких цветовых систем разработана фирмой Pantone. Ее компьютерный вариант PANTONE® HEXACHROME$^{(TM)}$ Colors впервые введен в интегрированный пакет CorelDRAW 7. Палитра базируется на цветовой модели CMYK, дополнительно к четырем цветам которой добавлены два новых цвета: зеленый (G) и оранжевый (O). Это позволяет существенно расширить диапазон воспроизводимых цветов при офсетной печати и заметно поднять качество цветопередачи.

В настоящее время наряду с шестицветной цветовой системой фирмы Pantone реализованы и другие системы. Так, в системе HiFi Color3000 фирмы LinoTipe-Hell для получения ярких красных, зеленых и синих цветов используется семь цветов (три аддитивных RGB-модели и четыре субтрактивных цвета CMYK-модели).

Использование плашечных цветов

Плошечными (простыми, смесовыми) цветами называются цвета, которые воспроизводятся на бумаге готовыми смесовыми красками, созданными с помощью специальной технологии, базирующейся на использовании для каждого цвета соответствующего ему уникального красителя (чернил). Поскольку они в отличие от триадных (CMYK) цветов не прозрачны, то отражают свет поверхностным слоем. Это позволяет добиться

воспроизведения очень ярких тонов и специальных эффектов типа металлизации и иризации (перелива оттенков при разных углах зрения).

Плашечные краски используют вместо триадных (CMYK) красок или в добавление к ним. Несколько фирм занимаются производством таких цветов. Это в первую очередь Pantone, TRUMATCH и Focoltone. Более подробно плашечные цвета будут рассмотрены далее в разделе «Системы соответствия цветов и палитры».

На рис. 6.28 приведен пример сопоставления цветового охвата модели CMYK с цветами Pantone.

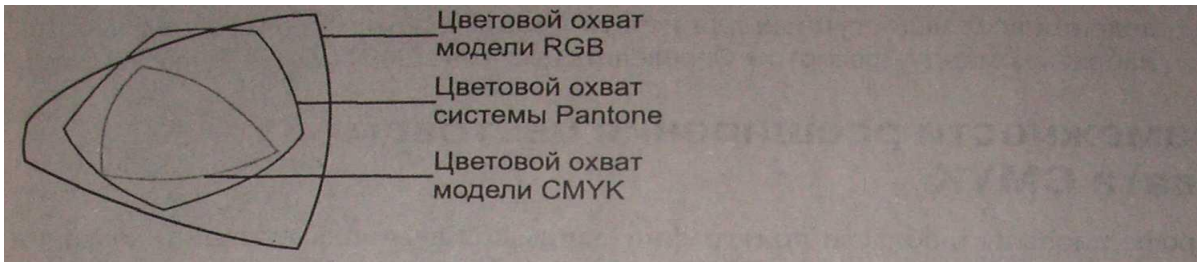


Рис. 6.28. Варианты расширения цветового охвата CMYK-модели путем использования технологии HiFi Color и плашечных цветов.

6.4.3 Перцепционные цветовые модели

Для дизайнеров, художников и фотографов основным инструментом индикации и воспроизведения цвета служит глаз. Этот естественный «инструмент» обладает цветовым охватом, намного превышающим возможности любого технического устройства, будь то сканер, принтер или фотоэкспонирующее устройство вывода на пленку.

Как было показано ранее, используемые для описания технических устройств цветовые системы RGB и CMYK являются аппаратнозависимыми. Это значит, что воспроизводимый или создаваемый с помощью них цвет определяется не только составляющими модели, но и зависит от характеристик устройства вывода. Для устранения аппаратной зависимости был разработан ряд так называемых перцепционных (иначе — интуитивных) цветовых моделей. В их основу заложено раздельное определение яркости и цветности. Такой подход обеспечивает ряд преимуществ:

- позволяет обращаться с цветом на интуитивно понятном уровне;
- значительно упрощает проблему согласования цветов, поскольку после установки значения яркости можно заняться настройкой цвета.

Прототипом всех цветовых моделей, использующих концепцию разделения яркости и цветности, является HSV-модель. К другим подобным системам относятся HSI, HSB, HSL и YUV. Общим для них является то, что цвет задается не в виде смеси трех основных цветов — красного, синего и зеленого, а определяется путем указания двух компонентов: цветности (цветового тона и насыщенности) и яркости.

Цветовая модель HSB

Модель HSB (Hue — цветовой тон, Saturation — насыщенность, Brightness — яркость) или ее ближайший аналог HSL представлены в большинстве современных графических пакетов. Из всех используемых в настоящее время моделей эта модель иаиболее точно соответствует способу восприятия цветов человеческим глазом. Она позволяет описывать цвета интуитивно ясным способом.

В HSB-модели все цвета определяются с помощью комбинации трех базовых параметров (рис. 6.29):

цветовой тон (H);

насыщенность(S);

яркость (B).

Рис. 6.29. Цветовой тон определяет положение цвета на цветовом круге, насыщенность и яркость задают количество «чистого» цвета и света в данном тоне.

Для лучшего понимания природы HSB-модели давайте познакомимся с физическим смыслом ее основных компонентов.

Цветовой тон

Как уже отмечалось, каждый реальный источник света воспроизводит его в виде смеси волн, имеющих разные длины. Под цветовым тоном (hue) понимается свет с доминирующей длиной волны. Обычно для описания цветового тона (в некоторых источниках применяется термин оттенок) используется название цвета, например красный, оранжевый или зеленый. В традиционной интерпретации этой модели каждый цветовой тон занимает определенное положение на периферии цветового круга и характеризуется величиной угла в диапазоне от 0 до 360° (рис. 6.30). Обычно для красного цвета берется угол 0°, для чисто зеленого -120° и для чисто синего — 240°.

На цветовом круге первичные цвета расположены на равном расстоянии друг от друга. Вторичные цвета находятся между первичными. В свою очередь, каждый цвет расположен напротив дополняющего его (комплиментарного) цвета, причем он находится между цветами, с помощью которых получен. Например, сложение желтого и голубого цветов дает зеленый. Таким образом, на цветовом круге зеленый цвет должен располагаться между желтым и голубым. Хотя оранжевый или фиолетовый не являются первичными или вторичными цветами (они представляются комбинацией первичного и вторичного цветов), они показаны на круговой диаграмме цветов, чтобы проиллюстрировать их положение относительно других цветов.

Однако само по себе понятие цветового тона не содержит всей полноты информации о цвете. Например, свет, в котором преобладает компонента с длиной волны около 450 нанометров, будет восприниматься большинством людей с нормальным зрением, как оттенок, обычно ассоциируемый с синим цветом (ему соответствует на цветовом круге угол 240°).

Чтобы усилить в изображении какой-либо цвет, нужно ослабить дополняющий его цвет (расположенный напротив него на цветовом круге). Например, чтобы изменить общее цветовое содержание изображения в сторону зеленого цвета, следует снизить в нем содержание пурпурного цвета. Именно на этом принципе основана цветовая коррекция изображения.

Рис. 6.30. Расположение цветов на цветовом круге

Вопрос в том, что понимать под понятием синий? Темно-синее или голубое небо лазурное море, полевой василек и незабудка — это все примеры цветов, в которых доминирует синий цвет, но, несмотря на это, они воспринимаются нашим глаза как разные. Что обусловливает их различие, как много или, наоборот, мало содержат они в своем составе других компонентов, которые наш глаз интерпретирует как составные части цвета? Этими дополнительными компонентами являются насыщенность (saturation) и яркость (brightness).

Насыщенность

Цветовой тон не единственный атрибут цвета, различаемый людьми. Другой компонент — насыщенность — характеризует чистоту цвета. Он определяет соотношение между основной, доминирующей компонентой цвета и всеми остальными длинами волн (количеством серого), участвующими в формировании цвета. Количественное значение этого параметра выражается в процентах от 0% (серый) до 100% (полностью насыщенный).

По другому определению, насыщенности отражает, насколько далеко отстоит данный цвет от равного с ним по яркости белого цвета. В этом случае насыщенность можно измерять числом едва заметных переходов (градаций), лежащих между данным цветом и белым.

Чем выше значение насыщенности, тем сильнее и яснее ощущается цветовой тон. Например, пастельный синий цвет воспринимается как размытый синий цвет из-за незначительного содержания в нем чистого оттенка. Снижение насыщенности приводит к тому, что цвет становится нейтральным, без четко выраженного тона.

Если вы возьмете цветную фотографию и понизите насыщение до 0%, то в итоге получите черно-белую фотографию (в градациях серого). Примерами цветов с максимальной насыщенностью могут служить спектральные цвета, в частности желтый цвет, соответствующий линии спектра натрия с длиной волны 536 мкм. В то же время желтый цвет, полученный путем аддитивного сложения красного с зеленым цветов, характеризуется пониженной насыщенностью. И совсем низкую насыщенность имеет желтый свет солнечного диска, содержащего практически полный спектр видимых цветов. Примерами «полностью» нейтральных (ахроматических) цветов могут служить серый, белый и черный цвета. По мере перемещения к центру круга цвет приближается к серому, поскольку при этом все базовые цвета смешиваются в равной пропорции.

Естественные цвета имеют низкую насыщенность, поэтому слишком насыщенные цвета выглядят ненатуральными и подчеркнутыми.

Перемещение поперек цветового круга (в отличие от движения по окружности) приводит к уменьшению доли цвета, от которого вы удаляетесь, и возрастанию доли цвета, к которому вы приближаетесь. В итоге это приводит к понижению насыщенности, которая имеет максимальное значение (100%) на поверхности окружности и минимальное (0%) — в центре круга.

Яркость

Яркость (B) характеризует интенсивность, с которой энергия света воздействует на рецепторы нашего глаза. Ее можно интерпретировать также как относительную освещенность или затемненность цвета (светлоту цвета). Солнечный зайчик — пример высокой интенсивности освещения (яркого). В то же время тлеющие угли — низкой. Любые цвета и оттенки независимо от их цветового тона можно сравнить по яркости, то есть определить, какой из них темнее, а какой светлее.

Яркость никоим образом не влияет на цветность, но от нее зависит, насколько сильно цвет будет восприниматься нашим глазом. При нулевой яркости мы не видим ничего, поэтому любой цвет будет восприниматься как черный. Исходя из этого яркость иногда трактуют подобно насыщенности, то есть как величину, обратную степени разбавленности цвета черным. В этом случае при отсутствии черного мы получаем чистый спектральный цвет, а максимальная яркость вызывает ощущение ослепительно белого цвета.

Когда говорят о яркости как атрибуте цвета, под белым цветом понимают абсолютную яркость, а под черным цветом — полное отсутствие яркости. Серый цвет характеризует промежуточное значение яркости.

Ахроматические цвета, то есть белые, серые и черные, характеризуются только яркостью. Это проявляется в том, что одни цвета темнее, а другие светлее.

Величина яркости измеряется в процентах в диапазоне от 0% (черный) до 100% (белый). По мере снижения процентного содержания яркости цвет становится темнее, стремясь к черному. Данная компонента является нелинейной, что соответствует нашему восприятию светлых и темных цветов.

Яркость и цветовой тонне являются полностью независимыми параметрами. Изменение яркости изображения влияет на изменение цветового тона, что создает нежелательный цветовой отлив (сдвиг) в изображении. Так, при значительном уменьшении яркости зеленые цвета синеют, синие приближаются к фиолетовым, желтые — к оранжевым, а оранжевые — к красным. Сильное увеличение яркости излучения вызывает другой эффект. Красные цвета переходят в оранжевые, затем в желтые и, наконец, — в белые.

Универсальность яркостной компоненты

Яркость (светлота) — качество, присущее как хроматическим, так и ахроматическим цветам. Поэтому по яркости можно сравнивать между собой любые цвета и оттенки: бледно-зеленый с темно-зеленым, розовый с синим, красный с фиолетовым и т. д. Это свойство находит использование при конвертировании цветных изображений в черно-белые или полутоновые.

У художников принято светлотные отношения называть тональными. Поэтому различают светлотный и цветовой тон. Когда говорят, что картина написана в светлых тонах, то прежде всего имеют в виду светлотные отношения, а по цвету она может быть и серо-белой, и розовато-желтой, и светло-сиреневой — словом, самой разной.

Различие между HSB- и HSL-моделями состоит в замене нелинейного компонента brightness (яркость) на линейный компонент lightness (светлота).

Достоинства и ограничения HSB-модели

Модель HSB в отличие от моделей RGB и CMYK носит абстрактный характер. Отчасти это связано с тем, что цветовой тон и насыщенность цвета нельзя измерить непосредственно. Любая форма ввода цветовой информации всегда начинается с определения красной, зеленой и синей составляющих, на базе которых затем с помощью математического пересчета получают компоненты HSB-модели. В результате эта цветовая модель имеет то же цветовое пространство, что и RGB-модель, а значит, и присущий ей недостаток — ограниченное цветовое пространство.

Вместе с тем HSB-модель обладает по сравнению с RGB- и CMYK-моделями двумя важными преимуществами:

Аппаратной независимостью. Задание составляющих этой модели в виде значений цветового тона, насыщенности и яркости позволяют однозначно определить цвет без необходимости учета параметров устройства вывода.

Более простым и интуитивно понятным механизмом управления цветом.
Это связано с тем, что цветовой тон, насыщенность и яркость представляют собой независимые характеристики цвета. Например, чистый красный цвет расположен на цветовом круге под углом 0°. Если нужно сместить красный тон к оранжевому тону, то следует лишь несколько увеличить угол, определяющий цветовой тон. Для получения более блеклого цвета достаточно лишь снизить насыщенность, а для придания ему большей яркости соответственно увеличить значение яркости. Получение таких эффектов с помощью RGB-модели практически невозможно, поскольку значения ее цветовых компонентов очень сильно зависят друг от друга. Поэтому при изменении одной из ее составляющих, например красной, это окажет влияние не только на цветовой тон, но одновременно и на насыщенность и яркость.

6.4.4 Системы соответствия цветов и палитры

Как уже отмечалось ранее при рассмотрении цветовых моделей, каждая из них характеризуется собственным цветовым охватом. Это приводит к тому, что часть цветов, используемых в технологии многослойной печати, не может быть точно отображена на экране монитора. Кроме того, на воспроизведение цвета на экране монитора влияет множество других факторов: условия освещенности, срок эксплуатации, точность его настройки. Поэтому нельзя выбирать нужный нам цвет непосредственно на экране дисплея.

С целью повышения точности воспроизведения цвета на этапе печати в современные графические программы включены системы сопоставления цветов и палитры, которые предоставляют в ваше распоряжение еще один способ назначения
цветов, альтернативный цветовым моделям.

Системы соответствия цветов

Для упрощения процедуры идентификации цвета ведущими фирмами, специализирующимися в области полиграфии и производстве красителей, были созданы системы соответствия цветов.

Система соответствия цветов включает в себя набор следующих основных компонентов:

Эталонные таблицы (атласы или каталоги) цветов, содержащихся в одноименных палитрах.

Электронные палитры (или просто палитры).

Специальные программные и аппаратные средства для калибровки устройств вывода.

Назначение эталона

Эталонные таблицы предоставляют собой набор цветов (образцов), которые могут быть адекватным образом отображены в процессе печати на соответствующей им бумаге.

Изготовление эталона тщательно контролируется с целью минимизации вариаций цветов. Каждому цвету присваивается свое уникальное имя и указывается тип пигмента или состав смеси из различных пигментов, необходимых для его реализации. Указывается также идентифицированный с данным пигментом тип бумаги. В дополнение к этой таблице, используемой как справочник, пользователь получает образцы цветов, которые можно вырезать и прикрепить к изображению. Благодаря этим образцам система обеспечивает точный визуальный контроль соответствия того, что мы видим на экране, с тем, что мы получим на печати. Типичными примерами атласов цветов (или, как их еще называют, цветовых образцов) являются каталоги фирм TRUMATCH и Pantone, известные под названиями Colorfinder и Process Color Guide (рис. 6.31).

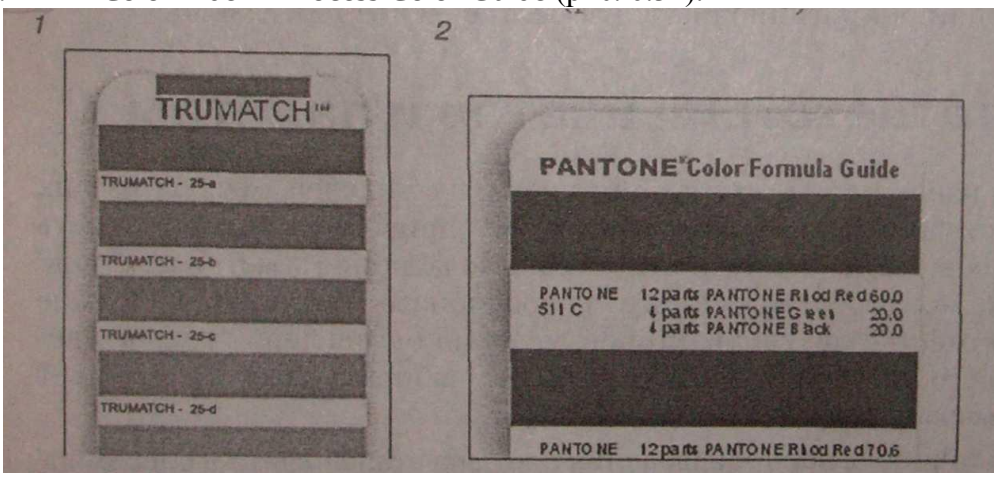


Рис. 6.31. Примеры оформления эталонных образцов цветов фирм TRUMATCH и Pantone

Вы можете выбрать из них нужные вам цвета, затем определить соответствующее им процентное содержание каждого из компонентов CMYK-модели и быть уверенными, что они точно отобразятся при печати (даже если цвет на экране не соответствует цвету выбранного вами образца).

Каждая из рассмотренных систем соответствия цветов имеет два варианта атласов образцов с одними и теми же CMYK-цветами, напечатанными на мелованной и немелованной бумаге.

Реальность такова, что цвет, напечатанный на немелованной бумаге, выглядит более темным и приглушенным по сравнению с аналогичным цветом, напечатанным на мелованной бумаге. Поэтому если вы собираетесь использовать при печати оба вида бумаги, вам понадобятся два каталога цветовых образцов.

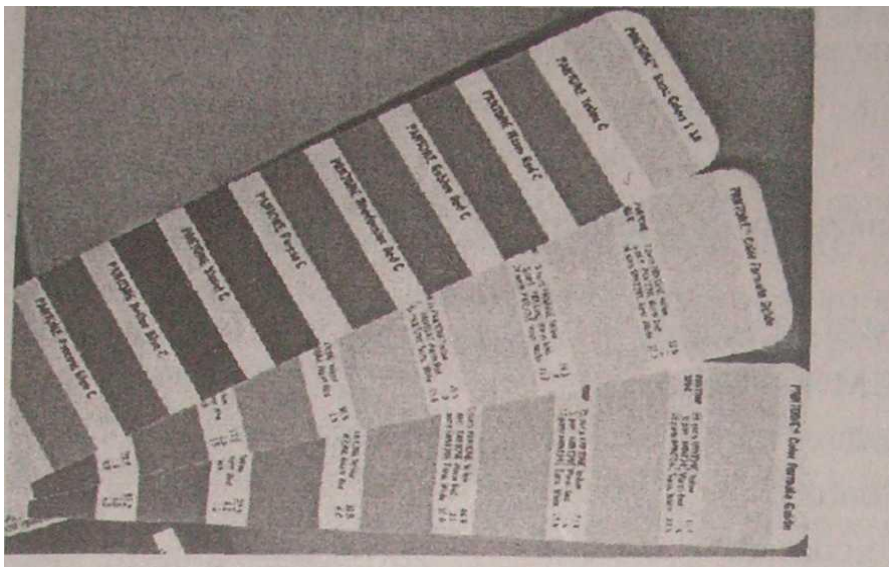


Рис. 6.32. Справочники с цветовыми образцами фирмы Pantone.

Итак, можно выбрать цвет в изображении и визуально сопоставить его с образцом, взятым из эталонной таблицы (рис. 6.33).

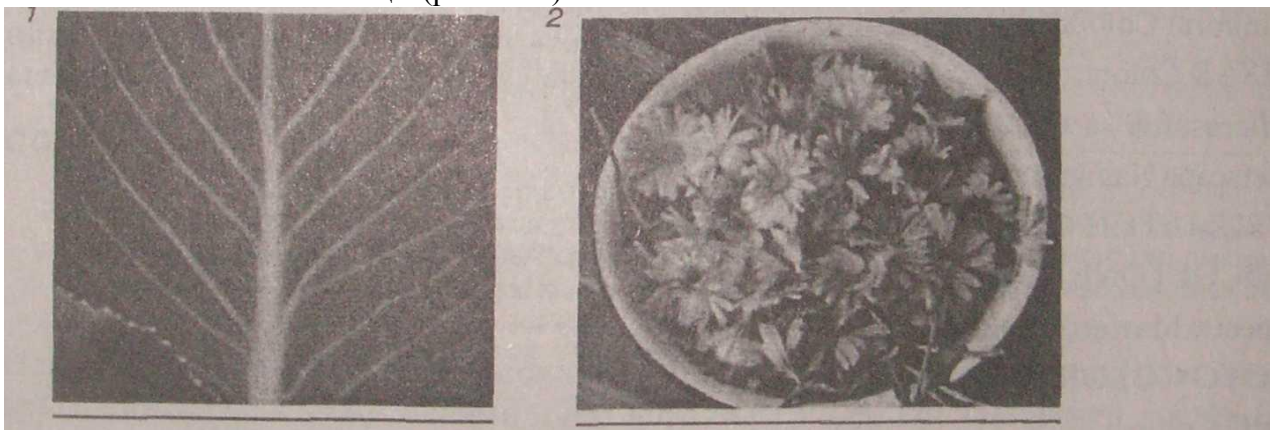


Рис. 6.33. Примеры задания цвета изображения с использованием системы Pantone.

В современных программах графики, таких как программа CorelDRAW, электронные палитры систем соответствия цветов поставляются вместе с высококачественными копиями цветных каталогов.

Кодирование цвета. Палитра

Для того чтобы компьютер имел возможность работать с цветными изображениями, необходимо представлять цвета в виде чисел — кодировать цвет. Способ кодирования зависит от цветовой модели и формата числовых данных в компьютере.

Для модели RGB любой из компонентов может быть представлен числами, ограниченными определенным диапазоном — например, дробными числами от 0 до 1, или целыми числами от 0 до какого-либо максимального значения. В настоящее время довольно распространен формат True Color, в котором каждый компонент представлен в виде банта, что дает 256 градаций для любого компонента: R - 0 ... 255, G = 0 ... 255, B = 0 ... 255. Количество цветов составляет 256x256x256 = 16.7 млн. (2^{24}).

Такой способ кодирования цветов можно назвать компонентным. В компьютере коды изображений True Color представлены в виде троек байтов или упаковываются в длинное целое (четырехбайтовое) — 32 бита (так, например, сделано в API Windows):

C = 00000000 bbbbbbbb gggggggg rrrrrrrr.

При работе с изображениями в системах КГ часто приходится искать компромисс между качеством изображения (требуется как можно больше цветов) и ресурсами, необходимыми для хранения и воспроизведения изображения. Ресурсы исчисляются, например, объемом памяти (нужно уменьшать количество бит на пиксел).

Кроме того, некоторые изображения сами по себе могут использовать ограниченное количество цветов. Например, для черчения может быть достаточно двух цветов, для фото человеческого лица важны оттенки красного, розового, желтого, пурпурного, зеленого; а для неба — оттенки голубого и серого. В таких случаях использование полноцветного кодирования цвета является избыточным.

При ограничении количества цветов используют палитру, представляющую набор цветов, важных для данного изображения. Палитру можно воспринимать как таблицу цветов. Палитра устанавливает взаимосвязь между кодом цвета и его компонентами в выбранной цветовой модели.

Недостатком такой палитры можно считать отсутствие оранжевого цвета — этот цвет является одним из семи основных цветов радуги. Существуют также другие стандартные палитры, например, 256-цветная для VGA. Компьютерные видеосистемы обычно предоставляют возможность программисту установить собственную палитру.

Каждый цвет изображения, используя палитру, кодируется индексом, которой будет определять номер строки в таблице палитры. Вот почему такой способ кодирования цвета называется индексным.

6.4.5 Триадные и плашечные цвета

Для печатания результатов работы, выполненной вами в графической программе на полиграфическом оборудовании, возможно использование одной из двух схем печати: плошечной или многослойной.

Шашечными (или простыми) цветами называются цвета, которые воспроизводятся на бумаге готовыми смесовыми красками.

Каждый плашечный цвет репродуцируется с помощью отдельной печатной формы (плашки).

Многослойная печать основана на использовании триадных (иначе составных) цветов и включает в себя как минимум четыре процесса.

Триадные цвета воспроизводятся путем смешивания в разных пропорциях триадных красок (голубой, пурпурной, желтой), применяемых в стандартной четырехкрасочной печати.

В графических программах все цветовые модели работают именно с триадными цветами. Поэтому воспроизведение плашечного цвета на экране монитора с помощью, например, цветовой модели RGB приводит к аппроксимации плашечного
цвета триадным цветом.

Плашечная схема печати применяется тогда, когда количество цветов в рисунке меньше четырех или когда отдельные цвета не могут быть получены путем смешивания красок (например, неоновые или имитирующие металлизированную поверхность).

В случае необходимости прецизионного воспроизведения цвета или создания специальных цветовых эффектов возможны реализация плашечной печати с большим количеством цветов или совмещение плашечной и многослойной печати.

Некоторые плашечные цвета можно точно передать с помощью триадных красок, другие находятся за пределами цветового охвата CMYK. Например, пастельные, неоновые или металлизированные краски не имеют аналогов в цветовой системе CMYK, а оттенок зеленого цвета легко заменить его составным аналогом.

О различии в физических механизмах воспроизведения цвета плашечными и триадными цветами

Различие между плашечными и триадными цветами напрямую связано с процессами взаимодействия света с чернилами, используемыми для создания этих красок.

Чернила для плашечной печати непрозрачны, поэтому они отражают свет поверхностным слоем. В результате для получения на бумаге, например, пурпурного цвета потребуется использование пурпурных чернил. Это позволяет, в свою очередь,
добиваться очень ярких тонов и специальных эффектов типа металлизации и ирилизации (перелива оттенков).

Чернила для многослойной печати, наоборот, прозрачны. Поэтому свет отражается не их поверхностным слоем, а поверхностью материала, на который они нанесены. Это приводит к тому, что образование цвета происходит за счет удаления из спектра лишних компонентов путем поглощения их слоем краски. В результате для воспроизведения пурпурного цвета на поверхность страницы необходимо наложить два типа чернил — бирюзового и синего цветов. Они поглотят синюю и зеленую части спектра, оставив (отразив) для нашего глаза только пурпурную часть спектра.

Палитры PANTONE

Фирмой PANTONE создана одна из самых больших и полных систем соответствия цветов. Она первой получила международный статус системы стандартизации цветов и пока остается доминирующей на рынке полиграфической продукции. Система соответствия цветов PANTONE возникла задолго до того, как она стала использоваться в качестве цветной электронной палитры при создании компьютерных изображений.

В последнее время фирма PANTONE увеличила число цветовых наборов, что вызвано необходимостью использования в дизайне металлических, текстильных и пастельных тонов, а также цветов, входящих в новое шестимерное цветовое пространство PANTONE Hexachrome.

Это нашло отклик и в соответствующем обеспечении графических программ. Например, в девятой версии CorelDRAW значительно расширено число палитр PAN-TONE. Это достигнуто за счет введения двух вариантов систем соответствия цветов — для печати на мелованной к немелованной бумаге. Тенденция коснулась и палитры PANTONE® Hexachrome, преобразованной в новой версии в две палитры. Одновременно добавлены электронные аналоги ряда новых палитр PANTONE для создания металлических и пастельных цветов. В итоге девятая версия пополнилась двумя новыми системами соответствия цветов и пятью новыми палитрами PANTONE:

PANTONE MATCHING SYSTEM® Coated;
PANTONE MATCHING SYSTEM® Uncoated;
PANTONE® Hexachrome® Coated;
PANTONE® Hexachrome® Uncoated;
PANTONE® Metallic Colors; Ш
PANTONE® Pastel Colors Coated;
PANTONE® Pastel Colors Uncoated.

Это привело к увеличению количества поставляемых стандартных цветов до 10 ООО с одновременным расширением цветового охвата устройств вывода на печать по сравнению с моделью CMYK.

Включенные в графические пакеты электронные палитры плашечных и триадных цветов фактически представляют лишь малую часть соответствующих оригинальных палитр.

Задание нужного цвета: PANTONE MATCHING SYSTEM

В случае использования при печати дополнительной краски помимо стандартных красок модели CMYK необходимо однозначно указать этот цвет. Определение типа «очень светлый зеленый с легким оттенком цвета морской волны» может казаться вам необычайно точным, но для печатника это будет что-то типа разновидности
серого.

Промышленным стандартом является система соответствия цветов Pantone Matching System, представляющая собой собранные в группы пронумерованные образцы

примерно тысячи цветов. Некоторые из них будут соответствовать описанию зеленого с оттенком цвета морской волны. Поэтому нужно заглянуть в каталог образцов, позвонить в типографию, сказать «PMS 3258» или что-то в этом роде и повесить трубку. Хотя на полке в печатном цехе может и не быть краски точно такого цвета, система Pantone дает четкую инструкцию, как получить этот цвет из комбинации других, более распространенных красок. Просто и эффективно как для типографии, так и для клиента.

Проблемы могут возникнуть только в том случае, если вы выберете цвет из палитры PMS, а потом попытаетесь синтезировать его красками CMYK вместо того, чтобы использовать отдельную (плашечную) краску в качестве пятого цвета. Для того чтобы вы могли получить визуальное представление о соответствии между плашечным цветом и ближайшим к нему CMYK-аналогом, в палитре Pantone представлены два варианта каждого цвета. Один показывает специальную смесь красок (плашечный цвет), а другой — ближайший к ней эквивалент CMYK.

В общем случае следует помнить, что реальный (плашечный) цвет значительно более интенсивный. Это связано с тем, что он сплошной в отличие от наложения двух растров бледной краски триадного (CMYK) цвета на белую бумагу. Голубая краска, используемая в печати, очень бледная, поэтому синий, зеленый или фиолетовый цвет из палитры Pantone невозможно получить с помощью красок CMYK -можно получить оттенок цвета, но не более.

И здесь выбор за вами, поскольку только вы можете определить, насколько аналог соответствует реальному цвету.

Система соответствия цветов FOCOLTONE

Палитра FOCOLTONE представляет собой систему создания плашечных цветов из основных цветов модели CMYK. Эта палитра позволяет оптимизировать процесс печати созданных изображений за счет организации ловушек цвета. Ловушки — это способ устранения ошибок совмещения окрашенных областей за счет некоторого увеличения размеров совмещаемых областей, приводящих к их перекрытию. Цвета FOCOLTONE организованы таким образом, чтобы новый цвет можно было реализовать из предыдущего путем добавления к нему не менее 10% одного из основных цветов. Это минимизирует необходимость использования фильтров и оптимизирует процедуру цветоделения.

Система соответствия цветов TRUMATCH

Палитра TRUMATCH содержит свыше 2000 основных цветов, поддерживаемых большинством принтеров.

Trumatch создала специальный электронный эквивалент своей системы цветов на базе компьютерной электронной системы (CEPS). Эта система основана на цветовой модели CMYK, и поэтому при печати здесь не требуется создания дополнительных слоев цветоделения. В цветовой палитре цвета расположены в следующем порядке: сначала по цвету (от красного до фиолетового), затем контрастности (от глубоких до пастельных тонов) и, наконец, яркости (добавление или удаление черного). Для отображения цветовых ячеек, не представленных в окне палитры, можно воспользоваться полосой прокрутки.

В настоящее время система соответствия цветов TRU MATCH внедрена в продукты ведущих фирм-разработчиков программного обеспечения компьютерной графики, включая Adobe, Corel, Macromedia и Quark.

Цветовая палитра SpectraMasfer

Палитра триадных цветов SpectraMaster разработана фирмой DuPont для использования в производстве промышленных покрытий и красителей. Она содержит свыше 2400 различных цветов и основана на цветовой модели Lab. Для ее изображения на экране монитора используется RGB- или CMYK-модель, а для печати — CMYK-модель.

Цвета из палитр DIC и палитры TOYO 88

Эти палитры цветов широко распространены в странах Азии, особенно в Японии. Каждая из них имеет собственную систему кодировки цветов и коллекцию из основных

цветов. Цвета из палитр DIC (DIC Color Guide, DIC Color Guide Part II и DIC Traditional Colors of Japan) создаются при смешивании красок DIC. В Corel PHOTO-PAINT и других приложениях Corel воспроизведение цветов этой палитры выполняется в модели CMYK.

Диапазон цветов палитры TOYO 88 создается и воспроизводится с помощью красок TOYO. Репродуцирование цветов этой палитры на экране компьютера осуществляется с помощью RGB-модели, а для печати — CMYK-модели. Для согласования этих двух моделей используется модель Lab.

Специализированные палитры

Большинство современных графических программ позволяет использовать поставляемые в составе пакета специализированные палитры и создавать свои собственные палитры. Например, в программе CorelDRAW специализированные (пользовательские) палитры выделены в отдельную группу палитр Custom Palettes (пользовательские палитры). Из нее вы можете загрузить для текущей работы любую нужную вам палитру. Это и краски осени (rAutunmn.cpl), и всевозможные оттенки листвы (rFoliage.cpl), и лики любви (rLove.cpl), и многое, многое другое, что, безусловно, удовлетворит вкус самых взыскательных художников.

В отличие от стандартных палитр, имеющих жестко определенный набор цветов, вы можете не только свободно обновлять и пополнять цветовой состав пользовательских палитр, но и создавать собственные палитры под конкретную задачу. Например, вы можете создать собственную палитру на базе цветовой палитры
любого изображения или выделенной его части.

Эти палитры могут включать в себя как плашечные цвета, так и цвета, созданные с помощью цветовых моделей или путем смешивания.

6.4.6 Цветовые режимы

Цветовые режимы представляют собой практическую реализацию рассмотренных выше цветовых моделей. В большинстве графических программ только три цветовые модели — RGB, CMYK и Lab — имеют одноименные цветовые режимы. Вместе с тем в них широко представлены режимы с ограниченной цветовой палитрой.

Наиболее широким (и практически идентичным) охватом цветовых режимов характеризуются программы Adobe Photoshop и Corel PHOTO-PAINT.

Режим черно-белой графики

Художники и разработчики программного обеспечения иногда называют этот режим монохромной графикой, растровой графикой (bitmap art), или графикой с однобитовым разрешением.

Для отображения черно-белого изображения используются только два типа ячеек: черные и белые (рис. 6.34). Поэтому для запоминания каждого пикселя требуется только 1 бит памяти компьютера. Областям исходного изображения, имеющим промежуточные оттенки, назначаются черные или белые пиксели, поскольку других оттенков для этой модели не предусмотрено. В качестве аналога бинарного узла вы можете представить лампочку, которая может находиться только в одном из двух состояний: вкл или выкл. При такой кодировке цвет пикселя также может принимать только одно из двух состояний: черный или белый.

Рис. 6.34. Интерпретация двоичной 1-битовой информации

Этот режим можно использовать для работы с черно-белыми изображениями, полученными сканированием черно-белых чертежей и гравюр, а также иногда при выводе цветных изображений на черно-белую печать.

Остановимся на назначении и особенностях организации каждого из перечисленных типов монохромных изображений.

Line Art (Гравюра)

Этот вид монохромного черно-белого изображения характеризуется высоким контрастом изображения, что связано с отсутствием полутонов. При конвертировании в этот тип изображения все цветные пикселы, формирующие изображение, преобразуются только в черные и белые. В качестве критерия такого преобразования используется настраиваемый параметр Threshold (Порог). Цвета, яркость которых ниже установленного порогового значения, преобразуются в черный цвет. В противном случае происходит преобразование в белый цвет.

Методы, основанные на алгоритмах формирования случайных узоров

В современных графических программах для эмуляции оттенков серого широко используются алгоритмы, основу которых составляет генерация случайных узоров на базе наборов черных и белых пикселов.

Ordered (Упорядоченный)

В отличие от рассмотренной выше группы методов в этом варианте для эмуляции оттенков серого используются фиксированные растровые узоры. Поэтому данный метод имеет достаточно высокое быстродействие.

Метод Cardinality-Distribution

Данный метод создает текстуроподобное изображение путем анализа и преобразования атрибутов каждого пиксела изображения.

Halftone (Полутон)

Такой способ реализации изображения базируется на специфике восприятия изображения человеческим глазом, для которого область изображения, заполненная крупными точками, ассоциируется с более темными тонами и, наоборот, область, заполненная точками меньшего размера, воспринимается как более светлая. Режим Halftone поддерживается большинством принтеров.

Полутоновые изображения представляют собой однобитовые изображения с непрерывным тоном, которые реализуются с помощью конгломерата точек разного размера и формы.

В полученном таким образом изображении оттенки серого имитируются точками разного размера, внесенными в специальный шаблон, форму которого можно выбрать из раскрывающегося списка Screen type (Тип растра).

Режим Grayscale (Градации серого)

Использование режима Grayscale (Градации серого) позволяет увеличить информационную емкость изображения за счет повышения цветового разрешения каждого пиксела. Поскольку в этом режиме на каждый пиксел выделяется до 8 бит, то требуется иная форма организации информации по сравнению с ранее рассмотренными однобитовыми монохромными режимами. Если, как уже отмечалось, режим Черно-белая графика может быть сравним с элементарной математикой, в которой основной элемент графического изображения — пиксел — может принимать только два состояния: включен и выключен, то режим Градации серого — это высшая математика, позволяющая оперировать с комбинацией до 256 оттенков, обеспечивая более высокое тоновое разрешение изображения.

Это связано с тем, что устройства, использующие двоичную математику, сводят все многообразие явлений к комбинации вариантов, количество которых равно числу 2 в соответствующей степени. Для пиксела с 4-битовым разрешением число возможных вариантов составит 2^4, что соответствует 16 комбинациям. В случае 8-битового разрешения это число возрастет до 2^8, или 256 комбинаций. Именно такое количество оттенков может быть реализовано при сканировании изображения в режиме Оттенки серого большинством непрофессиональных сканеров. Растровые редакторы воспринимают полученное в этом режиме цифровое изображение в виде одноцветного (монохромного) канала, содержащего 256 различных уровней яркости.

Для организации информации в режиме Градации серого используется один цветовой канал, который при работе с Corel PHOTO-PAINT называется серым каналом, а с Adobe Photoshop — альфа-каналом.

Напомним, что канал — это изображение, сформированное в режиме Градации серого. Следует подчеркнуть, что понятие канала является очень важным средством, используемым при редактировании изображений с помощью растровых редакторов. Более подробно вопросы организации и использования цветовых каналов будут рассмотрены далее в разделе этой главы «Режим RGB Color» и в главах, посвященных знакомству с работой растровых редакторов.

С технической точки зрения монохромное изображение, содержащее гамму из 256 оттенков серого, перекрывает весь диапазон оттенков от черного до белого, создавая непрерывную для глаза шкалу. Поэтому для получения монохромного изображения, близкого к оригиналу, при сканировании изображения можно использовать режим

Градации серого.

Последние версии профессиональных редакторов, включая Adobe Photoshop и Corel PHOTO-PAINT, наряду со стандартной 8-битовой глубиной цвета полутоновых изображений поддерживает 16-битовую глубину цвета, которая позволяет воспроизводить 65 536 оттенков серого.

Художественная ценность черно-белого изображения определяется композицией и световой контрастностью. Многие прекрасные цветные изображения плохо смотрятся при преобразовании их в черно-белые из-за одинаковой световой тональности разных цветов. Для получения хороших результатов в режиме Градации серого нужно использовать монохромный источник с высокой контрастностью.

Режим Duotone (Дуплекс)

Дуплекс — это 8-разрядный цветовой режим, использующий 256 оттенков не более четырех цветовых тонов.

Фактически дуплексную цветовую модель можно рассматривать как изображение в цветовой модели Grayscale, улучшенное с помощью дополнительных цветов (от одного до четырех). В дуплексном цветовом режиме изображение состоит из 256 оттенков одной (Monotone, тоновое), двух (Duotone, двухтоновый дуплекс), трех (Tritone, тритон) или четырех (Quadtone, квадртон) красок.

Двухтоновый вариант данной цветовой модели широко распространен в полиграфии.

Здесь в качестве дуплекса используется модифицированное изображение в градациях серого, отпечатанное с помощью красок двух цветов - как правило, черного и акцентирующего цвета, хотя могут использоваться любые другие два цвета. В общем случае этот термин относится также к дуплексам с тремя и четырьмя красками.

Использование двух красок вместо четырех значительно сокращает расходы на печать, обеспечивая вместе с тем широкий диапазон выбираемых оттенков. Дуплекс идеален для добавления акцентирующего цвета к фотографии или расширения тонального диапазона красителей.

Этот режим можно использовать для того, чтобы придавать цветность черно-белым изображениям либо создавать интересные эффекты с помощью различных параметров тонирования.

Режим RGB Color

Данный режим часто называют RGB-цветом. Он наиболее удобен для редактирования изображений на экране компьютера, так как обеспечивает цветовое разрешение 24 бит/пиксел. Это позволяет использовать для реализации цветных цифровых изображений палитру из 16,7 млн цветов.

На жаргоне программистов цветовую модель RGB называют естественным щеглом (true color), так как 16 млн цветов, доступных при такой глубине цвета, достаточно для представления всех различимых человеческим глазом оттенков.

Очевидно, что для источников изображения, имеющих ограниченную цветовую палитру, такое количество цветовых оттенков может оказаться избыточным.

Назначение каналов

Канал — это 8-разрядный монохромный вариант изображения, содержащий информацию об этом изображении. В программах растровой графики применяются каналы двух типов: каналы выделения (называемые также альфа-каналами) и цветовые.

Цветовые каналы генерируются в растровых редакторах автоматически при созданий или открытий изображения. У каждого компонента цветовой моделиизображения имеется свой цветовой канал. Например, у RGB-изображения три раздельных цветовых канала — красный (Red), зеленый (Green) и синий (Blue) — по одному для каждого цветового компонента. При открытии нового (пустого) изображения с белым цветом фона каждый из каналов заполнен соответствующим однородным цветом максимальной интенсивности (255), объединение которых в составной канал и дает белый цвет.

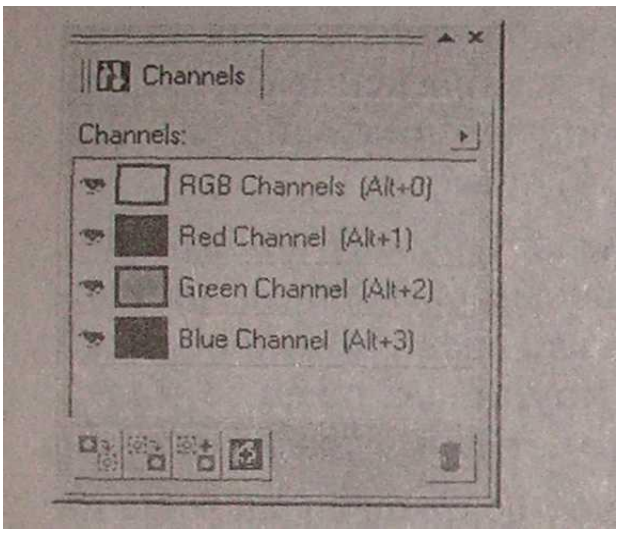


Рис. 6.35. Назначение каналов.

Каналы несут информацию о том, сколько красного, зеленого или синего цвета должно содержаться в каждом пикселе изображения для образования соответствующего

цвета. Каждый канал имеет 8-битовое разрешение и позволяет воспроизводить 256 градаций яркости. В результате комбинации трех основных (аддитивных) цветов, каждый из которых воспроизводит 256 градаций интенсивности, удается получить палитру из 16,7 млн. цветов (256^3). Такое громадное количество цветовых тонов обеспечивает большой простор для экспериментов с редактируемым изображением. Когда цветовые каналы сливаются, в полученном составном изображении воспроизводится весь диапазон цветов исходного изображения.

Так как каналы представляют собой монохромные изображения, с ними можно работать точно так же, как и с любым полутоновым изображением. Например, при увеличении яркости красного канала в RGB-изображении с помощью фильтра Brightness-Contrast-Intensity (Яркость-Контрастность-Интенсивность) увеличивается уровень красного цвета в составном изображении.

Глава 7. Методы и алгоритмы построения сложных трехмерных объектов

Понятие “трехмерная” графика в настоящее время можно считать наиболее распространенным для обозначения темы, которая будет рассмотрена ниже (в литературе это название часто сокращается до "3D-графики"). Однако необходимо заметить, что такое название неточное, так как речь пойдет о создании изображения на плоскости, а не в объеме. Действительно трехмерные средства отображения пока что недостаточно широко распространены.

7.1 Модели описания поверхностей

Для описания формы поверхностей могут использоваться разнообразные методы. Сделаем обзор этих методов.

7.1.1. Аналитическая модель

Аналитической моделью будем называть описание поверхности математическими формулами. В КГ можно использовать много разновидностей такого описания. Например, в виде функции двух аргументов $z = f(x, y)$. Можно использовать уравнение $F(x, y, z) = 0$.

Наиболее часто используется параметрическая форма описания поверхности. Запишем формулы для трехмерной декартовой системы координат (x, y, z):

$$x = F_x(s, t),$$
$$y = F_y(s, t),$$
$$z = F_z(s, t),$$

где s и t — параметры, которые изменяются в определенном диапазоне, а функции F_x, F_y, и F_z определяют форму поверхности.

Преимущества параметрического описания — легко описывать поверхности, которые соответствуют неоднозначным функциям, замкнутые поверхности. Описание можно сделать таким образом, что формула не будет существенно изменяться при поворотах поверхности, масштабировании.

В качестве примера рассмотрим аналитическое описание поверхности шара. Сначала как функцию двух аргументов:

$$z = \pm \sqrt{R^2 - x^2 - y^2}$$

В виде уравнения: $x^2 + y^2 + z^2 - R^2 = 0$.

А также в параметрической форме:

$$x = R \sin s \cos t,\ y = R \sin s \sin t,\ z = R \cos s.$$

Для описания сложных поверхностей часто используют ***сплайны.*** Сплайн - это специальная функция, наиболее пригодная для аппроксимации отдельных фрагментов поверхности. Несколько сплайнов образуют модель сложной поверхности. Другими словами, сплайн — эта тоже поверхность, но такая, для которой можно достаточно просто вычислить координаты ее точек. Обычно используют кубические сплайны. Почему именно кубические? Так как третья степень является наименьшей, позволяющей описывать любую форму, и при стыковке сплайнов можно обеспечить непрерывную первую производную — такая поверхность будет без излома в местах стыка. Сплайны часто определяют параметрически. Запишем формулу для координаты $x(s,t)$ кубического сплайна в виде многочлена третьей степени параметров s и t:

$$\begin{aligned} x(s, t) = \quad & - a_{11} s^3 t^3 + a_{12} s^3 t^2 + a_{13} s^3 t + a_{14} s^3 + \\ & + a_{21} s^2 t^3 + a_{22} s^2 t^2 + a_{23} s^2 t + a_{24} s^2 + \\ & + a_{31} s t^3 + a_{32} s t^2 + a_{33} s t + a_{34} s + \\ & + a_{41} t^3 + a_{42} t^2 + a_{41} t + a_{44}. \end{aligned}$$

Для других координат можно записать подобные формулы — в виде функций *y(s, t), z(s,t).*

В математической литературе вы можете ознакомиться со способами определения коэффициентов a_{ij} для сплайнов, которые имеют заданные свойства.

Рассмотрим одну из разновидностей сплайнов — сплайн Безье. Приведем его сначала в обобщенной форме — степени $m x n$:

$$P(s,t) = \sum_{i=0}^{m} \sum_{j=0}^{n} C_m^i s^i (1-s)^{m-i} C_n^j t^j (1-s)^{n-j} P_{ij}$$

где P_{ij} — опорные точки-ориентиры, $0 \le s \le 1$, $0 \le t \le 1$, C_m^i и C_n^j – коэффициенты бинома Ньютона, они рассчитываются по формуле:

$$C_a^b = \frac{a!}{b!(a-b)!}$$

Кубический сплайн Безье соответствует m = 3, n = 3. Для его определения необходимы 16 точек-ориентиров P_{ij} (рис. 7.1); коэффициенты C_m^i и C_n^j равняются 1,3, 3, 1 при i , j = 0,1,2,3.

Характеризуя аналитическую модель в целом, можно сказать, что эта модель наиболее пригодна для многих операций анализа поверхностей. С позиций КГ можно указать такие положительные черты: простота (впрочем, не всегда) расчета координат каждой точки поверхности, нормали; небольшой объем информации для описания довольно сложных форм.

К недостаткам можно отнести: сложность формул описания с использованием функций, которые медленно вычисляются в компьютере, снижают скорость выполнения операций отображения; невозможность в большинстве случаев использования данной формы описания непосредственно для построения изображения поверхности. В таких случаях поверхность обычно отображают как многогранник, используя формулы аналитического описания для расчета координат вершин граней в процессе отображения, что уменьшает скорость сравнительно с полигональной моделью описания.

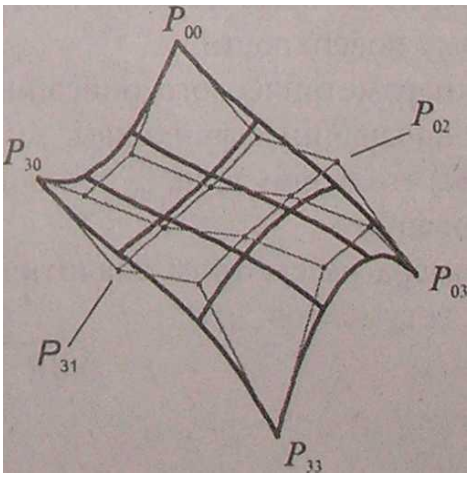


Рис. 7.1. Кубические сплайны Безье

7.1.2 Векторная полигональная модель

Для описания пространственных объектов здесь используются такие элементы: *вершины,* отрезки прямых *(векторы), полилинии, полигоны, полигональные поверхности* (рис. 7.2).

Элемент *"вершина" (vertex)* — главный элемент описания, все другие являются производными. При использовании трехмерной декартовой системы координат вершины определяются как *(x*,y,z). Каждый объект однозначно определяется координатами собственных вершин.

Вершина может моделировать отдельный точечный объект, размер которого не имеет значения, а также может использоваться в качестве конечной точки для линейных объектов и полигонов. Двумя вершинами задается ***вектор.***

Р

ис. 7.2. Элементы векторно-полигональной модели

Несколько векторов составляют ***полилинию.*** Полилиния может моделировать отдельный линейный объект, толщина которого не учитывается, а также может представлять собой контур полигона.

Полигон моделирует площадный объект. Один полигон может описывать плоскую грань объемного объекта. Несколько граней составляют объемный объект в виде полигональной поверхности — многогранник или незамкнутую поверхность (в литературе часто употребляется название "полигональная сетка").

Векторную полигональную модель можно считать наиболее распространенной в современных системах трехмерной КГ. Ее используют в системах автоматизированного проектирования, в компьютерных играх, в геоинформационных системах и т. п.

Рассмотрим структуры данных, которые используются в векторной полигональной модели. В качестве примера объекта будет куб. Рассмотрим, как можно организовать описание такого объекта в структурах данных.

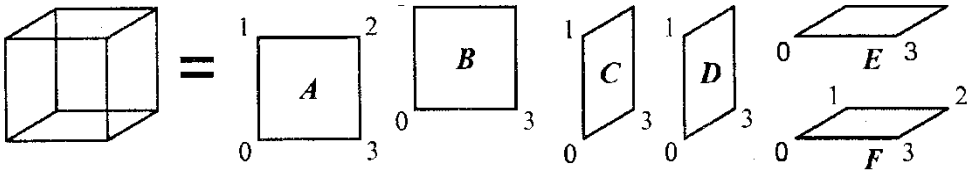


Рис. 7.3. Первый способ описания

Первый способ. Сохраняем все грани в отдельности

$$
\begin{aligned}
\text{Грань } A = \{\ & (x_{A0}, y_{A0}, z_{A0}), && (x_{A1}, y_{A1}, z_{A1}), && (x_{A2}, y_{A2}, z_{A2}), && (x_{A3}, y_{A3}, z_{A3})\ \}\\
\text{Грань } B = \{\ & (x_{B0}, y_{B0}, z_{B0}), && (x_{B1}, y_{B1}, z_{B1}), && (x_{B2}, y_{B2}, z_{B2}), && (x_{B3}, y_{B3}, z_{B3})\ \}\\
\text{Грань } C = \{\ & (x_{C0}, y_{C0}, z_{C0}), && (x_{C1}, y_{C1}, z_{C1}), && (x_{C2}, y_{C2}, z_{C2}), && (x_{C3}, y_{C3}, z_{C3})\ \}\\
\text{Грань } D = \{\ & (x_{D0}, y_{D0}, z_{D0}), && (x_{D1}, y_{D1}, z_{D1}), && (x_{D2}, y_{D2}, z_{D2}), && (x_{D3}, y_{D3}, z_{D3})\ \}\\
\text{Грань } E = \{\ & (x_{E0}, y_{E0}, z_{E0}), && (x_{E1}, y_{E1}, z_{E1}), && (x_{E2}, y_{E2}, z_{E2}), && (x_{E3}, y_{E3}, z_{E3})\ \}\\
\text{Грань } F = \{\ & (x_{F0}, y_{F0}, z_{F0}), && (x_{F1}, y_{F1}, z_{F1}), && (x_{F2}, y_{F2}, z_{F2}), && (x_{F3}, y_{F3}, z_{F3})\ \}
\end{aligned}
$$

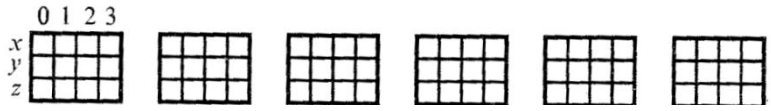


Рис. 7.4. Отдельные грани

В компьютерной программе такой способ описания объекта можно реализовать разнообразно. Например, для каждой грани открывать в памяти отдельный массив. Можно все грани записывать в один массив-вектор (это уже лучше). А можно использовать классы (язык С++) как для описания отдельных граней, так и объектов в целом. Можно создавать структуры, которые объединяют тройки (*x, y, z),* или сохранять координаты в отдельности. Принципиально это мало что меняет — так или иначе в памяти необходимо сохранять координаты вершин граней плюс некоторую информацию в качестве накладных затрат.

Рассчитаем объем памяти, необходимый для описания куба указанным способом:

$$П_1 = 6 \times 4 \times 3 \times Рв$$

где *Рв* — разрядность чисел, которые используются для представления координат вершин. Шесть граней здесь описываются 24 вершинами. Такое представление избыточно — каждая вершина записана трижды. Не учитывается то, что у граней есть общие вершины.

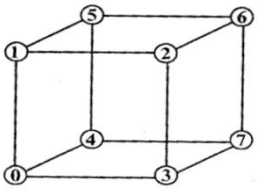


Рис. 7.5. Индексы вершин

Второй способ описания. Для этого варианта координаты восьми вершин сохраняются без повторов. Вершины пронумерованы (рис. 7.5), а каждая грань представлена в виде списка индексов вершин (рис. 7.6).

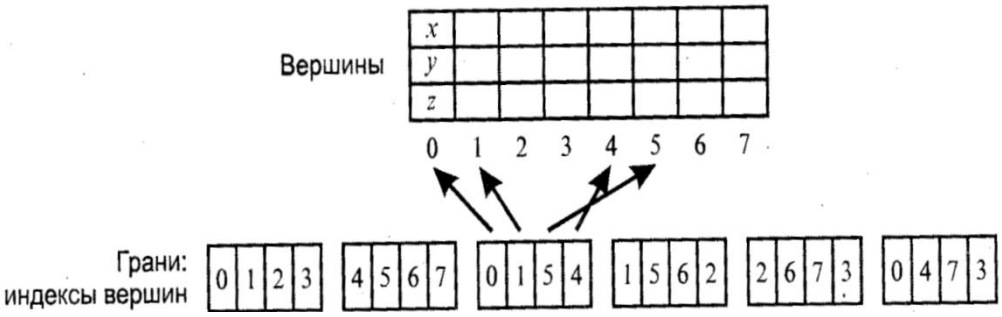


Рис. 7.6. Второй способ описаня

Оценим затраты памяти:

$$П2 = 8 \times 3 \times Рв + 6 \times 4 \times Риндекс$$

где *Рв* — разрядность координат, *Риндекс* — разрядность индексов.

Третий способ. Пронумеруем также и ребра (рис. 7.7). Будем сохранять грани в виде списка индексов ребер. Каждое ребро будет, в свою очередь, представлено списком индексов вершин. Этот способ в литературе иногда называют *"линейно-узловой "* моделью.

Оценим затраты памяти:

$$П_3 = 8 \times 3 \times Рв + 12 \times 2 \times Риндекс\ верш. + 6 \times 4 \times Риндекс\ ребер$$

где *Рв* - разрядность координат, *Риндекс верш.* и *Риндекс ребер* - разрядность индексов вершин и ребер

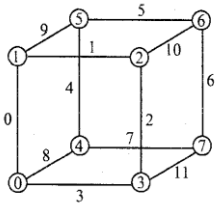


Рис. 7.7. Индексы вершин и ребер

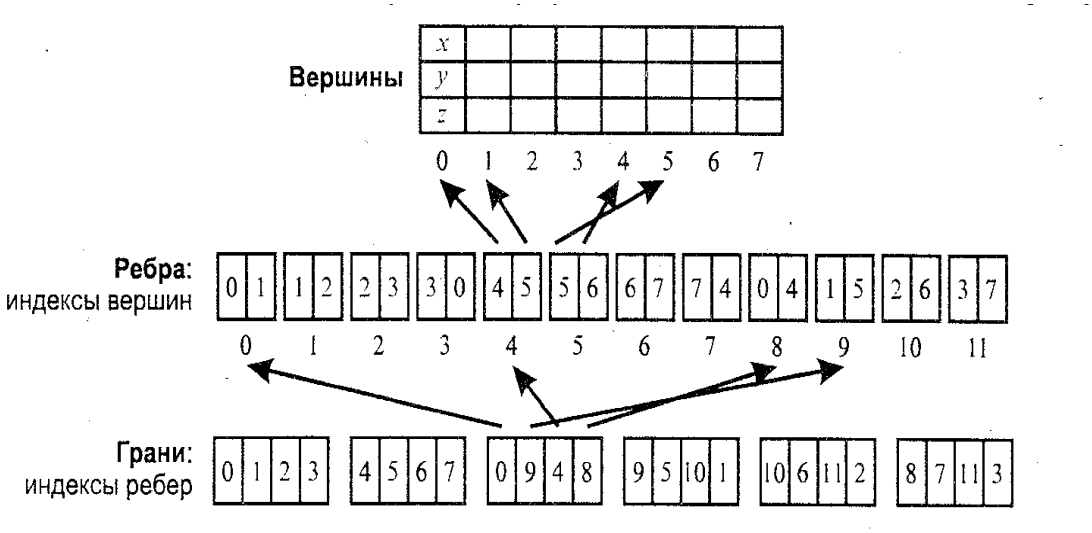


Рис. 7.8. Линейно-узловая модель

7.1.3 Воксельная модель

Воксельная модель – это трехмерный растр. Воксел это элемент объема. По аналогии с 2D растрами, состоящими из пикселов. Вокселы заполняют объем в трехмерном растре (рис.7.9)

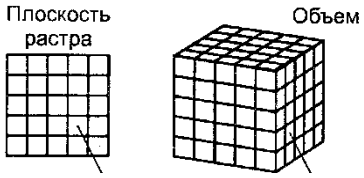


Рис. 7.9. Пикселы и вокселы

Как мы знаем, любой пиксел должен иметь свой цвет. Любой воксел также имеет свой цвет, и, кроме того, прозрачность. Полная прозрачность воксела означает пустоту соответствующей точки объема.

Поскольку вокселы располагаются в узлах равномерной сетки, то, обычно, чем меньше шаг сетки, тем большее количество вокселов помещается в определенном объеме, и тем меньший размер каждого отдельного воксела. Основная характеристика вексельной модели — разрешающая способность — количество вокселов в определенном объеме. Она и определяет точность моделирования трехмерных объектов (рис. 7.10).

Для современной КГ вексельный метод считается одним из перспективных. Его используют в компьютерных системах, предназначенных для медицинских целей.

Например, при сканировании томографом *(computed tomography)* получаются изображения срезов объекта, которые потом объединяются в виде объемной модели для дальнейшего анализа. Вексельный метод используется в геологии, сейсмологии, в компьютерных играх. Вокселы также используются для графических устройств отображения, которые создают действительно объемные изображения.

Положительные черты воксельной модели:

- достаточно простое описание сложных объектов и сцен;

- простая процедура отображения объемных сцен;
- простое выполнение топологических операций над отдельными объектами и сценой в целом. Например, просто выполняется показ разреза — для этого соответствующие вокселы можно сделать прозрачными.

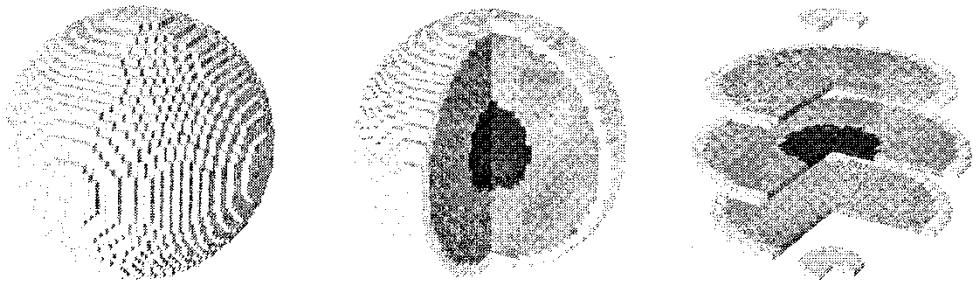


Рис. 7.10. Воксельная модель шара и ее разрезы.

Недостатки воксельной модели:

- большое количество информации, необходимой для представления объемных данных. Например, объем 256x256x256 имеет небольшую разрешающую способность, но требует свыше 16 миллионов вокселов;
- значительные затраты памяти ограничивают разрешающую способность, точность моделирования;
- большое количество вокселов обуславливает маленькую скорость создания изображений объемных сцен;
- как и для любого растра, возникают проблемы при масштабировании — увеличении или уменьшении.

7.1.4 Равномерная сетка

Эта модель описывает координаты отдельных точек поверхности следующим способом (рис. 7.11). Каждому узлу сетки с индексами (i,j) приписывается значение высоты $z_{i,j}$. Индексам (i,j) соответствуют определенные значения координат (x,y). Расстояние между узлами одинаковое — dx по оси x и dy по оси y.

Фактически, такая модель является двумерным массивом (растром, матрицей), каждый элемент которой хранит значение высоты.

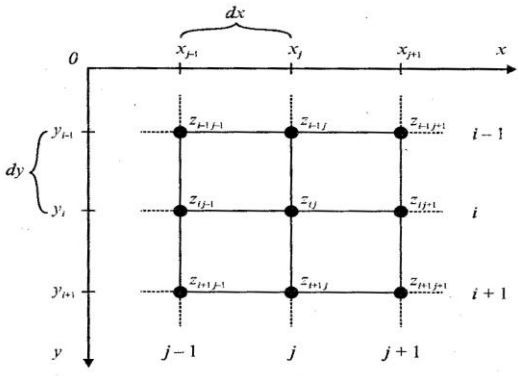


Рис. 7.11. Узлы равномерной сетки

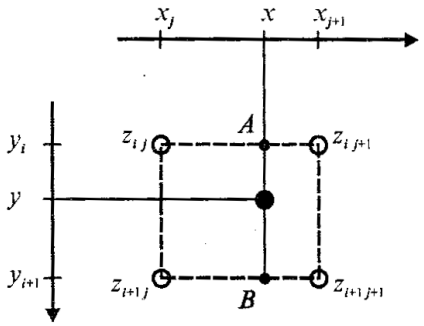


Рис. 7.12. Точка в сетке с координатами *(x, y, z)*

Не каждая поверхность может быть представлена этой моделью. Если в каждом узле записывается только одно значение высоты, то это означает, что поверхность описывается однозначной функцией *z=f(xy).* Иначе говоря, это такая поверхность, которую любая вертикаль пересекает только один раз. Не могут моделироваться также вертикальные грани. Следует отметить, что для сетки не обязательно использовать только декартовы координаты. Например, для того, чтобы описать поверхность шара однозначной функцией, можно использовать полярные координаты. Равномерная сетка часто используется для описания рельефа земной поверхности.

Рассмотрим, как можно рассчитать значение высоты для любой точки внутри границ сетки. Пусть ее координаты равняются *(x, y).* Надо найти соответствующее значение *z.* Решение этой задачи — интерполяция значений координат 2 ближайших узлов (рис 7.12).

Сначала необходимо вычислить индексы *i* и *j* одного из узлов:

$$i = \left]\frac{y-y_o}{dy}\right[, \qquad j = \left]\frac{x-x_o}{dx}\right[$$

где *]a[* — целая часть числа *a,* то есть наибольшее целое, которое не превышает *a.*

Далее используем, например, билинейную интерполяцию. Для этого сначала найдем значения *z* в точках *A и B.* Из пропорции

$$\frac{z_A - z_{ij}}{z_{ij+1} - z_{ij}} = \frac{x - x_j}{x_{j+1} - x_j},$$

учитывая, что $x_{j+1} - x_j = dx,$ будем иметь:

$$z_A = z_{ij} + (x - x_j)\frac{z_{ij+1} - z_{ij}}{dx}.$$

Аналогично найдем $z_B = z_{i+1,j} + (x - x_j)\frac{z_{i+1,\,j+1} - z_{i+1,\,j}}{dx}$

Теперь нужно вычислить такое значение z, поделив отрезок AB пропорционально значению y:

$$\frac{z - z_A}{z_B - z_A} = \frac{y - y_i}{dy}.$$

И будем иметь: $z = z_A + (y - y_i)\frac{z_B - z_A}{dy}$

Положительные черты равномерной сетки:

1. Простота описания поверхностей

2. Возможность быстро узнать высоту любой точи поверхности простой интерполяцией.

Недостатки равномерной сетки:

1. Поверхности, которые соответствуют неоднозначной функции высоты в узлах сетки, моделироваться не могут

2. Для описания сложных поверхностей нужно большое количество узлов, которое может быть ограничено объемом памяти компьютера

3. Описание отдельных типов поверхностей может быть более сложным, чем в других моделях. Например, многогранная поверхность требует избыточный объем данных для описания в сравнении с полигональной моделью.

7.1.5 Неравномерная сетка. Изолинии

Неравномерной сеткой назовем модель описания поверхности в виде множества отдельных точек $\{(x_0, y_0, z_0), (x_1, y_1, z_1), ..., (x_{n-1}, y_{n-1}, z_{n-1})\}$ принадлежащих поверхности. Эти точки могут} быть получены, например, в результате измерений поверхности какого-либо объекта с помощью определенного оборудования.

Такую модель можно считать обобщением для некоторых рассмотренных нами моделей. Например, векторная полигональная модель и равномерная сетка могут считаться разновидностями неравномерной сетки. Эти разновидности мы рассмотрели в отдельности, так как они играют важную роль при решении задач КГ. А вообще, может существовать не один вариант классификации способов описания поверхностей. Следует учитывать определенную условность нашего перечня моделей поверхностей, последовательность перечисления таких моделей может быть и другой.

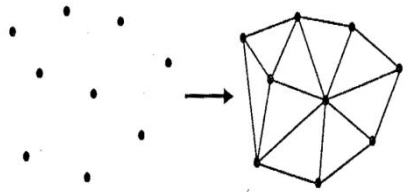


Рис. 7.13. Триангуляция неравномерной сетки

Рассмотрим модель описания поверхности в виде ***множества точечных значении,*** логически никак не связанных между собою. Неравномерность задания опорных точек усложняет определение координат для других точек поверхности, которые не совпадают с опорными точками. Нужны специальные методы пространственной интерполяции. Так, например, можно поставить такую задачу - по известным координатам (x, *y)* вычислить значение координаты z. Для этого необходимо найти несколько ближайших точек, а потом рассчитать искомое значение, исходя из взаимного расположения этих точек в проекции (x,y). Как мы уже рассмотрели выше, для равномерной сетки это намного проще— поиска фактически нет, мы сразу вычисляем индексы ближайших опорных точек. Еще одна задача — отобразить поверхность. Эту задачу можно решать несколькими способами, в том числе *триангуляцией.* Процесс триангуляции можно представить себе так (рис. 7.13).

Сначала находим первые три самые близкие друг другу точки — и получаем одну плоскую треугольную грань. Потом находим точку, ближайшую к этой грани, и образуем смежную грань. И так далее, пока не останется ни одной отдельной точки. Это общая схема, в литературе описано много разных способов триангуляции. Довольно часты ссылки на триангуляцию *Делон*.

Пусть нужно связать треугольниками четыре точки. Возможны два варианта триангуляции (рис. 7.14).

В соответствии с критерием Делоне ни одна окружность, проходящая через три точки любого треугольника, не должна охватывать точки, которые принадлежат другим треугольникам. Здесь необходимо указать, что не всегда триангуляция Делоне дает нужный результат. Может так случиться, что необходимо соединять точки треугольниками вопреки указанному выше критерию. Например, кроме точек триангуляции нужно учитывать особенности формы некоторых структурных элементов — горных хребтов, ям с отвесными стенами и т.п. Тогда нужно соответственно управлять триангуляцией.

Представление поверхности треугольными гранями в настоящее время очень часто используется в разнообразных областях — от компьютерных игр и фильмов до систем автоматизированного проектирования и ГИС. Треугольник сейчас — базовый элемент для современных видеоадаптеров. В англоязычной литературе для триангуляционной модели встречается и такое название: *TIN (Triangulated Irregular Network)*

Следует заметить, что, по всей видимости, преобразование неравномерной сетки в триангуляционную модель в ходе визуализации нецелесообразно. Для обеспечения высокой скорости графики, например в режиме анимации, следует заранее один раз преобразовать неравномерную сетку во множество треугольников (а это по нашей классификации уже векторная полигональная модель), а затем в ходе создания кадров многократно использовать готовый результат триангуляции.

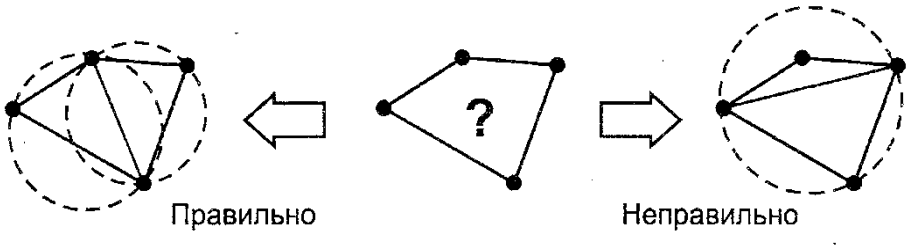


Рис. 7.14. Триангуляция в соответствии с критерием Делоне

Рассмотрим еще один из вариантов описания поверхности — *изолинии высоты.* Любая изолиния состоит из точек, которые представляют одно числовое значение какого-либо показателя, в данном случае — значение высоты (рис. 7.15). Изолинии высоты также можно представить себе как контуры разреза поверхности горизонтальными плоскостями (поэтому для изолиний высоты часто используется название "горизонтали").

Описание поверхности изолиниями высоты часто используется, например, в картографии. По бумажной карте можно с определенной точностью рассчитать высоту в точках местности, углы наклона и прочие параметры рельефа. Необходимо заметить, что описание рельефа земной поверхности изолиниями высоты неправильно представлять как разрезы горизонтальными плоскостями, так как поверхность Земли не плоская. Если бы Земля была шаром, то изолинии высоты можно было бы трактовать как изолинии радиусов. Однако Земля — не шар, она имеет намного более сложную форму, названную *геоидом.* В геодезии и картографии геоид аппроксимируют с определенной точностью разнообразными эллипсоидами. Таким образом, здесь можно говорить об изолиниях некоторых условных высот в специальных системах координат.

Конечно, для описания поверхности можно использовать не только изолинии высоты, а также другие изолинии, например *x*- или *y-изолинии.*

В компьютерных системах изолинии часто описываются векторно — как полилинии. Используются также изолинии в виде сплайновых кривых.

Точки, которые составляют изолинии, и отдельные опорные точки располагаются неравномерно.

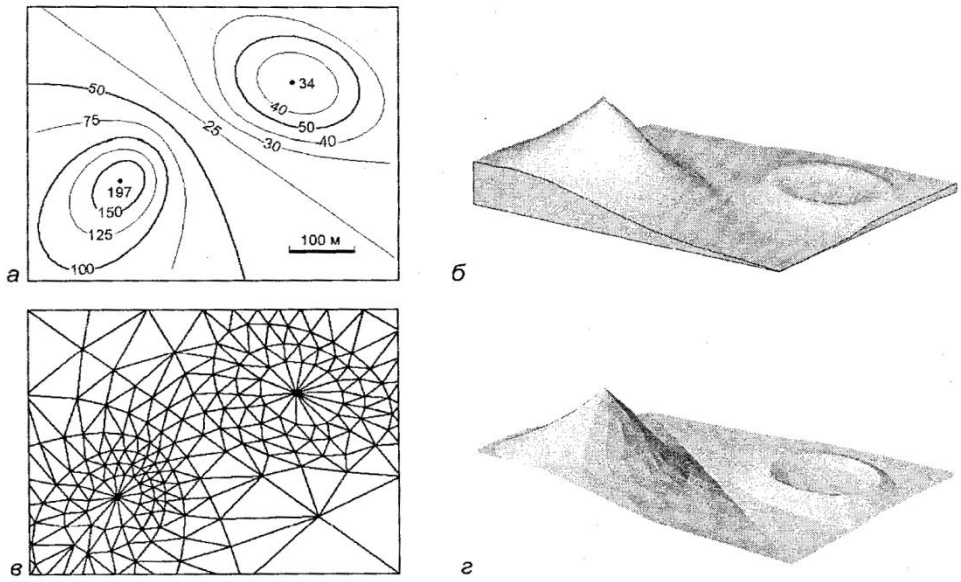


Рис. 7.15 Пример поверхности: а — изолинии высоты; *б* — вид поверхности; в — фрагмент триангуляционной сетки; *г* — поверхность из треугольников (здесь нарочно подчеркнуты отдельные треугольные грани)

Это усложняет расчет координат точек поверхности. В графических компьютерных системах для выполнения многих операций, и в первую очередь — для показа поверхности, обычно необходимо преобразовать описание поверхности в другую форму. Преобразование изолиний в полигональную модель также выполняется методами триангуляции (здесь алгоритмы триангуляции сложнее, чем для триангуляции массива отдельных точек). Для преобразования неравномерной сетки в равномерную используют специальную интерполяцию.

Положительные черты неравномерной сетки: наглядность показа рельефа поверхности изолиниями на картах, планах; использование отдельных опорных точек, наиболее важных для заданной формы поверхности, обуславливает меньший объем информации по сравнению с другими моделями, например, с равномерной сеткой.

Недостатки неравномерной сетки: невозможность или сложность выполнения многих операций над поверхностями; сложные алгоритмы преобразования в другие формы описания поверхностей.

7.2. Визуализация трехмерных объектов

Любой трехмерный объект может быть изображен по-разному и различными способами. В одном случае нужно показать форму объекта, во втором – внутреннюю структуру объекта, в третьем имитировать реальную действительность, в четвертом – возбудить воображение зрителя чем-то неизвестным. Условно разделим способы визуализации по характеру изображений и по степени сложности соответствующих алгоритмов на такие уровни:

1. Каркасная визуализация
2. Показ поверхностей в виде многогранников с плоскими гранями или сплайнов с удалением невидимых точек
3. То же что и для второго уровня, плюс сложное закрашивание объектов для имитации отражения света, затенения, прозрачности, использование текстур.

Простейшая, каркасная, визуализация часто используется в процессе редактирования объемных объектов. Визуализация второго уровня используется для упрощенного показа объемных объектов. Например, для графиков функций $z=f(x,y)$ (в виде рельефа поверхности) часто достаточно показать все грани сетки одним цветом, зато

нужно обязательно удалить невидимые точки. Это более сложная процедура в сравнении с выводом каркасного изображения.

Сложность процесса графического вывода возрастает по мере приближения к некоторому идеалу — созданию полной иллюзии естественных, живых, реалистических изображений. Усилия многих ученых и инженеров всего мира направлены на разработку методов и средств достижения этой цели. Здесь полнее всего ощущается связь компьютерной графики с естественными науками, с дисциплинами, посвященными изучению окружающего мира. Например, для создания реалистических изображений нужно принимать во внимание законы оптики, которые описывают свет и тень, отражение и преломление. Компьютерная графика находится на стыке многих дисциплин и разделов науки.

7.2.1 Каркасная визуализация

Каркас обычно состоит из отрезков прямых линий — ребер многогранника, хотя можно строить каркас и на основе кривых, в частности сплайновых кривых Безье. Все ребра, которые показаны в окне вывода, видно — как ближние, так и дальние.

Для построения каркасного изображения надо знать координаты всех вершин в мировой системе координат. Потом превратить координаты каждой вершины в экранные координаты в соответствии с выбранной проекцией. Потом выполнить цикл вывода в плоскости экрана всех ребер как отрезков прямых (или кривых), соединяющих вершины.

7.2.2 Показ с удалением невидимых точек

Здесь мы будем рассматривать поверхности в виде многогранников или полигональных сеток. Известны такие методы показа с удалением невидимых точек: сортировка граней по глубине, метод плавающего горизонта, метод Z-буфера и т.п.

Сортировка граней по глубине. Это означает рисование полигонов граней в порядке от самых дальних к ближним. Этот метод не является универсальным, так как иногда нельзя четко различить, какая грань ближе (рис. 7.16). Известны модификации этого метода, которые позволяют корректно рисовать подобные грани. Метод сортировки по глубине эффективен для показа поверхностей, заданных функциями $z=f(x,y)$.

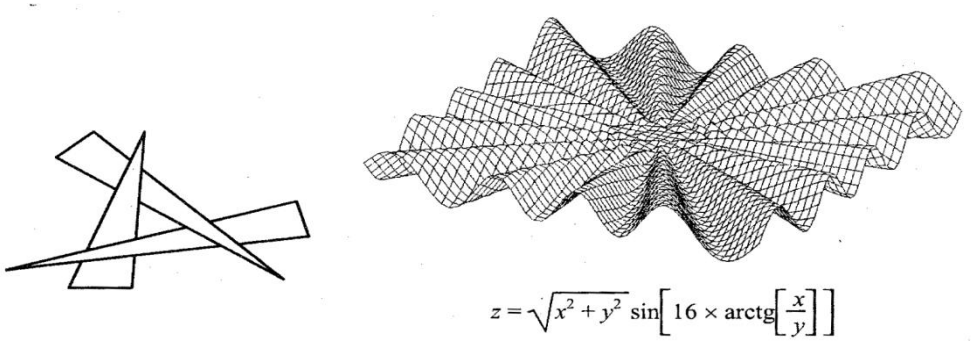


Рис. 7.16. В каком порядке рисовать грани? Поверхность нарисована четырехуголными гранями.

В общем виде алгоритм сортировки по глубине выглядит следующим образом.

Грани после разбиения сортируются по минимальному расстоянию до экранной плоскости, и выводятся в порядке их приближения (z-буфер).

Если же объекты пересекаются, то алгоритм несколько усложняется и проверяется еще ряд тестов перед выводом на экран некоторой грани P. Надо убедиться, что никакая другая грань Q не закроет P.

1. Пересекаются ли проекции граней P и любой другой Q
   -на ось Ox?
   -на ось Oy?

2. Пересекаются ли проекции этих граней на экранной плоскости?
3. Находится ли грань P по другую сторону плоскости, проходящей через Q, чем начало координат (наблюдатель)?
4. Обратное (2), т.е. находится ли Q по ту же сторону от плоскости, проходящей через P, что и начало координат (наблюдатель)?

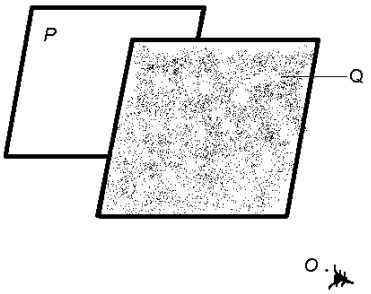


Рис. 7.17. Алгоритм сортировки по глубине.

Если хотя бы один из этих тестов отрицательный, то, значит, грани упорядочены верно, и P сравнивают со следующей гранью в списке, а в противном случае грани меняют местами в списке. Для чего проверяют тесты 3. 4.

Метод плавающего горизонта. Здесь, в отличие от предыдущего метода, грани выводятся в последовательности от ближних к самым дальним. На каждом шаге границы граней образуют две ломаные линии — верхний горизонт и нижний горизонт. В течение вывода каждой новой грани рисуется только то, что выше верхнего горизонта, и то, что ниже нижнего горизонта. Соответственно, каждая новая грань поднимает верхний и опускает нижний горизонты. Этот метод часто используют для показа поверхностей, которые описываются функциями *z=f(x,y).*

В общем виде метод выглядет следующим образом.

Пусть надо построить график функции двух переменных *Z=f(x, y)*, в виде сетки координатных осей.

При параллельном проектировании проекций вертикальной линии является вертикальная линия.

Любая точка *P(x, y, z)* переходит в точку [*(p,e1), (p, e2)*] на плоскости экрана, где

e_1= *(cosφ, sinφ, 0)*

e_2= *(sinφ, sinq, - cosφ sinq, cosq)*, а направление проектирования:

e_3= (sinφ cosq, - cosφ - cosq, sinq), где φϵ[0, 2π], qϵ[-π/2, π/2], - углы подобраны так, чтобы плоскость $y=y_1$ была ближе к экранной плоскости, чем $y=y_2$, т.е. $y_1<y_2$.

Из этого следует, что линия $Z=f(x, y_j)$, не закрывает линию $Z=f(x, y_i)$.

Тогда возможен алгоритм, когда линии рисуются в порядке удаления (возрастания y) и при рисовании очередной линии рисуется только та ее часть, которая не закрывается ранее нарисованной.

Для определения частей линий $Z=f(x, y_к)$, которые не закрываются ранее нарисованными, вводятся линии горизонта.

Пусть проекцией линии $Z=f(x, y_к)$ на плоскость экрана является линия $y=y_к(x)$, где *(x, y)* – это координаты на экране. Тогда линии горизонта определяются как:

$$\begin{cases} y^k{}_{\max}(x) = \max \cdot y_i(x), \text{ где } 1 \le i \le k-1; \\ y^k{}_{\min}(x) = \min \cdot y_i(x), \text{где } 1 \le i \le k-1 \end{cases}$$

На экране рисуются только те части линии $y=y^к(x)$, которые выше $y^к{}_{max}(x)$ или ниже $y^к{}_{min}(x)$.

Проще всего этот метод реализовать с помощью модифицированного растрового алгоритма Брезенхейма, который перед выводом очередного пикселя сравнивает его ординату с верхней и нижней линией (массивы ординат).

Аналогичным методом можно воспользоваться при построении объемных предметов. Только в этом случае изображение выводится по мере удаления от экранной плоскости, а по мере – приближения, т.е. начиная с дальних граней, и, заканчивая ближними, которые будут закрывать собой невидимые части более дальних граней.

Для определения порядка вывода граней считают, что грань, лежащая в полосе $(y, x) y_i \le y \le y_{i+1}$ не может закрыть грань из полосы $(y, x) y_{i-1} \le y \le y_i$

Метод Z-буфера. Здесь используется специальный дополнительный массив (буфер), в который записывается координата *Z* для каждого пиксела растра изображения. Координата Z означает расстояние соответствующей точки объекта до плоскости проецирования — это может быть, например, видовая координата *Z* (ось *Z* располагается перпендикулярно плоскости проецирования).

Рассмотрим алгоритм рисования объектов в соответствии с этим методом. Пусть, чем ближе точка в пространстве к плоскости проецирования, тем больше значение Z. Тогда сначала Z-буфер заполняется минимальными значениями. Потом начинается вывод всех объемов. Причем, не имеет значения порядок вывода объектов. Для каждого объекта выводятся все го пикселы в любом порядке. Во время вывода каждого пиксела по его координатам *(X,Y)* находится текущее значение Z в Z-буфе$_{Р}$е. Если рисуемый пиксел имеет большее значение Z, чем значение в Z-буфере, то этот пиксел действительно рисуется, а его Z-координата записывается в Z-буфер. Таким образом, после рисования всех пикселов всех объектов растровое изображение будет состоять из пикселов, которые соответствуют точкам объектов с наибольшими значениями координат Z, то есть видимые точки являются ближайшими к нам.

Метод Z-буфера сейчас очень популярен благодаря простоте и эффективности. Современные 3D-акселераторы аппаратно поддерживают этот метод как на уровне операции, так и памяти. Видеоадаптер имеет собственную память для Z-буфера, доступ к которой осуществляется быстрее, чем к оперативной памяти компьютера. В конвейере графического процессора и манипуляции со значениями Z-буфера легко совместить с другими пиксельными операциями для вывода полигонов. Существует разновидность этого метода - для ускорения вычислений используется не Z, а обратное значение *(W-буфер)*.

Укажем некоторые проблемы использования метода Z-буфера.

1. Необходимость выделения дополнительной памяти. Для Z-буфера необходима память объем которой соответствует размерам растра изображения и точности чтения координаты Z. Используют обычно 32, 24 или 16 битов на один пиксел Z-буфера. Например для Z-буфера 1024x768x32 нужно 3 Мбайта. Сейчас такие затраты памяти не считаются существенными. Если объем памяти _ критичен, то кадровый и Z-буфер разделяют на фрагменты (тайлы) и выполняют визуализацию для любого фрагмента отдельно. Файловая организация Z-буфера может использоваться также и для повышения скорости визуализации.

2. Полная инициализация Z-буфера (запись "самых дальних" значений) перед началом вывода того кадра уменьшает скорость анимации. Тем не менее, часто используется такой прием - инициализировать Z-буфер один раз для пары кадров. Это возможно, если разделить полный диапазон значений Z пополам. Например, если полный диапазон значений от 0 до 1, то в первом кадре работать со значениями Z, смещенными в интервал от 1 до 0.5 (дальняя половина), а в следующем кадре — от 0.5 до 0 (ближняя половина). Однако за повышение скорости здесь приходится платить точностью вычисления глубины -она уменьшается вдвое.

3. Большое количество лишних операций. Поскольку видимость устанавливается на уровне пикселов, то в цикле выполняются лишние операции для тех полигонов, которые полностью невидимы. Такие полигоны желательно отсекать до цикла вывода. Для ускорения вывода насыщенных сцен, содержащих миллионы полигонов, известны модифи-

кации метода, например, иерархический Z-буфер, в котором используется пирамида 2-х буферов разной разрешающей способности.

4. Проблемы с выводом полупрозрачных объектов.

5. Использовать в качестве расстояния координату 2 нельзя при углах обзора 180 градусов и больше. Для цилиндрических и сферических проекций лучше использовать радиальное расстояние от текущей точки объекта до точки схода лучей проецирования.

Несмотря на кажущуюся простоту, эта задача является достаточно сложной. Существует 2 основных подхода к решению задачи:

первый – заключается в определении точек объекта (пикселей), которые вдоль направления проектирования ближе всего расположены к картинной плоскости;

второй – заключается в непосредственном сравнении объектов друг с другом для выяснения видимых частей.

Существует много смешанных методов, которые объединяют оба эти подхода.

Отсечение нелицевых граней

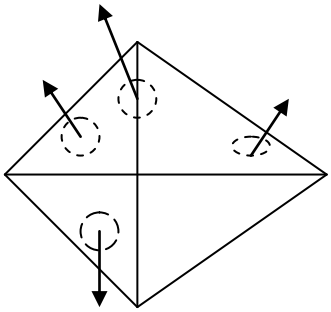


Рис. 7.18. Лицевые и нелицевые грани

Пусть для каждой грани объекта задан единичный вектор внешней нормали $\vec{n}$, а вектор $\vec{l}$ - задает направление проектирования. Если нормаль грани $\vec{n}$ с вектором $\vec{l}$ составляет тупой угол, то эта грань заведомо не может быть видна.

При параллельном проектировании: это можно записать как скалярное произведение $(\vec{n},\vec{l}) \le 0$.

При центральном проектировании : с центром в точке E, для любой точки P вектор проектирования $\vec{l} = \vec{E} - \vec{P}$, а для определения, является ли грань лицевой или нет? Достаточно взять любую точку P на ней и проверить условие $(\vec{n},\vec{l}) \le 0$. *Знак этого скалярного произведения* не зависит от выбора точки грани, а определяется тем, в каком полупространстве относительно плоскости, содержащей данную грань, *лежит центр проектирования.*

Если строится только один *выпуклый* многогранник, то задача может быть решена этим способом.

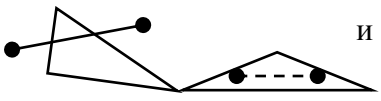
Если строится комбинация объектов, то используя этот подход можно хотя бы сократить вдвое количество рассматриваемых граней.

Алгоритм:

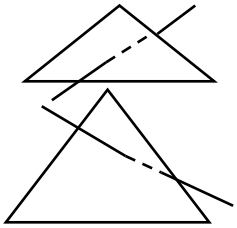
1 – сначала отбрасываются все ребра, обе грани которых не являются лицевыми, т.е. они заведомо невидны.

2 – проверяются все оставшиеся ребра со всеми гранями многоугольника на закрывание:

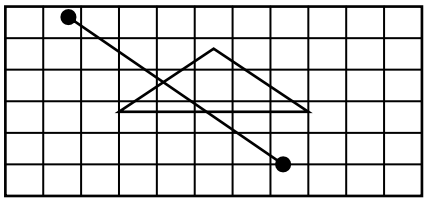
- грань не закрывает ребро и оно выводится.
- грань полностью закрывает ребро и оно удаляется из списка рассматриваемых.



- грань частично закрывает ребро, тогда ребро разбивается на части, из которых видимыми могут быть *не более двух частей*. Само ребро удаляется из списка, но в список проверенных ребер добавляются те его части, которые не закрываются гранью.



Алгоритм можно значительно сократить (имеем в виду количество проверок) если экранную плоскость разбить на клетки, и для каждой клетки составить список граней, проекции которых имеют *непустое пересечение* с этой клеткой.



Тогда для произвольного ребра сначала находятся все клетки, в которые попадает проекция этого ребра, и, следовательно, рассматриваются не все грани, а только те, которые содержатся в списке данных клеток.

При этом подходе требуется время для разбиения и составления списка, но алгоритм работает эффективней.

Алгоритм Аппеля.

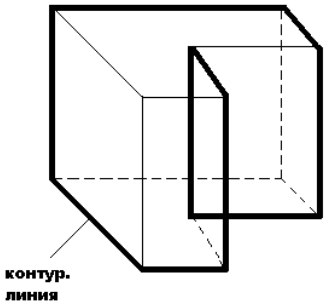


Рис. 7.19. Контурная линия

Здесь вводится понятие ***количественной невидимости***, как количество граней, закрывающих вершину (точку).

Если количественная невидимость равна нулю, то точка видима.

Количественная невидимость точек ребра при прохождении так называемой контурной линии может изменяться.

Берется какая-нибудь вершина и прослеживается изменение количественной невидимости вдоль каждого из ребер, выходящих из этой вершины. Эти ребра проверяются на прохождение позади контурной линии. Где количественная невидимость равна нулю, ребро сразу рисуется. Если не равно нулю, то проверяется количественная невидимость для всех ребер, выходящих из новой вершины, и. т. д.

Этот алгоритм более эффективен, чем алгоритм Робертса, так как количество ребер, входящих в контурную линию, намного меньше общего числа ребер.

Методы двоичного разбиения пространства.

Пусть некоторая плоскость в объектном пространстве разбивает множество всех граней на два не пересекающихся множества (кластера) по одну и по другую сторону от этой плоскости.

При этом очевидно, что ни одна из этих граней, лежащих в пространстве, не содержащем наблюдателя, не может закрыть собой грань, находящуюся в другом полупространстве (где наблюдатель).

Сначала выводятся грани из дальнего кластера, затем из ближнего.

Разбиение продолжается до тех пор, пока в каждом кластере не останется по одной грани.

Обычно в качестве разбивающей плоскости рассматривают плоскость, проходящую через одну из граней. При этом реально множество граней разбивается не на 2 части, а на 4:

- грань лежит на плоскости;
- пересекает плоскость;
- в положительном пространстве;
- в отрицательной области;

Каждый узел дерева разбиения граней можно представить структурой.

Процесс построения дерева заключается в выборе грани, проведении через нее плоскости и разбиение всех граней. В процессе выбора граней следует придерживаться 2 критериев:

- получить более сбалансированное дерево;
- минимизировать количество разбиений.

Эти критерии взаимоисключающие - и надо выбрать компромисс.

Преимущество этого метода - полная независимость от положения центра проектирования, что необходимо при построении изображений одной сцены, но с разных точек наблюдения.

Метод построчного сканирования.

Это еще один метод, который успешно используется для создания компьютерных игр (типа прохода по лабиринту, когда вся сцена представляет собой прямоугольный лабиринт с постоянной высотой пола и потолка и набором вертикальных стен).

Рассматривается сечение сцены горизонтальной (вертикальной) плоскостью, проходящей через центр проектирования.

Каждая линия - однозначно определяет одну вертикальную плоскость. Среди всех пересечений видимым будет только одно – ближайшее – плоскостью, проходящей через центр проектирования.

Каждая линия – однозначно определяет одну вертикальную плоскость. Среди всех пересечений видимым будет только одно – ближайшее.

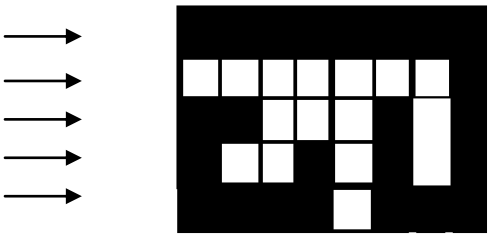


Рис. 7.20. Построчное сканирование

Алгоритм Варнака (Вариока).

Вся видимая часть картинной плоскости разбивается на 4 равные части и проверяется:

- эта часть полностью накрывается проекцией ближайшей грани;
- часть не покрывается проекцией ни одной грани.

Когда ни одно из условий не выполнено, часть разбивается еще на 4 части и т. д., пока размер части больше, чем размер пикселя.

Когда часть равна одному пикселю, явно находится ближайшая к ней грань и закрашивается.

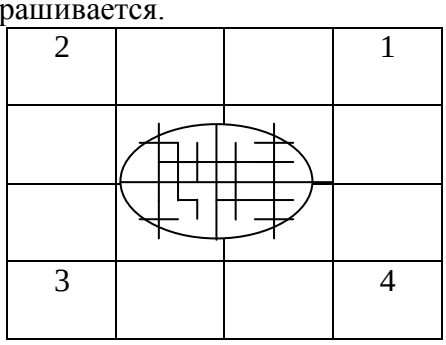


Рис. 7.21. Части алгоритма Варнака

Отсечение отрезка. Алгоритм Сазерленда-Кохена.

Рассмотрим теперь случай, когда необходимо отсечь линии, выходящие за границы окна вывода.

Простой и эффективный алгоритм отсечения отрезков по границе произвольного прямоугольника называется алгоритмом Сазерленда - Кохена. Он заключается в разбиении всей плоскости на 9 областей. Определив, в какие области попали концы отрезка, легко понять, где именно необходимо отсечение. Для точки P (x ,y) соответствует 4 - битовый код , причем каждый бит соответствует определенному положению .

КОД: B3 B2 B1 B0

B3 = (x <x min) B1 = (y < y min)
B2 = (x > x max) B0 = (y > y max)

Идея алгоритма заключается в том, что отрезок анализируется на предмет пересечения поочередно со всеми сторонами окна. Если пересечение существует, то отбрасывается часть отрезка между концом P1 (вн. окна) и найденной точкой пересечения Pn (Xn , Yn). Причем в алгоритме отсечения рассматриваются только те отрезки, видимость которых неочевидна.

F -переменная, определяющая вид отрезка, причем:
0, отрезок горизонтальный
-1, отрезок вертикальный
1, иначе

Глава 8. Реалистическое представление сцен

Основные направления реалистического представления сцен трехмерной графики определяются как:

- синтез реалистичных изображений,
- реалистическое оживление синтезированных объектов (анимация).

В этом разделе будут рассмотрены только некоторые базовые методы синтеза реалистических изображений:

- Модели освещения
- Модели закраски
- Трассировка лучей
- Имитация микрорельефа
- Механизмы отражения света

Другие методы синтеза – прозрачность, тени, задание фактуры, излучательность и т.д. выносятся на самостоятельное изучение.

8.1 Закрашивание поверхностей

В этом разделе мы рассмотрим методы, которые позволяют получить более-менее реалистичное изображение для объектов, которые моделируются многогранниками и полигональными сетками.

8.1.1 Модели отражения света

Рассмотрим, как можно определить цвет пикселов изображения поверхности в соответствии с интенсивностью отраженного света при учете взаимного расположения поверхности, источника света и наблюдателя.

Зеркальное отражение света. Угол между нормалью и падающим лучом равняется углу между нормалью и отраженным лучом. Падающий луч, отраженный луч и нормаль располагаются в одной плоскости (рис. 8.1).

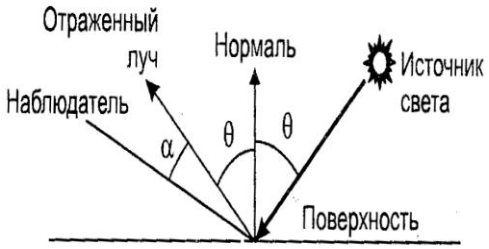


Рис. 8.1. Зеркальное отражение света

Поверхность считается *идеально зеркальной,* если на ней отсутствуют какие-либо неровности, шероховатости. Собственный цвет у такой поверхности не наблюдается. Световая энергия падающего луча отражается только по линии отраженного луча. Любое рассеивание в стороны от этой линии отсутствует. В природе, вероятно, нет идеально гладких поверхностей, поэтому полагают следующее: если глубина шероховатостей существенно меньше длины волны излучения, то рассеивание не наблюдается. Для видимого спектра можно принять, что глубина шероховатости поверхности зеркала должна быть меньше 0.5 мкм.

Если поверхность зеркала отполирована неидеально, то наблюдается зависимость интенсивности отраженного света от длины волны — чем больше длина волны, тем лучшее отражение. Например, красные лучи отражаются сильнее, чем синие.

При наличии шероховатости есть зависимость интенсивности отраженного света от угла падения. Отражение света максимально для углов Э, близких к 90 градусам.

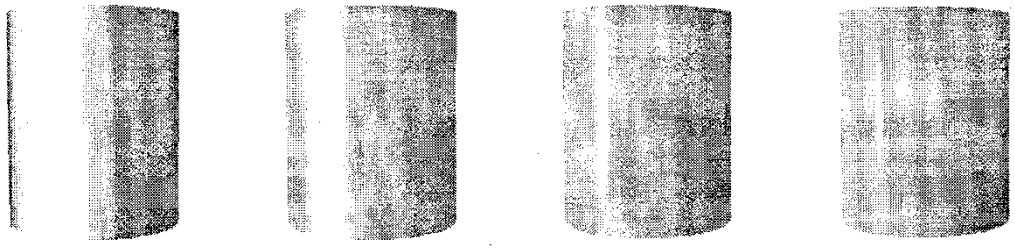


Рис. 8.2. Примры для разных значений показателя р модели Фонга

Падающий луч, попадая на шероховатую поверхность реального зеркала, порождает не один отраженный луч, а несколько лучей, которые рассеиваются в разных направлениях. Зона рассеивания зависит от качества полировки и может быть описана некоторым законом распределения. Как правило, форма зоны рассеивания симметрична относительно линии идеального зеркально отраженного луча. К числу простейших, но довольно часто используемых, относится эмпирическая *модель Фонга,* соответственно которой интенсивность зеркально отраженного излучения пропорциональна (cosa) где а — угол отклонения от линии идеально отраженного луча. Показатель находится в диапазоне от 1 до 1000 и зависит от качества полировки. Запишем это таким образом:

$$Is = I \cos^{p} \alpha,$$

где *I* - интенсивность излучения источника

Диффузное отражение. Этот вид отражения присущ *матовым* поверхностям. Матовой можно считать такую поверхность, размер шероховатостей которой уже настолько большой, что падающий луч рассеивается равномерно во все стороны. Такой тип отражения характерен, например, для гипса, песка, бумаги. Диффузное отражение описывается *законом Ламберта,* согласно которому интенсивность отраженного света пропорциональна косинусу угла между направлением на точечный источник света и нормалью к поверхности:

$$Id = I \cos\theta,$$

где *I*— интенсивность источника света.

При создании реалистических изображений следует учитывать то, что в природе, вероятно, не существуют идеально зеркальные или идеально матовые поверхности. При изображении объектов средствами компьютерной графики обычно моделируют сочетание зеркальности и диффузного рассеивания в пропорции, характерной для конкретного материала. В этом случае модель отражения записывают в виде взвешенной суммы диффузной и зеркальной составляющих:

$$I_{рез} = I_d K_d + I_s K_s = I (K_d \cos\theta + K_s \cos^{p}\alpha),$$

где константы *Kd* и *Кс* определяют отражательные свойства определенного материала. Константы *Kd* и *Кс* обычно принимают значения в диапазоне от 0 до 1, причем *Kd* + *Кс* = *1.*

Соответственно этой формуле интенсивность отраженного света равняется нулю для некоторых углов θ и α. Однако в реальных сценах обычно нет абсолютно затемненных объектов, следует учитывать фоновое подсвечивание — освещение рассеянным светом, отраженным от других объектов (рис. 8.3).

Рис. 8.3. Модель отражения света и три ее составляющие

В таком случае интенсивность может быть эмпирически выражена следующей формулой:

$$Iрез = IaKa + IdKd + IsKs = IaKa + I\,(Kd\cos\theta + Ks\cos^{p}\alpha)$$

где *Ia* - интенсивность рассеянного света, K_a — константа.

Можно еще усовершенствовать модель отражения, если учесть то, что энергия от источника света уменьшается соответственно расстоянию до него

$$I_{рез} = I_a K_a + I\,F(R)(K_d\cos\theta + K_s\cos^{p}\alpha),$$

где *R* - расстояние от источника света до поверхности, *F(R)* — функция ослабления. Для точечного источника света энергия излучения уменьшается пропорционально квадрату расстояния. На практике обычно используют линейную аппроксимацию функции ослабления.

Для нескольких источников света диффузную и зеркальную составляющую рассчитывают в отдельности для каждого источника, а результат — сумма всех составляющих:

$$I_{рез} = I_a K_a + K_d\sum_j I_d(j) + K_S\sum_j I_S(j).$$

8.1.2 Вычисление нормалей и углов отражения

Вычисление координат вектора нормали. Рассматривая модели отражения света, вы, наверное, обратили внимание на то, что нормаль к поверхности — важный элемент. Определение вектора нормали к поверхности в заданной точке может быть выполнено разными способами. В значительной степени это определяется типом модели описания поверхности. Для поверхностей, заданных в аналитической форме, известны методы дифференциальной геометрии, которые основываются на вычислении частных производных функций описания. Например, если поверхность задана параметрическими функциями

$$x = x\,(s,\ t),$$
$$y = y\,(s,\ t),$$
$$z = z\,(s,\ t),$$

тогда координаты вектора нормали можно вычислить так:

$$x_N = \begin{vmatrix} \begin{vmatrix} \frac{\partial y}{\partial s} & \frac{\partial z}{\partial s} \\ \frac{\partial y}{\partial t} & \frac{\partial z}{\partial t} \end{vmatrix} \end{vmatrix} = \frac{\partial y}{\partial s} \cdot \frac{\partial z}{\partial t} - \frac{\partial y}{\partial t} \cdot \frac{\partial z}{\partial s}, \quad y_N = \begin{vmatrix} \begin{vmatrix} \frac{\partial z}{\partial s} & \frac{\partial x}{\partial s} \\ \frac{\partial z}{\partial t} & \frac{\partial x}{\partial t} \end{vmatrix} \end{vmatrix} = \frac{\partial z}{\partial s} \cdot \frac{\partial x}{\partial t} - \frac{\partial z}{\partial t} \cdot \frac{\partial x}{\partial s},$$

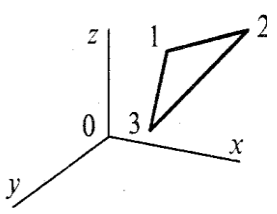
$$z_N = \begin{vmatrix} \begin{vmatrix} \frac{\partial x}{\partial s} & \frac{\partial y}{\partial s} \\ \frac{\partial x}{\partial t} & \frac{\partial y}{\partial t} \end{vmatrix} \end{vmatrix} = \frac{\partial x}{\partial s} \cdot \frac{\partial y}{\partial t} - \frac{\partial x}{\partial t} \cdot \frac{\partial y}{\partial s}.$$

В случае описания поверхности векторно-полигональной моделью для определения нормалей можно использовать методы векторной алгебры.

Пусть в пространстве задана некоторая многогранная поверхность. Рассмотрим одну ее плоскую грань, имеющую вид треугольника (рис. 8.4).

Для вычисления координат вектора нормали воспользуемся векторным произведением любых двух векторов, которые лежат в плоскости грани. Такими векторами могут служить и ребра грани, например, ребра 1-2 и 1-3. Однако формулы для векторного произведения были определены нами только для радиус-векторов. Чтобы перейти к радиус-векторам, введем новую систему координат, центр которой совпадает с вершиной 1, а оси — параллельны осям бывшей системы. Координаты вершин в новой системе:

$$x'_i = x_i - x_1,$$
$$y'_i = y_i - y_1,$$
$$z'_i = z_i - z_1.$$



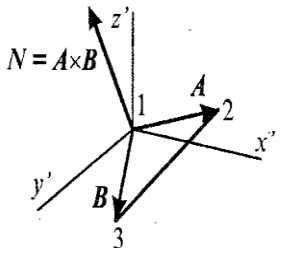


Рис. 8.4. Одна грань поверхности Радиус-векторы

Теперь назовем ребро (1-2) вектором *A,* а ребро (1-3) — вектором *B,* как показано на рис. 8.4. Таким образом, положение нормали к грани в пространстве будет описываться радиус-вектором *N.* Его координаты в системе *(x′, y′, z′)* выразим формулами для векторного произведения

$$x'_N = (y_2 - y_1)(z_3 - z_1) - (z_2 - z_1)(y_3 - y_1),$$
$$y'_N = (z_2 - z_1)(x_3 - x_1) - (x_2 - x_1)(z_3 - z_1),$$
$$z'_N = (x_2 - x_1)(y_3 - y_1) - (y_2 - y_1)(x_3 - x_1).$$

Плоская грань может быть изображена в разных ракурсах. В каждой конкретной ситуации необходимо выбирать направление нормали, которое соответствует *видимой стороне* грани. Если плоская грань может быть видна с обратной стороны, то тогда в расчетах отраженного света необходимо выбирать для нормали обратный вектор, то есть (-N).

Если полигональная поверхность имеет не треугольные грани, а, например, плоские четырехугольные, то расчет нормали можно выполнять по любым трем вершинам грани.

Диффузное отражение. Рассчитаем косинус угла между вектором нормали и направлением на источник света. Это можно выполнить таким способом.

Сначала необходимо определить радиус-вектор, направленный на источник света. Обозначим его как S. Потом для вычисления косинуса угла между радиус-векторами S и N воспользуемся формулами скалярного произведения векторов. Поскольку

$$\boldsymbol{S} \cdot \boldsymbol{N} = |\boldsymbol{S}||\boldsymbol{N}| \cos\theta,$$

а также

$$\boldsymbol{S} \cdot \boldsymbol{N} = x_S x_N + y_S y_N + z_S z_N,$$

то получим

$$\cos\theta = \frac{x_S x_N + y_S y_N + z_S z_N}{|\boldsymbol{S}|\,|\boldsymbol{N}|}.$$

Для упрощения вычислений целесообразно использовать векторы S и N единичной длины, то есть $|S|*|N| = 1$.

Использование скалярного произведения здесь можно считать универсальным методом, который можно использовать для любого расположения точечного источника света. В отдельных случаях можно рассчитать косинус угла падения по-иному. Например, если источник света располагается на оси *Z* видовых координат в бесконечности позади камеры, тогда косинус угла нормали к грани с осью Z равняется отношению координаты г и длины радиус-вектора нормали

$$\cos\theta = \frac{z_N}{|\boldsymbol{N}|} = \frac{z_N}{\sqrt{x_R^2 + y_R^2 + z_R^2}}.$$

Зеркальное отражение. Будем считать, что задан радиус-вектор S, направленный на источник света, а также известен радиус-вектор нормали N. Нужно найти косинус угла между отраженным лучом и направлением камеры.

Сначала необходимо вычислить радиус-вектор отраженного луча. Обозначим его как *R.* Выполним некоторые геометрические построения, как показано на рис. 8.5.

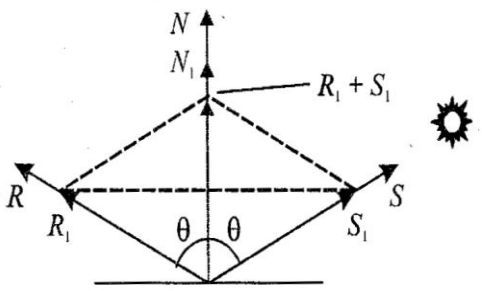


Рис. 8.5. Векторы R1, S1 и N1 –единичной длины

Для решения нашей задачи сначала рассмотрим единичные векторы *R1, S1*, и *N1*. Поскольку векторы нормали, падающего луча и отраженного луча находятся в одной плоскости, то можно записать $R1 + S1 = N^1$, где N^1 — это вектор, который соответствует диагонали ромба и совпадает по уаправлению с нормалью. Длина вектора N1 равняется 2cosθ. Поскольку вектор N^1 по направлению совпадает с N1, то

$$\boldsymbol{N'} = \boldsymbol{N}_1 2\cos\theta$$

или

$$\boldsymbol{R}_1 + \boldsymbol{S}_1 = \boldsymbol{N}_1 2\cos\theta.$$

Отсюда находится единичный вектор отраженного луча:

$$\boldsymbol{R}_1 = \boldsymbol{N}_1 2\cos\theta - \boldsymbol{S}_1 = \frac{\boldsymbol{N}}{|\boldsymbol{N}|} 2\cos\theta - \frac{\boldsymbol{S}}{|\boldsymbol{S}|}.$$

Найдем cosθ. Для этого используем скалярное произведение векторов *N* и *S*:

$$\cos\theta = \frac{\boldsymbol{N} \cdot \boldsymbol{S}}{|\boldsymbol{N}||\boldsymbol{S}|}.$$

Подставим это значение в выражение для R_1:

$$\boldsymbol{R}_1 = \boldsymbol{N} \cdot 2\frac{\boldsymbol{N} \cdot \boldsymbol{S}}{|\boldsymbol{N}|^2|\boldsymbol{S}|} - \frac{\boldsymbol{S}}{|\boldsymbol{S}|}.$$

Полагая, что искомый вектор отраженного луча будет иметь такую же длину, что и вектор падающего луча, т.е. $R = S\,R_1$, получим:

$$\boldsymbol{R} = \boldsymbol{N} \cdot 2\frac{\boldsymbol{N} \cdot \boldsymbol{S}}{|\boldsymbol{N}|^2} - \boldsymbol{S}.$$

Это решение в векторной форме. Запишем координаты вектора *R:*

$$x_R = 2x_N \frac{x_S x_N + y_S y_N + z_S z_N}{x_N^2 + y_N^2 + z_N^2} - x_S,$$

$$y_R = 2y_N \frac{x_S x_N + y_S y_N + z_S z_N}{x_N^2 + y_N^2 + z_N^2} - y_S,$$

$$z_R = 2z_N \frac{x_S x_N + y_S y_N + z_S z_N}{x_N^2 + y_N^2 + z_N^2} - z_S.$$

Теперь осталось найти косинус угла между отраженным лучом и направлением камеры. Пусть K – радиус-вектор, направленный на камеру. Найдем искомый косинус угла:

$$\cos\theta = \frac{\boldsymbol{K} \cdot \boldsymbol{R}}{|\boldsymbol{K}||\boldsymbol{R}|} = \frac{x_K x_R + y_K y_R + z_K z_R}{\sqrt{x_K^2 + y_K^2 + z_K^2}\sqrt{x_R^2 + y_R^2 + z_R^2}}.$$

Для упрощения вычислений целесообразно задавать векторы S, N и R единичной длины (тогда и R будет единичным).

8.2 Метод Гуро

Этот метод предназначен для создания иллюзии гладкой криволинейной поверхности, которая описана в виде многогранников или полигональной сетки с плоскими гранями. Если каждая плоская грань имеет один постоянный цвет, который определен в соответствии с учетом отражения, то разные цвета соседних граней очень заметны, и поверхность выглядит именно как многогранник. Казалось, этот дефект можно замаскировать за счет увеличения количества граней при аппроксимации поверхности. Но зрение человека имеет способность подчеркивать перепады яркости на границах смежных граней — такой эффект называется эффектом *полос Маха.* Вследствие этого, для создания иллюзии гладкости нужно намного увеличить количество граней, что приводит к существенному замедлению визуализации — чем больше граней, тем меньше скорость рисования объектов.

Метод Гуро основан на идее закрашивания каждой плоской грани не одним цветом, а плавно изменяющимися оттенками, которые вычисляются путем интерполяции цветов прилегающих граней. Закрашивание граней по методу Гуро осуществляется в четыре этапа.

- Вычисляются нормали к каждой грани.
- Определяются нормали в вершинах. Нормаль в вершине определяется усреднением нормалей прилегающих граней (рис. 8.6).

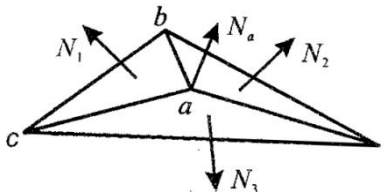


Рис. 8.6. Нормаль в вершине *a*

На основе нормалей в вершинах вычисляются значения интенсивностей в вершинах в соответствии с выбранной моделью отражения света. Закрашиваются полигоны граней цветом, который соответствует линейной интерполяции значений интенсивности в вершинах.

Вектор нормали в вершине *(a)* равняется

$$N_a = \frac{N_1 + N_2 + N_3}{3}.$$

Определение интерполированных значений интенсивности отраженного света в каждой точке грани (и, следовательно, цвет каждого пиксела) удобно выполнять во время цикла заполнения полигона. Рассмотрим заполнение контура грани горизонталями в экранных координатах (рис. 8.7).

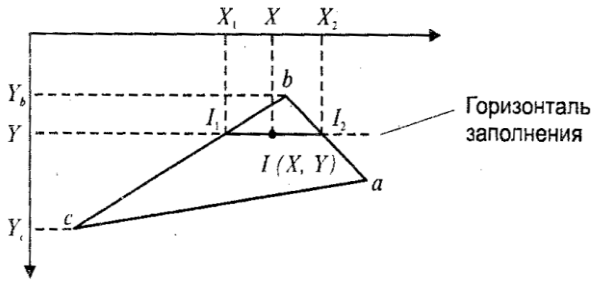


Рис. 8.7. Заполнение полигона грани

Интерполированная интенсивность *I* в точке *(X, У)* определяется, исходя из пропорции

$$\frac{I-I_1}{X-X_1}=\frac{I_2-I_1}{X_2-X_1}.$$

откуда

$$I=I_1+\frac{(I_2-I_1)(X-X_1)}{X_2-X_1}.$$

Значение интенсивности I_1 и I_2 на концах горизонтального отрезка вычислим путем интерполяции интенсивности в вершинах

$$\frac{I_1-I_b}{Y-Y_b}=\frac{I_c-I_b}{Y_c-Y_b},\qquad \text{или}\qquad I_1=I_b+\frac{(I_c-I_b)(Y-Y_b)}{Y_c-Y_b},$$

$$\frac{I_2-I_b}{Y-Y_b}=\frac{I_a-I_b}{Y_a-Y_b},\qquad\qquad I_2=I_b+\frac{(I_a-I_b)(Y-Y_b)}{Y_a-Y_b}.$$

8.3 Метод Фонга

Аналогичен методу Гуро, но при использовании метода Фонга для определения цвета в каждой точке интерполируются не интенсивности отраженного света, а векторы нормалей.

- Определяются нормали к граням.
- По нормалям к граням определяются нормали в вершинах. В каждой точке закрашиваемой грани определяется интерполированный вектор нормали.
- Цвет каждой точки грани вычисляется в соответствии с направлением интерполированного вектора нормали и согласно выбранной модели отражения света.

Метод Фонга сложнее метода Гуро. Для каждой точки (пиксела) поверхности необходимо выполнять намного больше вычислительных операций. Тем не менее, он дает значительно лучшие результаты, в особенности при имитации зеркальных поверхностей.

Общие черты и отличия методов Гуро и Фонга можно показать на примере образца цилиндрической поверхности, которая аппроксимирована плоскими гранями (рис. 8.8). Пусть точечный источник света находится позади нас. Проанализируем результаты закрашивания граней поверхности для имитации отражения света.

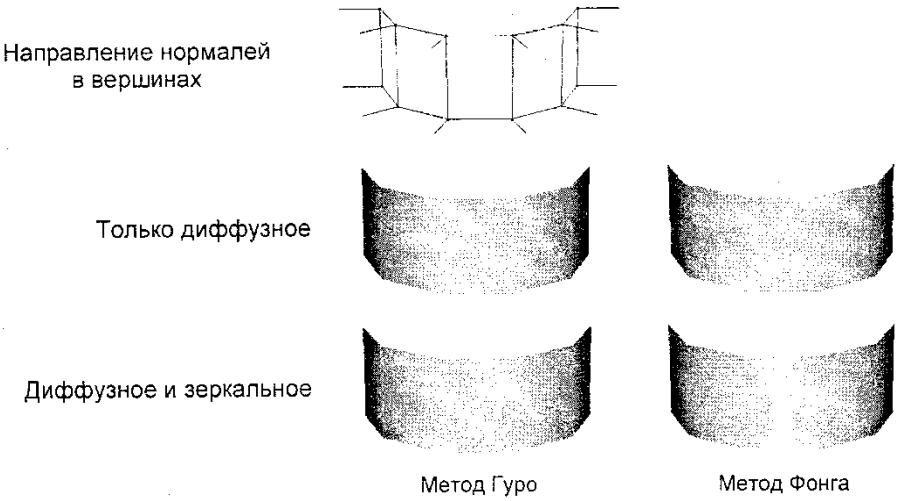


Рис. 8.8. Отличия закрашивания методами Гуро и Фонга

Основные отличия можно заметить, рассматривая закрашивания передней грани. Она перпендикулярна направлению лучей света. Поэтому нормали в вершинах этой грани располагаются симметрично — они образуют попарно равные по абсолютной величине углы с лучами света. Для метода Гуро это обуславливает одинаковые интенсивности отраженного света в вершинах передней грани. А раз интенсивности для всех вершин одинаковые, то цвет всех точек этой грани — константа (при линейной интерполяции), что, в данном случае, неправильно.

8.4. Имитация микрорельефа

Пусть нам необходимо показать поверхность, изобилующую мелкими неровностями. Можно попытаться создать полигональную модель, аппроксимирующую все видимые детали рельефа, вплоть до мельчайших бугорков, ямок, трещин и т. п. Однако это может потребовать такого количества треугольников, которое не в состоянии поддержать компьютерная система.

Для визуализации таких поверхностей часто используется следующий метод. Общие очертания поверхности моделируются полигонами, а имитация мелких деталей рельефа производится с помощью текстур.

Рассмотрим один из популярных в настоящее время методов рельефного текстурирова-ния — *DOT3*. Согласно этому методу, каждый тексел текстуры хранит координаты вектора нормали для соответствующей точки поверхности (рис. 8.9).

Почему лучше использовать карту нормалей, а не само изображение в виде текстуры? Потому что, это дает возможность создавать иллюзию игры света и тени при смене ракурса показа и возможном движении источников света, и даже при деформациях поверхности. Обычная текстура это обеспечить не в состоянии — она представляет изображение, сделанное только для одного ракурса показа и фиксированного положения источников света.

Рис. 8.9. Пимер имитации микрорельефа методом DOT3 Bump Mapping

Для хранения координат нормалей удобно использовать 24-битный растровый графический формат файлов — тут каждая тройка *RGB* будет представлять тройку координат *UVN*. Координаты *UVN* описывают векторы нормалей в так называемой касательной или тангенциальной системе координат (рис. 8.10).

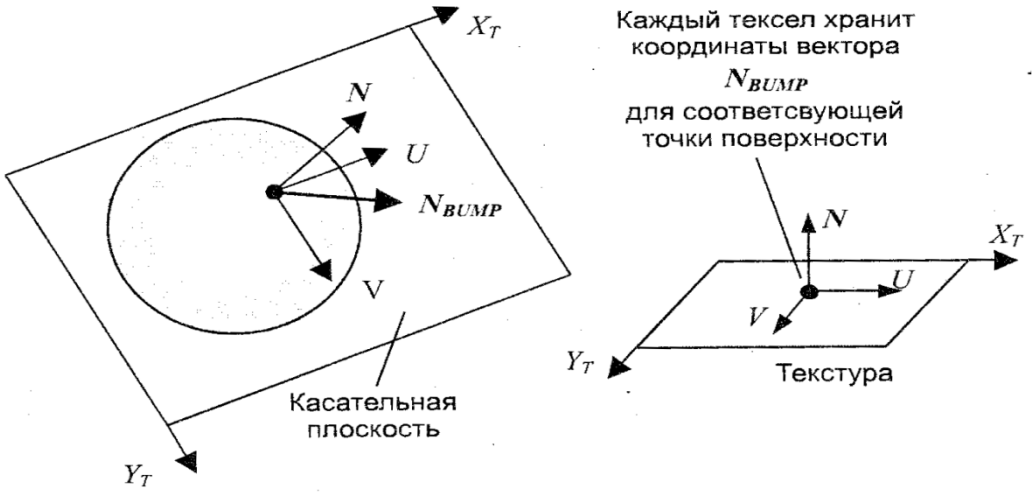


Рис. 8.10. Используемые системы координат

У системы координат *UVN* оси *U* и *N* лежат в плоскости текстуры, они параллельны осям текстурных координат X_T и Y_T, а ось *N* является нормалью к поверхности. Пусть задан вектор направления света *L*. Тогда, при расчете цвета пиксела поверхности, например, для диффузной модели отражения, можно использовать скалярное произведение векторов *L* и *Nbump:*

Цвет точки = (собственный цвет поверхности) x DOT3(L, Nbump)

где *DOT* — часто используемое в англоязычной литературе обозначение для скалярного произведения *(bot product)* векторов (отсюда и название *DOT3 Bump Mapping).*

Очевидно, что для корректного вычисления результата необходимо использовать в скалярном произведении одну и ту же систему координат для векторов *L и Nbump* . Для этого нужно привести вектор *Nbump* систему координат, описывающую вектор *L* (обычно это мировые координаты). А можно наоборот, выразить вектор *L* в системе координат *UVN*.

Преломление света

Законы преломления света следует учитывать при построении изображений прозрачных объектов.

Модель идеального преломления. Согласно этой модели луч отклоняется на границе двух сред, причем падающий луч, преломленный луч и нормаль лежат в одной плоскости (в этой же плоскости лежит и зеркально отраженный луч).

Обозначим угол между падающим лучом и нормалью как α_1, а угол между нормалью и преломленным лучом как α_2. Для этих углов известен закон *Снеллиуса,* согласно которому

$$n_1 \sin \alpha_1 = n_2 \sin \alpha_2$$

где n_1 и n_2 — ***абсолютные показатели преломления*** соответствующих сред.

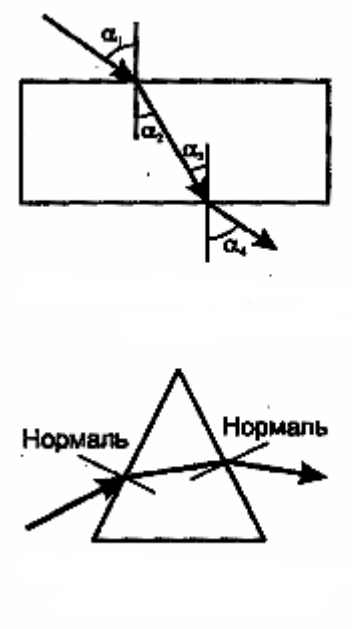


Рис. 8.11. Преломление луча и преломление в треугольной призме

На рис. 8.11 изображен пример отклонения луча при преломлении. В данном случае границей раздела сред служат две параллельные плоскости (например, при прохождении луча через толстое стекло). Очевидно, что угол α_1 равняется углу α_4, а угол α_2 равняется углу α_3. Другими словами, после прохождения сквозь стекло луч будет параллельно смещен. Это смещение зависит от толщины стекла и соотношения показателей преломление сред. Возможно, это простейший пример преломления. Вы, наверное, уже наблюдали и более сложные объекты, например, треугольную призму. Для нее границами сред являются непараллельные плоскости. Прозрачные объекты могут иметь и криволинейные поверхности (например, линзы в разнообразных оптических приборах).

Принято считать, что для вакуума абсолютный показатель преломления равняется единице. Для воздуха он составляет 1.00029, для воды — 1.33, для стекла разных сортов: 1.52 (легкий крон), 1.65 (тяжелый крон). Показатель преломления зависит от состояния вещества, например, от температуры. На практике обычно используют отношение показателей преломления двух сред *(n1/n2),* которое называют ***относительным показателем преломления.***

Кроме идеального преломления, в компъютерной графике используется **диффузное преломление.** Соответственно этой модели, падающий луч преломляется во все стороны. Примером может служить молочное стекло, обледеневшее стекло.

8.5 Трассировка лучей

Методы трассировки лучей *(Ray Tracing)* на сегодняшний день считаются наиболее мощными и универсальными методами создания реалистичных изображений. Известно много примеров реализации алгоритмов трассировки для качественного отображения самых сложных трехмерных сцен. Можно отметить, что универсальность методов трассировки в значительной мере обусловлена тем, что в их основе лежат простые и ясные понятия, которые отражают наш опыт восприятия окружающего мира.

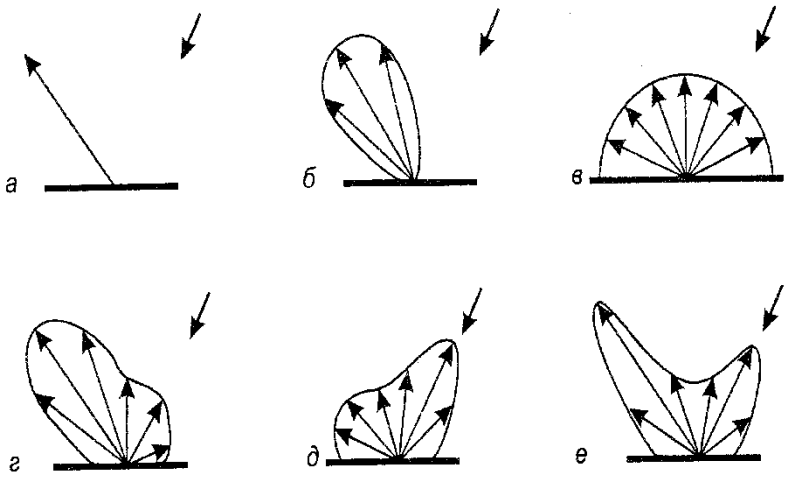


Рис. 8.12. Модели отражения: а – идеальное зеркало, б - неидеальное зеркало, в – диффузное, г – сумма диффузного и зеркального, д – обратное, е - сумма диффузного, зеркального и лбратного

Как мы видим окружающую реальность? Во-первых, нужно определиться с тем, что мы вообще способны видеть. Это изучается в специальных дисциплинах, а до некоторой степени, это вопрос философский. Но здесь мы будем считать, что окружающие объекты обладают такими свойствами относительно света:

- излучают;
- отражают и поглощают;
- пропускают сквозь себя.

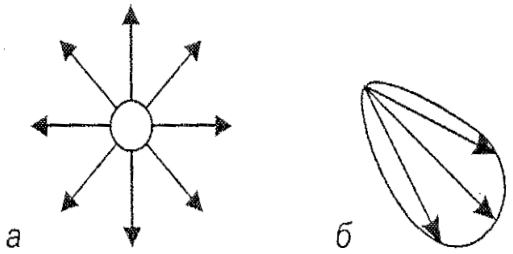


Рис. 8.13. Излучение – а – раномерно во все тороны, б - направленно

Каждое из этих свойств можно описать некоторым набором характеристик. Например, излучение можно охарактеризовать интенсивностью, направленностью, спектром. Излучение может исходить от условно точечного источника (далекая звезда) или от источника рассеянного света (скажем, от расплавленной лавы, извергающейся из кратера вулкана). Распространение излучения может осуществляться вдоль довольно узкого луча (сфокусированный луч лазера) или конусом (прожектор), или равномерно во все стороны (Солнце), или еще как-то. Свойство отражения (поглощение) можно описать характеристиками диффузного рассеивания и зеркального отражения. Прозрачность можно описать ослаблением интенсивности и преломлением.

Распределение световой энергии по возможным направлениям световых лучей можно отобразить с помощью векторных диаграмм, в которых длина векторов соответствует интенсивности (рис. 8.12 – 8.14).

В предшествующих параграфах мы с вами уже ознакомились с видами отражения, которые упоминаются наиболее часто — зеркальным и диффузным. Реже в литературе поминается обратное зеркальное или *антизеркальное отражение,* в котором максимум интенсивности отражения соответствует направлению на источник. Обратное зеркальное отражение имеют некоторые виды растительности на поверхности Земли, наблюдаемые с высоты рисовые поля.

Два крайних, идеализированных случая преломления изображены на рис. 8.13.

Некоторые реальные объекты преломляют лучи намного более сложным образом, например, обледеневшее стекло.

Один и тот же объект реальной действительности может восприниматься как источник света, а может, при ином рассмотрении, считаться предметом, только отражающим и пропускающим свет. Например, купол облачного неба в некоторой трехмерной сцене может моделироваться в виде протяженного (распределенного) источника света, а в других моделях это же небо выступает как полупрозрачная среда, освещенная со стороны Солнца.

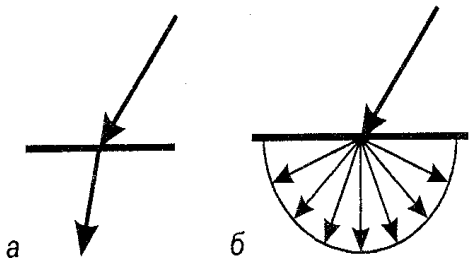


Рис. 8.14. Преломление а – идеальное, б - дифузное

В общем случае каждый объект описывается некоторым сочетанием вышеперечисленных трех свойств. В качестве упражнения попробуйте привести пример объекта, который обладает одновременно тремя указанными свойствами — сам излучает свет и, в то же время, отражает, а также пропускает свет от других источников. Вероятно, ваше воображение подскажет и другие примеры, нежели, скажем, раскаленное докрасна стекло.

Теперь рассмотрим то, как формируется изображение некоторой сцены, которая содержит несколько пространственных объектов. Будем считать, что из точек поверхности (объема) излучаемых объектов выходят лучи света. Можно назвать такие лучи первичными — они освещают все другое.

Важным моментом является предположение, что световой луч в свободном пространстве распространяется ***вдоль прямой линии*** (хотя в специальных разделах физики изучаются также и причины возможного искривления). Но в *геометрической оптике* принято, что луч света распространяется прямолинейно до тех пор, пока не встретится отражающая поверхность или граница среды преломления. Так будем полагать и мы.

От источников излучения исходит по разным направлениям бесчисленное множество первичных лучей (даже луч лазера невозможно идеально сфокусировать — все равно свет будет распространяться не одной идеально тонкой линией, а конусом, пучком лучей). Некоторые лучи уходят в свободное пространство, а некоторые (их также бесчисленное множество) попадают на другие объекты. Если луч попадет в прозрачный объект, то, преломляясь, он идет дальше, при этом некоторая часть световой энергии поглощается. Подобно этому, если на пути луча встречается зеркально отражающая поверхность, то он также изменяет направление, а часть световой энергии поглощается. Если объект зеркальный и одновременно прозрачный (например, обычное стекло), то будут уже два луча — в этом случае говорят, что луч расщепляется.

Можно сказать, что в результате воздействия на объекты первичных лучей возникают вторичные лучи. Бесчисленное множество вторичных лучей уходит в свободное пространство, но некоторые из них попадают на другие объекты. Так, многократно отражаясь и преломляясь, отдельные световые лучи приходят в точку наблюдения — глаз человека или оптическую систему камеры. Очевидно, что в точку наблюдения может попасть и часть первичных лучей непосредственно от источников

излучения. Таким образом, изображение сцены формируется некоторым множеством световых лучей.

Цвет отдельных точек изображения определяется спектром и интенсивностью первичных лучей источников излучения, а также поглощением световой энергии в объектах, встретившихся на пути соответствующих лучей.

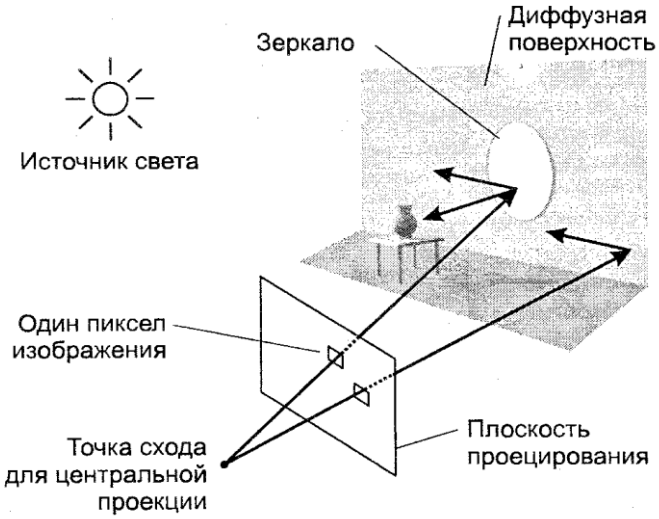


Рис. 8.15. Схема обратной трассировки лучей

Непосредственная реализация данной лучевой модели формирования изображения представляется затруднительной. Можно попробовать разработать алгоритм построения изображения указанным способом. В таком алгоритме необходимо предусмотреть перебор всех первичных лучей и определить, какие из них попадают в объекты и в камеру. Потом выполнить перебор всех вторичных лучей, и также учесть только те, которые попадают в объекты и в камеру. И так далее. Можно назвать такой метод *прямой* трассировкой лучей. Практическая ценность такого метода вызовет сомнение. В самом деле, как учитывать бесконечное множество лучей, идущих во все стороны? Очевидно, что полный перебор бесконечного числа лучей в принципе невозможен. Даже если каким-то образом свести это к конечному числу операций (например, разделить всю сферу направлений на угловые секторы и оперировать уже не бесконечно тонкими линиями, а секторами), все равно остается главный недостаток метода — много лишних операций, связанных с расчетом лучей, которые потом не используются. Так, во всяком случае, это представляется в настоящее время.

Метод ***обратной трассировки*** лучей позволяет значительно сократить перебор световых лучей. Метод разработан в 80-х годах, основополагающими считаются работы *Уитте-да* и *Кэя*. Согласно этому методу отслеживание лучей осуществляется не от источников света, а в обратном'направлении — от точки наблюдения. Так учитываются только те лучи, которые вносят вклад в формирование изображения.

Рассмотрим, как можно получить растровое изображение некоторой трехмерной сцены методом обратной трассировки. Предположим, что плоскость проецирования разбита на множество квадратиков — пикселов. Выберем центральную проекцию с центром схода на некотором расстоянии от плоскости проецирования. Проведем прямую линию из центра схода через середину квадратика (пиксела) плоскости проецирования (рис. 8.15). Это будет первичный луч обратной трассировки. Если прямая линия этого луча попадает в один или несколько объектов сцены, то выбираем ближайшую точку пересечения. Для определения цвета пиксела изображения нужно учитывать свойства объекта, а также то, какое световое излучение приходится на соответствующую точку объекта.

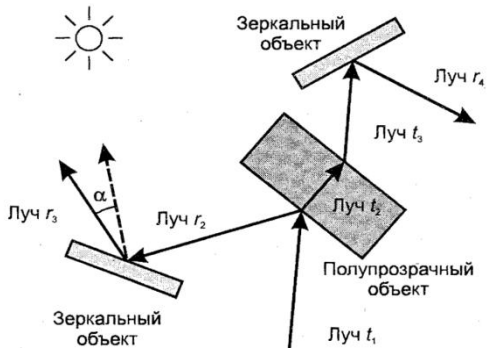


Рис. 8.16. Обратная трассировка для объектов, имеющих свойства зеркального отражения и преломления

Если объект зеркальный (хотя бы частично), то строим вторичный луч — луч падения, считая лучом отражения предыдущий, первичный, трассируемый луч. Выше мы рассматривали зеркальное отражение и получили формулы для вектора отраженного луча по заданным векторам нормали и луча падения. Но здесь нам известен вектор отраженного луча, а как найти вектор падающего луча? Для этого можно использовать ту же формулу зеркального отражения, но определяя необходимый вектор луча падения как отраженный луч. То есть отражение наоборот.

Для идеального зеркала достаточно потом проследить лишь очередную точку пересечения вторичного луча с некоторым объектом. Что означает термин "идеальное зеркало"? Будем считать, что такое зеркало имеет идеально равную отполированную поверхность, поэтому одному отраженному лучу соответствует только один падающий луч. Зеркало может быть затемненным, то есть поглощать часть световой энергии, но все равно выполняется правило: один луч падает — один отражается. Можно рассматривать также "неидеальное зеркало". Это будет означать, что поверхность неровная. Направлению отраженного луча будут соответствовать несколько падающих лучей (или наоборот, один падающий луч порождает несколько отраженных лучей), которые образуют некоторый конус, возможно, несимметричный, с осью вдоль линии падающего луча идеального зеркала. Конус соответствует некоторому закону распределения интенсивностей, простейший из которых описывается моделью Фонга — косинус угла, возведенный в некоторую степень. Неидеальное зеркало резко усложняет трассировку — нужно проследить не один, а множество падающих лучей, учитывать взнос излучения от других видимых из данной точки объектов.

Если объект прозрачный, то необходимо построить новый луч, такой, который при преломлении давал бы предшествующий трассируемый луч. Здесь также можно воспользоваться обратимостью, которая справедлива и для преломления. Для расчета вектора искомого луча можно применить рассмотренные выше формулы для вектора луча преломления, считая, что преломление происходит в обратном направлении (рис. 8.16).

Если объект обладает свойствами диффузного отражения и преломления, то, в общем случае, как и для неидеального зеркала, необходимо трассировать лучи, которые приходят от всех имеющихся объектов. Для диффузного отражения интенсивность отраженного света, как известно, пропорциональна косинусу угла между вектором луча от источника света и нормалью. Здесь источником света может выступать любой видимый из данной точки объект, способный передавать световую энергию.

Если выясняется, что текущий луч обратной трассировки не пересекает любой объект, а направляется в свободное пространство, то на этом трассировка для этого луча заканчивается.

Обратная трассировка лучей в том виде, в котором мы ее здесь рассмотрели, хотя и сокращает перебор, но не позволяет избавиться от бесконечного числа анализируемых лучей. В самом деле, данный метод позволяет сразу получить для каждой точки изображения один первичный луч обратной трассировки. Однако вторичных лучей отражения уже может быть бесконечное число. Так, например, если объект может

отражать свет от любого другого объекта, и если эти другие объекты имеют довольно большие размеры, то какие именно точки излучающих объектов нужно учитывать для построения соответствующих лучей, например, при диффузном отражении? Очевидно, все точки.

При практической реализации метода обратной трассировки вводят ограничения. Некоторые из них необходимы, чтобы можно было в принципе решить задачу синтеза изображения, а некоторые ограничения позволяют значительно повысить быстродействие трассировки. Примеры таких ограничений.

1. Среди всех типов объектов выделяются некоторые, которые назовем ***источниками*** *света.* Источники света могут только излучать свет, но не могут его отражать или преломлять (будем рассматривать только *точечные* источники света).

2. Свойства отражающих поверхностей описываются суммой двух компонентов — диффузного и зеркального.

3. В свою очередь, зеркальность также описывается двумя составляющими. Первая *(reflection)* учитывает отражение от других объектов, которые не являются источниками света. Строится только один зеркально отраженный луч *r* для дальнейшей трассировки. Вторая составляющая ***(Specular)*** означает световые блики от источников света. Для этого направляются лучи на все источники света и определяются углы, образованные этими лучами с зеркально отраженным лучом обратной трассировки *(r).* При зеркальном отражении цвет точки поверхности определяется цветом того, что отражается. В простейшем случае зеркало не имеет собственного цвета поверхности.

4. При диффузном отражении учитываются только лучи от источников света . Лучи от зеркально отражающих поверхностей игнорируются. Если луч, направленный на данный источник света, закрывается другим объектом, значит, данная точка объекта находится в тени. При диффузном отражении цвет освещенной точки поверхности определяется собственным цветом поверхности и цветом источников света.

5. Для прозрачных *(1гап5раген()* объектов обычно не учитывается зависимость коэффициента преломления от длины волны. Иногда прозрачность вообще моделируют без преломления, то есть направление преломленного луча *I* совпадает с направлением падающего луча.

6. Для учета освещенности объектов светом, который рассеивается другими объектами, вводится фоновая составляющая *(ambient).*

7. Для завершения трассировки вводят некоторое предельное значение освещенности, которое уже не должно вносить взнос в результирующий цвет, или ограничивают количество итераций.

Согласно *модели Уиттеда* цвет некоторой точки объекта определяется суммарной интенсивностью

$$I(\lambda) = KaIa(\lambda)C(\lambda) + KdId(\lambda)C(\lambda) + KsIs(\lambda) + KrIr(\lambda) + KtIt(\lambda)$$

где λ - длина волны,

$C(\lambda)$ - заданный исходный цвет точки объекта,

K_a, K_d, K_s, K_r и K_t — коэффициенты, учитывающие свойства конкретного объекта через параметры фонового подсвечивания, диффузного рассеивания, зеркальности, отражения и прозрачности,

I_a - интенсивность фонового подсвечивания,

I_d - интенсивность, учитываемая для диффузного рассеивания,

I_s - интенсивность, учитываемая для зеркальности,

I_r - интенсивность излучения, приходящего по отраженному лучу,

I_t - интенсивность излучения, приходящего по преломленному лучу.

Интенсивность фонового подсвечивания (1_a) для некоторого объекта обычно константа. Запишем формулы для других интенсивностей. Для диффузного отражения

$$I_d = \sum_i I_i(\lambda)\cos\theta_i$$

где $I_i(\lambda)$ — интенсивность излучения *i-го* источника света, θ_i — угол между нормалью к поверхности объекта и направлением на *i-vi* источник света.

Для зеркальности:

$$I_d = \sum_i I_i(\lambda)\cos^p \alpha i$$

где p - показатель степени от единицы до нескольких сотен (согласно модели Фонга), α_i - угол между отраженным лучом (обратной трассировки) и направлением на г'-й источник света.

Интенсивности излучений проходящих по отраженному лучу (I_r), а так же по преломленному лучу (I_t), умножают на коэффициент, учитывающий ослабление интенсивности в зависимости от расстояния, пройденного лучом. Такой коэффициент записывается в виде $e^{-\beta d}$ где d - пройденное расстояние, β – параметр ослабления, учитывающий свойства среды, в которой распространяется луч.

Для первичного луча необходимо задать направление, которое соответствует избранной проекции. Если проекция центральная, то первичные лучи расходятся из общей точки, для параллельной проекции первичные лучи — параллельные. Луч можно задать, например, координатами начальной и конечной точек отрезка, координатой начальной точки и направлением, или еще как-нибудь. ***Задание первичного луча однозначно определяет проекцию изображаемой сцены.*** При обратной трассировке лучей любые преобразования координат вообще не обязательны. Проекция получается автоматически — в том числе, не только плоская, но и, например, цилиндрическая или сферическая. Это одно из проявлений универсальности метода трассировки.

В ходе трассировки лучей необходимо определять точки пересечения прямой линии луча с объектами. Способ определения точки пересечения зависит от того, кокой это объект, и каким образом он представлен в определенной графической системе. Так, например, для объектов, представленных в виде многогранников и полигональных сеток, можно использовать известные методы определения точки пересечения прямой и плоскости, рассмотренные в аналитической геометрии. Однако, если ставится задача определения пересечения луча с гранью, то необходимо еще, чтобы найденная точка пересечения лежала внутри контура грани.

Известно несколько способов проверки произвольной точки на принадлежность полигону. Рассмотрим две разновидности, в сущности, одного и того же метода (рис. 8.17).

Первый способ. Находятся все точки пересечения контура горизонталью, которая соответствует координате Y заданной точки. Точки пересечения сортируются по возрастанию значений координат X. Пары точек пересечения образуют отрезки. Если точка, которая проверяется, принадлежит одному из отрезков (для этого сравниваются координаты X заданной точки и концов отрезков), то она – внутренняя.

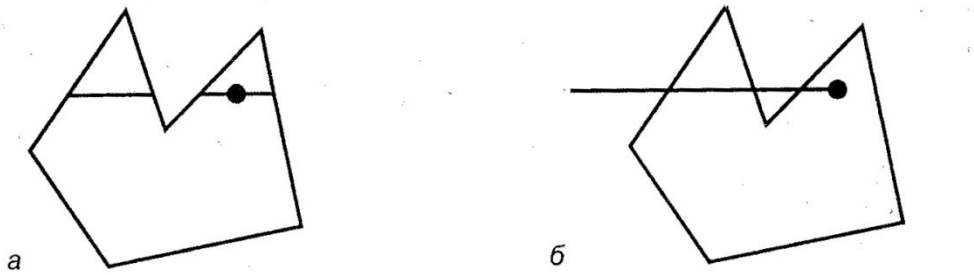


Рис. 8.17. Точка – внутренняя, если: а - точка принадлежит секущему отрезку, б – число пересечений нечетное

Второй способ. Определяется точка, лежащая на одной горизонтали с испытуемой точкой, причем требуется, чтобы она лежала вне контура полигона. Найденная внешняя точка и испытуемая являются концами горизонтального отрезка. Определяются точки пересечения данного отрезка с контуром полигона. Если количество пересечений нечетное, это значит, что испытуемая точка – внутренняя.

Если луч пересекает несколько объектов, то выбирается ближайшая точка по направлению текущего луча.

Сделаем общие выводы о относительно метода обратной трассировки лучей.

Положительные черты

1. Универсальность метода, его применимость для синтеза изображения довольно сложных пространственных схем. Воплощает много законов геометрической оптики. Просто реализуются разнообразные проекции.

2. Даже усеченные варианты данного метода позволяют получить довольно реалистичные изображения. Например если ограничится только первичными лучами(из точки проецирования), то это дает удаление невидимых точек. Трассировка уже одного – двух вторичных лучей дает тени, зеркальность, прозрачность.

3. Все преобразования координат (если таковые имеются) линейные, поэтому довольно просто работать с текстурами.

4. Для одного пиксела растрового изображения можно трассировать несколько близко расположенных лучей, а потом усреднять их цвет для устранения лестничного (ступенчатого) эффекта (антиалиасинг).

5. Поскольку расчет отдельной точки изображения выполняется независимо от других точек, то это может быть эффективно использовано при реализации данного метода в параллельных вычислительных системах, в которых лучи могут трассироваться одновременно.

Недостатки

1. Проблемы с моделированием диффузного отражения и преломления

2. Для каждой точки изображения необходимо выполнять много вычислительных операций. Трассировка лучей принадлежит к числу самых медленных алгоритмов синтеза изображений.

8.6 Анимация

В предыдущих параграфах мы рассмотрели методы и алгоритмы создания трехмерных моделей. В этом параграфе мы затронем вопросы, связанные с анимацией этих моделей.

Можно дать такое определение анимации — это создание зрительной иллюзии движения, изменения чего-то во времени.

Следует заметить, что в настоящее время существует некоторый разнобой в терминологии: наряду со словом "анимация" используется слово "видео". Значения этих слов несколько отличается. Термин "анимация" ранее использовался, когда говорили о "живых" ' куклах, а затем о персонажах мультфильмов. Само слово "видео" (со значением видеофильм) начало использоваться в русском языке с появлением видеомагнитофонов. Есть понятие "цифровое видео", которое часто используют для обозначения движущегося изображения, как правило фильма, сопровождающегося звуком. Таким образом, при создании видео (съемке фильма) какое-то непрерывное действие запечатлевалось на множестве фотографий (кадров). Кроме того, термин "видео" употребляется для обозначения изображений на экране, воспринимаемых органами зрения, в противопоставление звуковой информации: различают видеоданные и аудиоданные.

Эффект анимации основан на некоторых особенностях зрения человека, а именно: след изображения сохраняется некоторое время на сетчатке глаза свойственна способность объединять быстро сменяющие друг друга изображения в единый зрительный ряд, который даёт иллюзию непрерывности

Эти особенности зрения человека были использованы при создании игрушек тауматроп (1825 г.) и зоотроп (1834 г.). Тауматроп представлял собой плоский диск с рисунками, нанесенными на обе его стороны, а зоотроп — бумажную ленту с рисунками. При вращении этих игрушек возникала иллюзия движения.

В XX веке бурно развивалась мультипликация (традиционная анимация), средствами которой создавались рисованные двумерные фильмы.

Аналогично создаются движущиеся изображения на экране компьютера. Можно выделить два основных подхода к созданию компьютерной анимации: покадровый (каждый кадр с изображением движущегося объекта рисуется разработчиком) и по ключевым кадрам разработчик рисует опорные кадры, а программа рассчитывавает промежуточные кадры, помогая осуществить плавный переход - tween—из одного кадра в другой). Покадровый подход — очень трудоемкий, но дает более качественный результат. Анимация по ключевым кадрам значительно упрощает и ускоряет производство видеофильма, поэтому многие векторные редакторы имеют функцию Blend (перетекание), обеспечивающую такую анимацию. Например, при создании сложной анимации для Интернета часто используется векторный редактор Macromedia Flash. Анимация во Flash построена на использовании ключевых кадров. Поддерживаются тип motion tween (без изменения формы объекта) и тип shape tween (с изменением формы — с морфингом). Могут быть анимированы многие параметры, в частности, яркость, цвет, прозрачность. Объекты могут двигаться по прямой линии или по заданному пути (motion path), с ускорением или замедлением (easy in/easy out).

Методы компьютерной анимации

Известные к настоящему времени технологии компьютерной анимации можно разделить на два такие класса: 2D - и 3D - анимация. Несмотря на то, что результатом в обоих случаях является создание серии изображений в плоской проекции, методы 2D- и 3D-анимация существенно различаются. Под 2D-анимацией обычно подразумевается перемещение, наложение в определенном порядке отдельных спрайтов. Например, на фон накладываются изображения фигурок людей и животных. Для каждой движущейся

фигурки заготавливается несколько картинок, изображающих персонаж в различных фазах движения. Картинка может быть небольшим растром. Тогда для правильного наложения такого спрайта на фон в прямоугольнике растра пикселы за границами контура фигурки делаются прозрачными. До недавнего времени компьютерная 2D-анимация была очень популярной в разнообразных компьютерных играх. Для ее поддержки, как правило, все видеоадаптеры аппаратно выполняют соответствующие базовые функции, и в первую очередь, быстрое копирование прямоугольных фрагментов растра.

В настоящее время наблюдается переход на технологии 3D-анимации. Ее суть заключается в использовании трехмерных моделей объектов пространственных сцен и их отображение методами 3D-графики.

Следует отметить, что некоторые методы, разработанные ранее для 20-анимации, сейчас успешно реализуются и в системах 3D-анимации. Примером может служить следующий метод. Современные программы трехмерной анимации позволяют художнику-аниматору построить первоначальную сцену (ключевой кадр), передвинуть вперед указатель на временной шкале, изменить первоначальную сцену (следующий ключевой кадр*)* и получить построенные самим компьютером промежуточные кадры. Таким образом реализуется так называемый ***метод анимации по ключевым кадрам.*** Но идеи метода анимации по ключевым кадрам возникли и использовались еще при создании рисованных мультфильмов, когда ведущий аниматор определял, и сам делал ключевые кадры, а другие аниматоры рисовали остальные кадры. Аналогичный по идеям ***метод расчета промежуточных изображений & 3D-моделей*** - *tweening* - используется для уменьшения количество хранимых кадров (тогда при воспроизведении анимации на экране осуществляется интерполяция «на лету» положения вершин полигональной модели). Таким образом, одни и те же идеи и методы могут использовать и при традиционной, и при компьютерной **2D-** или 3D-анимации.

Рассмотрим вкратце остальные методы анимации.

Для создания ***иллюзии механического движения*** достаточно перемещать (поворачивать) одни модели относительно других моделей или относительно неподвижного фона (кстати, можно передвигать фон, оставляя модель неподвижной). Это методы, так сказать, простой анимации.

К методам простой анимации можно также отнести несложную деформацию (изменение масштаба всего предмета, неоднородное масштабирование, например, сжатие по одной оси и т. п.). Это также методы искажения объекта, связанные со смещением одной **(**нескольких) вершин объекта и распространением смещения по направлению к соседним вершинам. При этом величина смещения является заданной функцией расстояния.

Разработана также группа методов, связанных либо с *деформацией двумерной сетки,* на которую помещен объект, либо *с глобальной деформацией пространства,* в котором задан объект. Часто используются различные модификации *метода свободной деформации (FFD — free-form deformation),* являющегося трехмерным расширением метода деформации двумерной сетки. Общая идея этих методов основана на том, что художнику-аниматору проще оперировать системой локальных координат, в которую помещен искажаемый объект, чем вершинами этого объекта. Поэтому, после определенной художником-аниматором деформации локальной системы координат, производится пересчет координат вершин искажаемого объекта в глобальное пространство.

В современных программах, реализующих механическое движение, как правило, создают средства динамики, обеспечивающие анимацию объектов в соответствий с законами физики. Например, при расчете траекторий движений объектов учитываются влияния трения, гравитации, магнетизма и др. Художник-аниматор, работая с такими программами, может даже не создавать ключевые кадры — он задает начальное положение, физические свойства и законы взаимодействия объектов ит. п.

Наряду с методом анимации по ключевым кадрам, упомянутым выше, применяют и более изощренный ***метод анимации на основе событий*** (который не исключает и применения ключевых кадров). При этом методе "событиям и считаются изменения в состоянии того или иного параметра. В качестве параметров выступают предусмотренные конкретной анимационной программой элементы сцены (формы объектов, текстуры, параметры источников света, координаты камеры и т. д.). Для каждого параметра на временной шкале выделяется отдельная дорожка, что позволяет перемещать события вдоль временной шкалы или подвергать иным преобразованиям независимо друг от друга. При этом траектория объектов может быть нелинейной.

Для передачи движения "живых" объектов, которое характеризуется взаимосвязанным плавным перемещением нескольких (или всех) элементов объекта, возникает необходимость в сложной деформации модели. Например, если движется рука, то, как правило, слегка изменяются очертания плеча. При так называемой "лицевой анимации" необходимо менять взаиморасположение нескольких элементов объектов. Такой объект должен описываться единой моделью для того, чтобы при движении отдельных его частей, например, руки, не создавались грубые переходы, стыки.

Один из методов более совершенной анимации — ***метод вершинной (вертексной) анимации объектов*** — связан с представлением объекта как цельной полигональной модели (еще говорят, что он должен представлять собой одну сетку). Тогда, например, лицевая анимация выполняется путем передвижения по заданной траектории определенных вершин полигональной модели, в то время как положение остальных вершин не меняется, тем самым осуществляется деформация полигональной модели (деформация сетки объекта). Таким образом создается ряд трехмерных моделей, отображающих последовательность движения "живого" объекта в различные моменты времени. Этот ряд последовательно выводится на экран (естественно, уже с текстурами, светом и т. п.), создавая иллюзию движения. Однако при таком методе анимации приходится хранить положение каждой вершины полигональной модели в каждом кадре анимации. То, что анимации моделей хранятся как бы по трехмерным кадрам, приводит к росту размера файлов пропорционально количеству кадров.

Разумеется, существуют методы, позволяющие снизить объемы хранимой информации в том числе метод расчета промежуточных изображений (tweening) между изменениями объекта. В этих случаях для задания опорного кадра анимации используются наборы сеток, а между опорными кадрами положение точек и ориентация нормалей интерполируется необходимым образом (чаще всего линейно).

Следует заметить, что несмотря на преимущественное применение скелетной анимации (описана ниже), вершинная анимация продолжает использоваться по определенным соображениям (например, при желании уменьшить временные затраты или упростить код).

В качестве примера метода осуществления сложной деформации модели приведем *морфинг* (термин происходит от слова ***metamorphosing*** — проведение преобразования). Морфинг заключается в последовательном превращении одного объекта в другой посредством перемещении по определенной траектории заданных точек (линий) одного объекта в соответствующие точки (линии) другого объекта в сочетании с наложением этих двух объектов. Хотя он представляет собой двумерный анимационный метод, некоторые его модификации успешно применяется наряду с трехмерным методом скелетной анимации, например, для реалистической анимации лица. Кроме того, проводятся довольно успешные исследования по осуществлению морфинга трехмерных объектов.

Существует два подхода к осуществлению морфинга — подход, основанный на использовании криволинейной координатной сетки, и подход, основанный на установлении соответствия между изображениями при помощи линий особенности. Так или иначе, при использовании морфинга приходится решать на интуитивном уровне, какие из областей

изображения одного объекта и с какой скоростью преобразовывать в области изображения другого объекта.

Метод взвешенно-целевого морфинга достаточно успешно применяется для реалистичной анимации лица. Такое название метод получил вследствие особенности обеспечения требуемого выражения лица. В соответствии с этим методом вначале подготавливаются ключевые состояния лица: улыбка, широко раскрытые глаза, насупленные брови и т. д. Затем указываются "весовые" доли для каждого из этих ключевых состояний, тем самым создается требуемое выражение лица.

Одним из наиболее эффективных и часто применяемых является ***метод скелетной анимации,*** при котором перемещение вершин полигональной модели осуществляется с помощью невидимых анимированных "костей" *(bones),* составляющих иерархическую структуру - "скелет" *(skeleton).* Для каждой кости задаются длина и некоторые параметры, характеризующие ее положение.

Рассмотрим следующий пример (рис. 8.18).

Чтобы имитировать движения рыбки нужно как-то деформировать соответствующую полигональную сетку. Движения хвоста можно осуществлять простейшим способом — сдвигать координаты всех вершин пропорционально расстоянию вершины от головы. Но так невозможно имитировать поворот хвостового плавника на угол 90 градусов и более. Можно наряду с глобальным сдвигом применять локальные повороты групп вершин. Движения губ и плавников можно было бы выполнить как перемещения отдельных вершин.

Однако описание соответствия указанных преобразований сетки и движений рыбки здесь представляется достаточно сложным и запутанным. Как представляется, здесь лучше использовать деформацию поверхности на основе скелетной модели. Для построения изображения рис. 8.18 мы использовали один из простейших вариантов скелета.

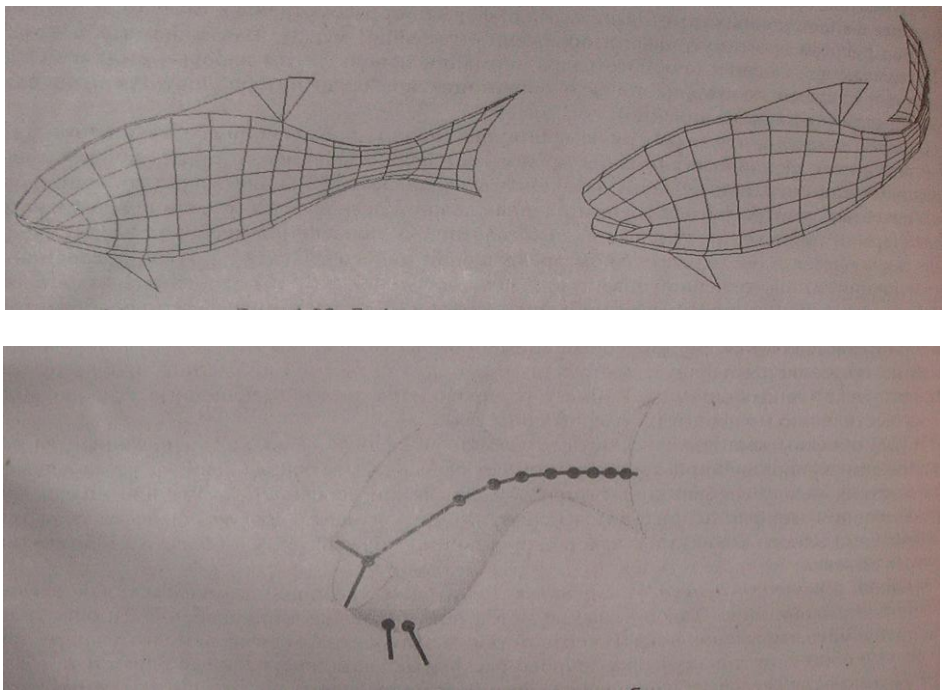


Рис. 8.18. Пример деформируемой полигональной сетки

Сложность скелета определяется требуемым уровнем детализации изображаемого объекта. Например, при изображении человека, шагающего где-то вдали, достаточно показать основные движения рук и ног, в то время как для крупных планов, возможно, потребуется показать движения отдельных пальцев (рис. 8.19).

Скелет состоит из костей (звеньев) и сочленений (на рис. 8.19 сочленения изображены кружочками). Каждая *i*-я кость описывается такими параметрами: длиной (l_i) и поворотами относительно родительской кости. Если поворот возможен только в одной плоскости, то говорят, что такое сочленение имеет одну вращательную степень свободы. Если повороты могут осуществляться в двух или трёх плоскостях, то это называют двумя или тремя степенями свободы. Заметим, что в теории манипуляторов кроме вращательных движений движений элементов шарнирных конструкций предусматривают также возможность изменения длины звена - это еще одна степень свободы. В качестве базового элемента для расчётов удобно использовать звено только с одной вращательной степенью свободы. В этом случае сочленение с несколькими вращательными степенями свободы представляется из нескольких сочленений звеньев нулевой длины. Таким образом, для описания каждой кости будем использовать один угол поворота (α_i).

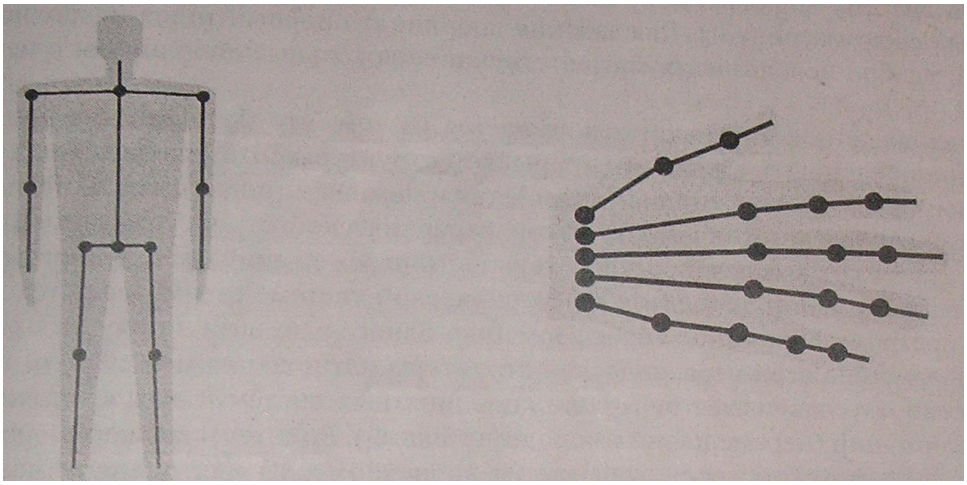


Рис. 8.19. Скелет человека для разных уровней детализации

Скелет имеет древовидную иерархическую структуру — с родительской костью соединяется одна или несколько костей, которые, в свою очередь могут являться родительскими для других соединенных с ними костями. Рассмотрим конструкцию из двух костей (рис. 8.20).

Зафиксируем систему трехмерных координат (x_0, y_0, z_0) в начале родительской кости (ось y_0 смотрит на нас). Угол поворота родительской кости (α_1) здесь отсчитывается от вертикали (хотя это не принципиально).

Найдем координаты произвольной точки *P,* связанной с концом второй кости:

$$P = R_1 \text{ x } T_1 \text{ x } R_2 \text{ x } T_2 \text{ x } P_2,$$

где P_2 — это координаты искомой точки, заданные в локальной системе координат (x_2, y_2, z_2), центр которой располагается в конце второй кости, R_1 и R_2 — матрицы поворотов на углы α_1 и α_2, T_1 и T_2 — матрицы сдвига вдоль оси *z* на длину костей.

Обобщим эту формулу для шарнирного соединения *n* костей.

$$P = M_n * P_n,$$

где P_n — это координаты искомой точки в локальной системе координат, связанной с концом последней кости шарнира, M_n — матрица преобразований координат. Эту матрицу удобно вычислять рекурсивно:

$$M_i = M_{i-1} \text{ x } R_i \text{ x } T_i,$$

где $i = 1,2, \ldots, n$, причем

$$M_1 = R_1 \text{ x } T_1.$$

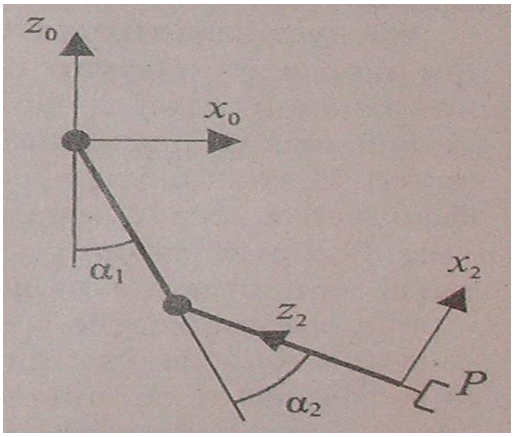


Рис. 8.20. Взаимное расположение двух костей.

Если требуется вычислить мировые координаты всех точек поверхности (опорных точек полигональных сеток) тела движущегося объекта, то необходимо вначале определить матрицу преобразований из локальной системы, связанной с начальной точкой всех родительских костей, в систему мировых координат. Затем произвести обход скелета, для каждой кости вычисляя свою матрицу M_i. Для задания координат опорных точек поверхности тела вокруг костей, удобно использовать соответствующие локальные координаты с центрами в концах костей.

Идея метода скелетной анимации основывается на том, что большинство движущихся объектов можно представить в виде иерархической структуры. Это относится как к живым объектам (людям, животным, растениям и т.п.), так и неживым (планетные системы, роботы и т. п.). Перемещение одного элемента такой иерархической структуры относительно другого элемента может вызвать последовательность перемещений ряда элементов. Другими словами, трансформации элементов в иерархической системе взаимозависимы, поэтому можно задать программе анимации трансформацию одного элемента и получить рассчитанную ею трансформацию всей иерархической структуры. Отношения между родительской и дочерней костями обуславливают иерархию координатных систем и соответствующую иерархию трансформаций (перемещений и/или деформаций). Родительская кость, для которой не существует родительской кости, называется корневой (root) или предком всех других Костей, ее локальная система координат является абсолютной. В частности, для скелетной модели человека в качестве корневой кости часто выбирают тазовую кость.

Следует заметить, что в целях экономии времени расчета и объема памяти при анимации иногда используют простейший вариант использования скелета — каждая вершина полигональной сетки поверхности связывается только с одной костью. Однако лучшее качество, большая реалистичность достигаются при учете влияния на одну вершину нескольких костей. В этом случае для вершины задают степень влияния каждой кости с помощью коэффициента веса, сумма которых для одной вершины, как правило, должна быть равна единице. В соответствии со значениями этих коэффициентов выполняется интерполяция координат вершин полигонов (метод интерполяции вершин — ***vertex blending).***

Как подмножество метода интерполяции вершин может рассматриваться ***vertex skinning*** — метод трансформации вершин геометрической сетки в местах сгиба модели. Применяется метод ***vertex skinning*** для того, чтобы на стыках текстур (особенно на сгибах и сочленениях — это, например, все суставы модели человека или животного) были плавные, естественные переходы, особенно во время движения. Таким образом, при помощи этого метода осуществляется правильное расположение вершин. Для этого используются матрицы, называемые *skinning matrices.* Текстуры же на корректно трансформированную геометрию натягиваются правильно автоматически. Основная идея этого метода — интерполировать результаты матричного преобразования, используя веса,

основанные на начальном местоположении каждой вершины. Это позволяет отдельному треугольнику сетки деформироваться (натягиваться подобно коже) и сохранять связь, например, с суставами, поскольку каждой вершине приписывают различный вес.

Таким образом, вычисление координат некоторой вершины *V* интерполяцией в соответствии с методом *vertex blending* можно описать так:

$$V = V_1 \times k + V_2 \times (1-k),$$

где V_1 , V_2 — используемые вершины, а коэффициент k принимает значение в диапазоне от 0 до 1.

Метод vertex skinning - это vertex blending для вершин, обработанных разными матрицами (M_1,M_2):

$$V = V_1 \times M_1 \times k + V_2 \times M_2 \times (1-k).$$

Фирма ATI начала продвигать возможности “матричного наложения” (“*matrix skinning”*) в видеоадаптерах RADEON. Эта технология получила высокую оценку разработчиков, т.к. она позволяет очень реалистично "растягивать" кожу персонажей и отображать их суставы

Итак, подытожим. Для осуществления скелетной анимации какого-нибудь персонажа необходимо, прежде всего, создать его полигональную модель (сетку персонажа). Деформация сетки персонажа осуществляется под влиянием скелета. Он создается и подгоняется под размеры и форму тела персонажа, затем каждая вершина сетки связывается с одной или несколькими костями скелета. Если какая-то кость подверглась трансформации, то необходимо выполнить расчет скелета, вследствие чего осуществляется деформация сетки. Иными словами, при анимации изменяются параметры положения частей скелета, а сетка лишь следует за ними.

Основные достоинства метода скелетной анимации заключаются в следующем:

- в меньшем размере файлов, в которых хранится анимация, так как в них нет необходимости хранить положения всех вершин для всех кадров;
- в лучшем качестве изображения по сравнению с вершинной анимацией при том же (и даже меньшем) количестве полигонов;
- в упрощении задач художника-аниматора, так как ему не нужно самому отслеживать трансформацию по всей иерархии костей.

Одной из проблем реализации метода скелетной анимации является управление движением костей скелета. Нужно как-то описывать последовательность движений костей, соответствующих требуемым движениям персонажа. Пусть, например, у персонажа таз — это корневое звено, для которого туловище и ноги являются дочерними звеньями. В свою очередь, ноги являются родительскими звеньями по отношению к ступням.

В соответствии с ***методом прямой кинематики (Forward Kinematics)*** управляя углами поворота костей скелета, добиваются требуемых поз. Процесс подбора углов можно представить следующим образом. Движение родительского звена (например, ноги) автоматически приводит в движение всю цепь дочерних звеньев (в данном случае ступню), причем дочерние звенья будут перемещаться, не изменяя своего положения относительно объекта-предка. Если родительское, звено поворачивается, то дочернее соответственно и перемещается, и поворачивается, чтобы его ориентация по отношению к родительскому звену осталось прежней. Этот метод несколько утомителен для художника-аниматора, так как требует указывать множество углов в сочленениях.

При использовании ***метода обратной (инверсной) кинематики (Inverse Kinematics)*** исходными являются позы, а точнее, координаты концевых точек звеньев скелета. Исходя из этих координат, находятся соответствующие углы поворота всех костей. Движение задается перемещением самого младшего дочернего звена (в нашем

случае ступни), что заставляет всю остальную цепочку (ногу, туловище, таз и т.д.) перемещаться. Как правило, расчет перемещений осуществляется с учетом ограничений на работу сочленений звеньев: например, вводятся приоритеты сочленений, их фиксация, угловые ограничения и трение в узлах сочленений и т.п. При этом метод обратной кинематики, в отличии от метода прямой кинематики, может дать несколько вариантов решения (или странные и непредвиденные решения) — это зависит от количества звеньев и ограничений. На рис. 8.21 показано, что даже для простейшего скелета из двух костей могут быть два варианта их углов поворота.

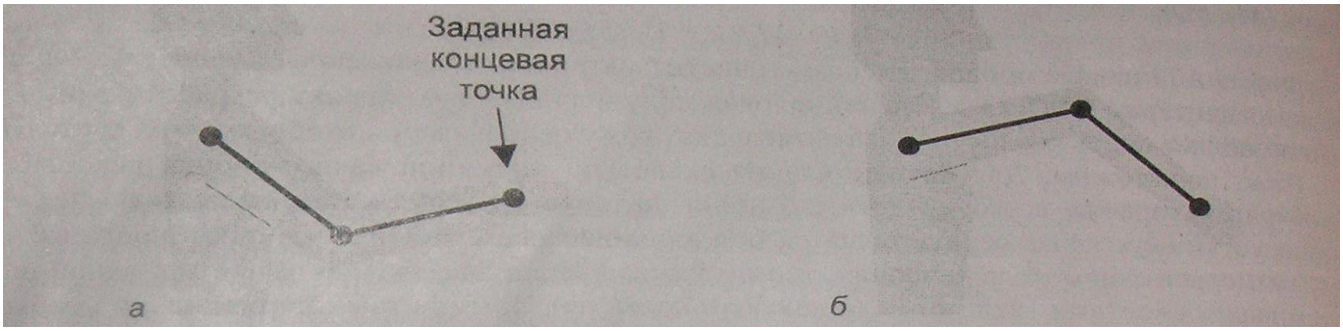


Рис. 8.21. Решение задачи обратной кинематики: а – правильное, б - неправильное

Задача обратной кинематики для простых скелетов с небольшим количеством звеньев может решаться аналитически непосредственно. Для расчета сложных скелетных конструкции применяются математический аппарат *якобианов* и соответствующие итерационные методы.

Следует отметить, что при использовании, например, метода обратной кинематики расчет скелета может не производиться заранее, а вычисляться процедурно. Так называемый ***метод процедурной анимации*** применяется в тех случаях, когда моделирования движений трудно (неэффективно) воспроизводить **с** помощью ключевых кадров. При процедурной анимации текущие значения параметров анимации рассчитываются на основе заданных начальных значений и математических выражений, описывающих изменение параметров во времени. Процедурная анимация часто используется для качественной анимации разнообразных физических эффектов.

Аналогично применяется *параметрическая анимация*. В роли параметра может выступать любой объект — кривая, поверхность, точка, систем координат и т.д. Например, частоту или скорость движения объекта можно задать графиком, а затем анимировать этот график, изменяя тем самым параметры движения объекта.

Упомянем также о таких средствах анимации как редактирование кривых действия и системы частиц (particle systems).

Движения, особенно "живых" объектов в реальном мире редко бывает линейным: например, объекты могут плавно замедлять ход, а затем резко останавливаться. Для подготовки подобных эффектов используются графические средства *редактирования кривых действия.*

При использовании ***метода моделирования частиц*** (particles) создается набор частиц (часто в качестве частиц используются точки, то есть объекты, не имеющие размеров). Для частиц могут быть заданы законы их существования, например, основанные на каких-то реальных физических законах, а именно, законах притяжения под действием силы тяжести, электростатических, магнитных сил, и т.п.

Например, система частиц в разработанной компанией RealSoft OY системе 3D-графики RealSoft 4D может работать с частицами следующих типов: 1D, 2D, 3D, NURBS-кривые (NURBS — Non Uniform Rational B-Spline - неоднородный рациональный Би-сплайн), причем любой объект сцены может быть представлен как частица. Частицы

рассчитываются как брызги воды, туман, огонь, они могут использоваться в качестве параметров для процедурной обработки. Любая поверхность может быть источником частиц, и любой объект может быть ими заполнен. При движении частицы могут быть заданы начальные параметры ее движения, действующие на нее силы, время жизни, цветовые параметры и т.п. Частицы могут быть текстурированы, все атрибуты частиц (например, толщина, длина или прозрачность) доступны для анимации. Комбинируя все эти возможности моделирования, художник может создать модели персонажей, декорации и окружающую среду.

Кроме того, в состав RealSoft 4D входит Envelope System — редактор графов, с помощью которого можно управлять поведением любого атрибута объекта в виде функции другого атрибута объекта сцены. Например, можно задать закон изменения угла скелетона в зависимости от гравитационного поля. Графы представлены как NURBS-кривые в 4D- пространстве, что позволяет получать очень мягкие движения при анимации

Методы обратной кинематики и анимации по ключевым кадрам используются совместно с методом канальной анимации. ***Метод канальной анимации (channel animation)*** основан на снятии информации о каком-либо параметру объекта с датчика (канала). Например, для снятия информации о движении актера датчики крепятся по всему его телу в тех местах, которые будут приведены в соответствие с контрольными точками компьютерной модели для ввода и оцифровки движения, а приемники информации подключены к компьютеру. Датчики могут быть разных видов, например, электромеханическими, электромагнитными (беспроводными или соединяющимися с компьютером проводами) или оптико-электронными, информацию с которых считывают специальные оптические устройства, подключенные к компьютеру. Следует заметить, что в настоящее время беспроводные датчики используются реже, так как для снабжения их энергией актеру приходится носить на себе аккумулятор. Оцифрованные движения реального человека служат для создания моделей, изображающих компьютерный персонаж.

На этом методе основано относительно новое направление в анимации — ***технология real time performance animation,*** основанная ***на захвате (видеозахвате) движения (Motion Capture)***, который дает возможность передавать естественные, реалистичные движения в реальном времени. Для захвата движений часто используют пассивные оптические метки и видеотехнологию для записи движений объекта. В этом случае актеру приходится носить только отражающие свет метки, закрепленные на одежде. Естественно, что качество синтезированного движения напрямую зависит от количества и расположения датчиков.

Таким образом могут создаваться данные для библиотеки движений, в которых содержится информация о движении тех или иных частей тела человека, животного при конкретных действиях. Использование библиотеки значительно упрощает и удешевляет создание анимации.

Глава 9. Архитектуры графических систем

Графические рабочие станции стали результатом сбалансированного объединения технологий:

- построения процессоров,
- организации связи,
- организации ввода/вывода,
- работы с графическими объектами и устройствами.

Графические рабочие станции построены, в основном из стандартных компонент. Операционные системы, как правило, Unix, реже - Windows. Следуют концепции открытых систем. Поддерживают 3D графику. В период с1980 по 2005 г. осуществлялось удвоение производительности за 2 года.

9.1 Суперстанции

Обычно проектируются по принципу объединения в одной системе возможностей рабочих станций (3D графика, интегрированность) и суперкомпьютеров (быстрый ввод/вывод, распараллеливание или векторизация вычислений).

Состав типичной суперстанции:

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

9.2 Компоненты растровых дисплейных систем

Рис. 9.1. Растровая графическая система

- графический процессор – отвечает за генерацию примитивов и BitBlt операции;
- видеопамять - за хранение изображения;
- видеоконтроллер – за отображение кадрового буфера, управление раскраской, некоторыми атрибутами и монитором.

9.3 Подходы к проектированию графических систем

1) максимум простоты, минимум гибкости - функционально завершенный графический сопроцессор, аппаратно реализующий основные графические примитивы. Изменения алгоритмов сопроцессора невозможны.

Недостающие графические средства реализуются за счет доступа к видеопамяти из программы пользователя (Intel, i82786);

2) гибкость на этапе проектирования - построение графической системы из набора сверх больших интегральных схем (СБИС), реализующих строго регламентированные функции (National Smiconductor, набор СБИС DP 85xx);

3) максимум гибкости - использование универсального процессора, программируемого на языке высокого уровня и аппаратно реализующего основные растровые операции (Texas Instruments, TMS 34010, TMS 34020 и Intel, i860).

9.4 Графические системы на базе сопроцессора i82786

Рис. 9.2. Графическая система на базе i82786

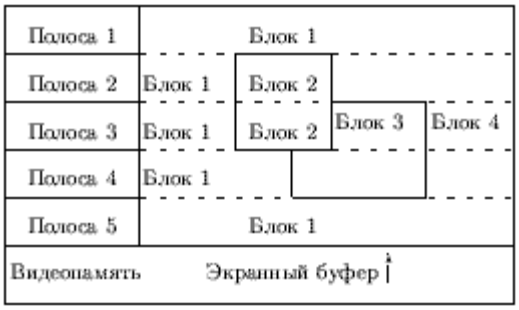
Имеют следующие характеристики:

- неавтономная работа под управлением ЦП на базе микропроцессора Intel, который передает команды и данные, отрабатываемые сопроцессором.
- 22-разрядная шина адреса - до 4 М видеопамяти,
- 16-разрядная шина данных,
- системное тактирование - 10 МГц,
- видеотактирование - 25 МГц,
- построение символов - 25 млн. символов/с,
- разрешающая способность:
  от 640×480×8 до 1024×1024×2,
- программирование сигналов монитора - до 4096×4096

Используют следующий набор команд:

- общее управление,
- управление курсором,
- управление атрибутами примитивов,
- генерация примитивов,
- блочная перепись и управление окнами.

ЦП в видеопамяти формирует таблицу, описывающую конфигурацию экранного буфера. Экранный буфер может быть разбит на полосы. Каждая строка может быть разбита на окна-сегменты. Минимальная ширина окна - 1/16 длины строки.



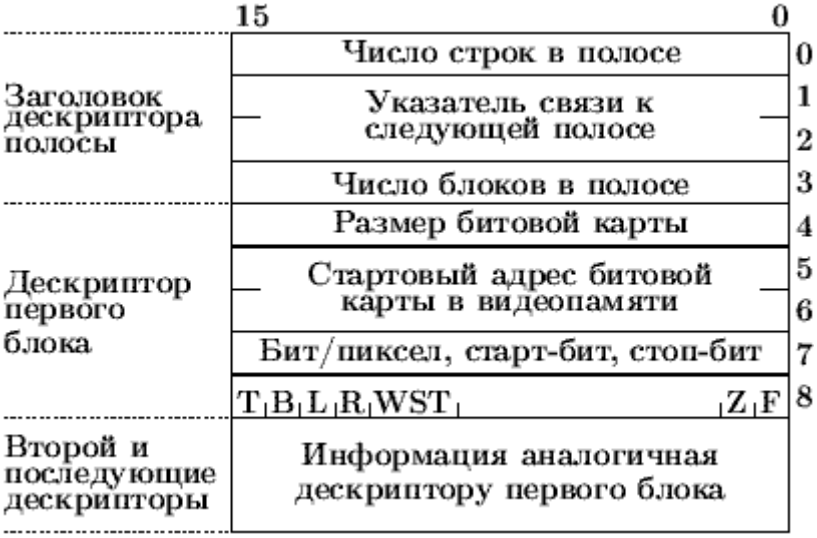


Рис. 9.3. Таблица конфигурации экранного буфера

9.5 Графические системы из набора сверх больших интегральных схем (СБИС)

В таких системах разработчик сам определяет требуемые функции и реализует их, используя те или иные компоненты набора. Например, набор СБИС AGCS 85xx (Advanced Graphics Chip Set) фирмы National Semiconductor состоит из:

- DP-8500 - растровый графический процессор (RGP - Raster Graphics Processor),
- DP-8510 процессор обмена блоками информации (BPU - BitBlt Processing Unit),
- DP-8512 - генератор тактовых импульсов для вывода видеоизображения (VCG - Video Clock Generator),
- DP-8515 - высокоскоростной сдвиговый регистр (VSR - Video Shift Register).

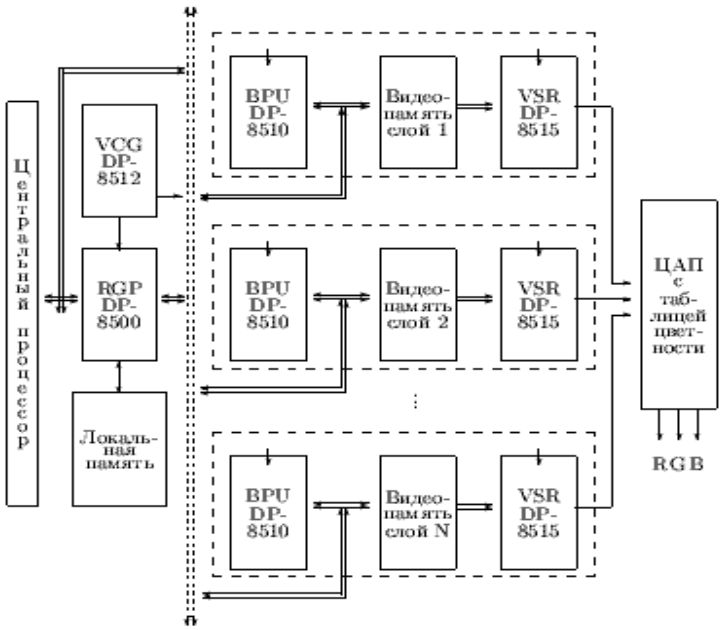


Рис. 9.4. Графическая система на базе AGCS 85xx

9.6 Растровый графический процессор DP-8500

Рис. 9.5. Графический процессор DP-8500

Имеет следующие характеристики:

- тактирование 20 МГц,
- цикл шины 100 нс,
- производительность графической системы 10÷160 Мпикселов/с,
- адресное АЛУ, 28 разрядов, 16 регистров,
- АЛУ данных, 16 разрядов, 16 регистров,
- аппаратная генерация линий,
- аппаратное отсечение,
- аппаратное копирование блоков бит,

• видеопамять послойная, в глубину, смешанная,
• получение дисплейного файла от ЦП по флагу.

9.7 Графические системы на универсальном процессоре

Одни из самых специфичных графических систем. Расширение числа аппаратно реализованных функций мало приемлемо по следующим причинам:

1. Набор графических функций был бы жестко зафиксирован. Новые примитивы или старые, но с расширенными возможностями, не смогут быть поддержаны.

2. Аппаратная реализация означает "жесткий" выбор поддерживаемых атрибутов, (ТИП ЛИНИИ, ШИРИНА ЛИНИИ, ЦВЕТ ЛИНИИ, ПРОЗРАЧНОСТЬ и т.п.) т.е. некоторые редкие, но существенные для отдельных применений атрибуты будут опущены, например, ФОРМА КОНЦОВ ЛИНИИ (endpoint shape).

3. Высококачественная графика требует точного контроля над алгоритмами формирования изображений, например, для устранения ступенчатости. При аппаратной реализации потребуются дополнительные параметры. В программной реализации сглаживание выполняется при необходимости.

4. Формат дисплейного списка, или команд формирования изображений может варьироваться в соответствии с требованиями пользователя. (Например, отображение простых и/или высококачественных шрифтов.)

Единственный способ гибкого удовлетворения требований - программирование функций процессора, интерпретирующего графические команды.

RISC процессор с графическим устройством i860

Основные технические характеристики:

• тактирование 50 МГц;
• пиковое быстродействие - 40 MIPS и 80 MFLOPS;
• процессор для целых - управление системой и операции над 8, 16 и 32-х битовыми целыми;
• векторный и скалярный режимы для вещественных;
• графическое устройство обрабатывает до нескольких пикселов одновременно в 64-х битном слове;
• глубина пиксела - 8/16/32 бита;
• имеется режим поддержки 3D отображения;
• аппаратная поддержка сравнения в Z-буфере;
• закраска Гуро - 50000 треугольников/с;
• до 500 000 однородных преобразований/с.

Рис. 9.6 . RISC процессор

9.8 Высокоскоростные графические системы

Кроме высокоскоростной генерации и манипулирования растровыми образами для формирования высокореалистичных картин в реальном времени, в подобных системах требуются сбалансированные по времени моделирование сцен, выполнение геометрических расчетов, расчет освещенностей.

Подходы для быстрого вычисления изображений:

- использование специализированной аппаратуры (Silicon Graphics);
- использование универсального вычислителя, дополненного средствами быстрого отображения (Stardent);

Рабочие (супер)станции Silicon Graphics

Джеймс Кларк (James Clark) в 1981 г. разработал Геометрическую машину, ориентированную на моделирование 3D сцен. Она послужила прототипом этих рабочих станций. В 1982 г. основана фирма Silicon Graphics, и в 1984 г. произошел выход на рынок рабочих станций на базе машины Дж. Кларка. С 1988 г. в рабочих станциях SG используются только RISC-процессоры фирмы MIPS.

Рис. 9.7 . Структура станции Silicon Graphics

В системе осуществляется более 10^6 Z-буферизованных треугольников в сек. Скорость доступа к кадровому буферу - более $20 \cdot 10^6$ пикс./с.

Для каждого пиксела хранятся:

- R, G, B и Альфа-каналы по 8 бит,
- Z-координата,
- текстурные планы для R, G, B и Альфа-каналов,
- биты управления перекрытиями окон,
- биты задания способа отображения (полноцветное или индексное изображение, одно/двухкратная буферизация).

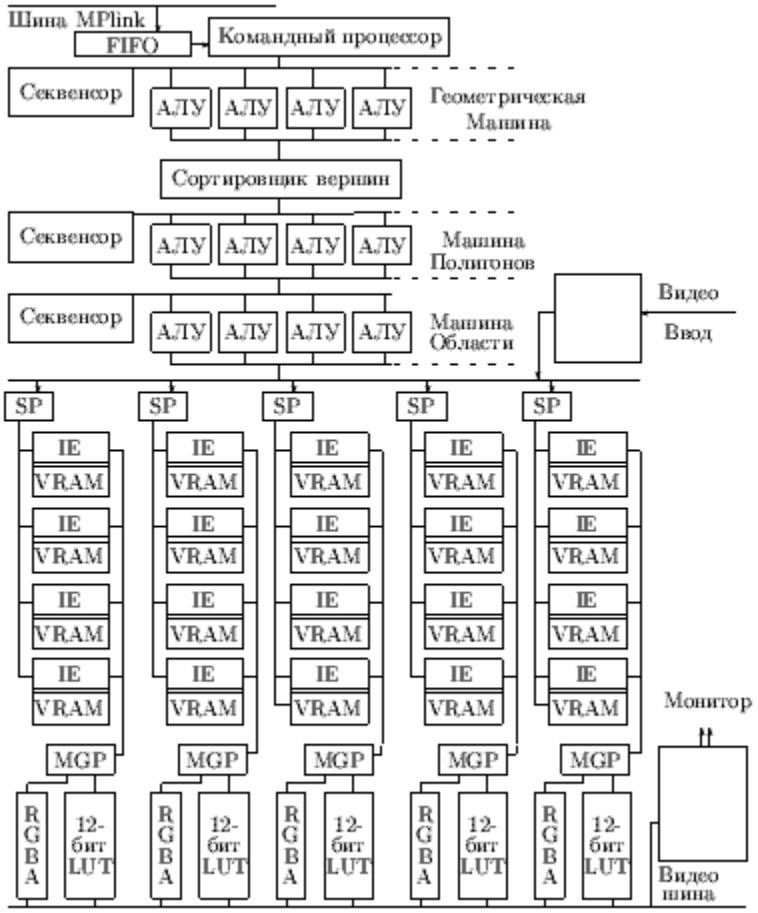


Рис. 9.8. Структура RISC-процессора фирмы MIPS.

9.9 Рабочие (супер)станции с использованием универсального вычислителя

Многие вычисления при формировании изображений близки к обработке, требуемой в научных вычислениях (геометрические преобразования, расчеты освещенностей и т.д.).

Сочетание высокоскоростного вычислителя и средств быстрого отображения позволяет построить систему, пригодную и для вычислительно интенсивных заданий и для графики реального времени.

Такой подход был реализован в системе GS2000 фирмы Stardent.

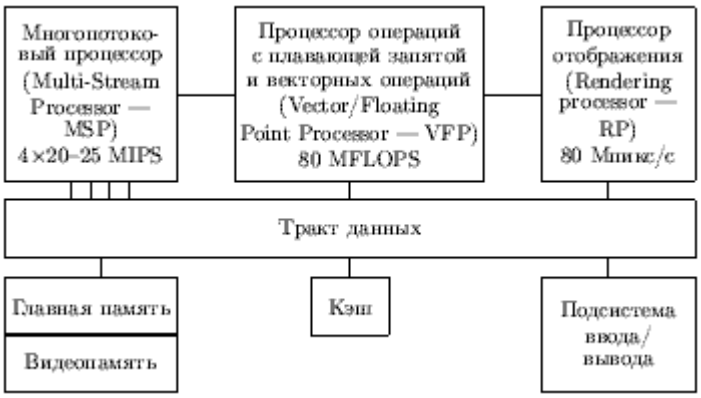


Рис. 9.9. Структура рабочей станции с использованием универсального вычислителя

Процессор VFP выполняет координатные преобразования (600 K 3D преобразований в сек), отсечение и вычисление освещенности.

Процессор RP ведет обработку изображений со скоростью 80 Мегапикселов/с. (600 K 3D клиппируемых векторов/с, длиной 10 пикселов и 160 K 100 пиксельных треугольников Гуро с Z-буферизацией).

Глава 10. Стандартизация в компьютерной графике

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

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

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

10.1 NGP (Network graphics protocol)

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

Модель работы пользователя в сети с применением графического протокола приведена на рис. 10.1

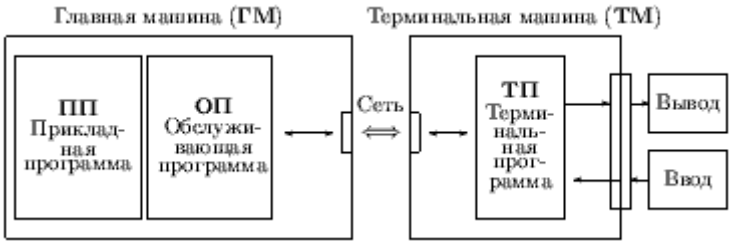


Рис. 10.1. Модель организации работы в сети

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

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

Данные в сети передаются только в стандартной форме, следовательно, на передающей стороне выполняется кодирование, а на приемной - декодирование информации. Предусмотрено два уровня протокола вывода: сегментированный и структурированный форматы. В сегментированном формате изображение строится из отдельных сегментов, представляющих собой список графических примитивов (точек, линий, строк текста). ТП выполняет только перекодировку и может быть достаточно простой. В структурированном формате изображение строится из вызовов отдельных подкартин, состоящих из примитивов и вызовов других подкартин. Объем передаваемых данных уменьшается, но на ТМ, как правило, требуется программное выполнение преобразований.

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

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

После публикации появился целый ряд работ, посвященных использованию идей протокола в нашей стране.

10.2 Международная деятельность по стандартизации в машинной графике

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

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

Структура прикладной графической системы, удовлетворяющей сформулированным требованиям, может быть представлена в виде шестиуровневой модели (рис. 10.2).

Рис. 10.2. Уровневая модель прикладной графической системы

Процесс преобразования информации при выполнении вывода может быть представлен состоящим из следующих этапов (рис. 10.3):

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

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

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

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

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

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

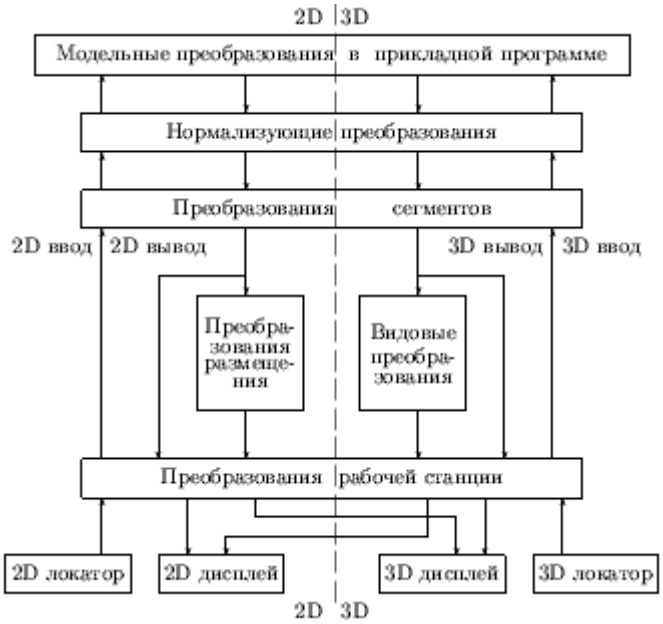


Рис.10.3. Схема преобразований координатной информации в графической системе

Концептуальная модель переносимой графической системы показана на рис. 10.4. Штриховые линии на нем обозначают интерфейсы, при стандартизации которых может быть обеспечена переносимость.

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

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

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

Рис. 10.4. Архитектура переносимой графической системы

Деятельность ISO, IEC по стандартизации в машинной графике

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

1. GKS (Graphical Kernel System) - набор базовых функций для 2D аппаратно-независимой машинной графики.
2. GKS-3D (Graphical Kernel System for 3 Dimensions) - расширение GKS для поддержки базовых функций в 3D.
3. PHIGS (Programmer's Hierarchical Interactive Graphics System) - набор базовых функций 3D графики аналогичный GKS-3D, но в отличие от GKS-3D, ориентированной на непосредственный вывод графических примитивов, группируемых в сегменты, графическая информация накапливается в иерархической структуре данных. В целом PHIGS ориентирован на приложения, требующие быстрой модификации графических данных, описывающих геометрию объектов.
4. Языковые интерфейсы (Language bindings) - представление функций и типов данных функциональных графических стандартов в стандартизованных языках программирования.
5. CGM (Computer Graphics Metafile) - аппаратно-независимый формат обмена графической информацией. Используется для передачи и запоминания информации, описывающей изображения.
6. CGI (Computer Graphics Interafce) - набор базовых элементов для управления и обмена данными между аппаратно-независимым и аппаратно-зависимым уровнями графической системы.
7. CGRM (Computer Graphics Reference Model) - модель стандартов в машинной графике, которая определяет концепции и взаимосоотношения применительно к будущим стандартам в машинной графике.
8. Регистрация - механизм регистрации стандартизуемых аспектов примитивов вывода, обобщенных примитивов, escape-функций (для доступа к аппаратным возможностям устройств) и других графических элементов.
9. Тестирование реализаций на соответствие графическим стандартам - основные цели этого проекта: специфицирование характеристик стандартизованных тестов, используемых для определения соответствия реализаций графическим стандартам, и выработка предписаний разработчикам функциональных стандартов относительно правил соответствия.

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

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

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

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

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

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

10.3 Классификация стандартов

Из рис. 10.4 видно, что для обеспечения мобильности программного обеспечения требуется стандартизовать:

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

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

Core-System

Существенным этапом в области стандартизации машинной графики явилась публикация проекта стандарта CORE-SYSTEM (GSPC-77) , модель которой приведена на рис. 10.5. Главные идеи, положенные в основу системы CORE : разделение функций ввода и вывода; минимизация отличий между выводом на графопостроитель и интерактивный дисплей; концепция двух координатных систем - мировой системы координат, в которой конструируется выдаваемое изображение, и приборной системы координат, в которой представляются данные для отображения; концепция дисплейного файла, содержащего приборную координатную информацию; понятие дисплейного файла сегментов, каждый из которых может независимо модифицироваться как элемент; обеспечение функций преобразования данных из мировой системы координат в приборную путем вызова видового преобразования.

В системе выделены следующие группы функций: вывода; сегментирования дисплейного файла; установления и опроса атрибутов примитивов (цвет, яркость, ширина линии и т.д.) и атрибутов сегментов (тип, видимость, указуемость и т.д.); визуализации; выполнения ввода с виртуальных устройств ввода типа указка, клавиатура, кнопка, локатор, датчик; управления и доступа к специальным аппаратным возможностям.

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

Рис. 10.5. Модель графической системы, положенная в основу CORE-SYSTEM

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

Группа уровней вывода включает в себя:

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

Группа уровней ввода включает:

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

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

Группа уровней размерности включает:

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

Предложения GSPC получили широкий отклик в виде многочисленных реализаций версий базовой системы.

GKS (Graphical Kernel System)

Результатом работ в ФРГ было создание системы GKS. Модель графической системы, положенная в ее основу, приведена на рис. 10.6. В 1979 г. GKS была принята в качестве отправной точки международного стандарта. В процессе разработки международного стандарта в исходную версию GKS был внесен целый ряд изменений, приблизивших ее к CORE-SYSTEM, но сохранивших ряд отличительных фундаментальных концепций. Основной из таких отличительных черт является введение понятия рабочей станции, представляющей собой абстракцию совокупности виртуальных устройств ввода/вывода, более полно соответствующей современной тенденции использования интеллектуальных терминалов.

По максимуму рабочая станция обладает следующими свойствами: имеет одну адресуемую поверхность вывода с фиксированным разрешением; допускается только прямоугольное поле отображения, которое не может состоять из нескольких отдельных частей; позволяет задание и использование поля отображения, которое меньше максимально возможного, с гарантией того, что изображение не генерируется вне заданного поля отображения; поддерживает несколько типов линий, шрифтов текста, размеров символов и т.д. для вывода примитивов с различными атрибутами; имеет одно или больше устройств ввода для каждого класса устройств, которые независимо друг от

друга могут работать в одном из трех режимов - синхронный ввод, опрос, асинхронный ввод; запоминает сегменты и обеспечивает средства для изменений сегментов и манипуляций с ними.

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

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

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

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

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

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

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

Рис. 10.6. Модель графической системы, положенная в основу GKS

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

Уровни вывода:

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

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

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

Уровни ввода:

A: Нет ввода.

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

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

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

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

Отличия GKS-3D от GKS заключаются в добавлении 3D функций:

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

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

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

PHIGS (Programmer's Hierarchical Interactive Graphics System)

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

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

PHIGS же комбинирует графику с техникой моделирования и представляет собой набор функций программирования графики с поддержкой быстрой модификации графических данных, описывающих геометрические соотношения объектов.

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

Принципиальное отличие PHIGS от GKS состоит в том, что в PHIGS создание и отображение изображения разделены на две независимых фазы - занесения в

централизованную структурную память (CSS - Centralized Structure Store) и отображения заданной структуры на требуемую рабочую станцию.

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

Таблица 10.1. Классы структурных элементов в CSS

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

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

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

"Обобщенный структурный элемент" позволяет определить нестандартные действия.

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

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

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

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

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

PHIGS+

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

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

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

CGI (Computer Graphics Interface)

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

Для эффективного использования аппаратных возможностей современных графических устройств набор функций CGI перекрывает аппаратно-реализуемые возможности и включает в себя следующие функции:

управление устройством,
вывод графических примитивов,
изменение графических атрибутов,
сегментация изображений,
графический ввод,
растровые операции.

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

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

10.4 Графические протоколы

Анализ применяемых в настоящее время графических протоколов и проектов по их стандартизации позволяет выделить протоколы следующих типов:

- аппаратно-зависимые графические протоколы или команды графических устройств,
- аппаратно-независимые графические протоколы или метафайлы,
- прикладные графические протоколы,
- растровые графические файлы.

10.4.1 Аппаратно-зависимые графические протоколы

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

Вопрос о поддержке тех или иных аппаратно-зависимых графических протоколов определяется составом используемого оборудования. Целесообразно, чтобы центральная

ЭВМ обеспечивала возможность генерации команд для наиболее распространенных графических устройств. В настоящее время значительная часть производящейся в мире графической аппаратуры работает с протоколами TEKTRONIX, REGIS и HPGL. Поддержка этих протоколов обеспечивается также в наиболее распространенных зарубежных программных продуктах.

Протокол TEKTRONIX

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

Существуют две разновидности протокола TEKTRONIX, соответствующие дисплеям серии 4010/12/14 и дисплеям серий 41XX, 42XX, 43XX. Дисплеи 4010/12/14 это дисплеи на запоминающей трубке с минимальным набором графических команд.

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

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

Протокол REGIS

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

Протокол HP-GL

Графический протокол HP-GL (язык описания данных Graphic Language) разработан фирмой Hewlett Packard в 1976 г. и поддерживается множеством других фирм, выпускающих графопостроители. В настоящее время используется версия HP-GL/2. Операторы языка содержат символьное имя команды и несколько параметров, также подготовленных в печатном текстовом виде. Всего в языке 88 операторов, разбитых на 9 функциональных групп. Ядро языка содержит 5 групп из 55 операторов, которые должны поддерживаться на всех устройствах. Оставшиеся 3 группы из 33 операторов являются специфичными для некоторых из устройств. В целом, благодаря использованию явного текстового представления, язык легко читается и интерпретируется.

10.4.2 Языки описания страниц

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

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

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

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

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

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

Язык PostScript

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

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

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

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

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

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

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

В язык встроены следующие изобразительные возможности:

 изображения строятся из отрезков линий, дуг и кубических кривых;

 примитивы могут быть выведены линиями требуемого вида, закрашены любым цветом или использоваться для задания области отсечения для других графических элементов;

 текст полностью интегрирован с графикой, символы как стандартных шрифтов, так и определенных пользователем, рассматриваются как графические образы, которые обрабатываются графическими операторами языка;

 2D общая координатная система поддерживает обычные линейные преобразования и их комбинации (сдвиг, поворот, масштабирование, отражение и т.д.);

 как построенные графическими операторами, так и естественные изображения, введенные, например, со сканера, могут иметь требуемые разрешение и динамический диапазон.

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

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

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

Язык PCL

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

10.4.3 Аппаратно-независимые графические протоколы

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

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

NAPLPS - North American Presentation Level Protocol Syntax

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

- возможность передачи графической информации в потоке буквенно-цифровых данных,
- минимальность объема передаваемых графических данных,
- минимальность усилий для интерпретации и возможность вывода изображений на простейшие графические устройства.

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

GKSM - Graphical Kernel System Metafile

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

Недостатком GKSM-протокола (равно как и системы GKS) является, например, минимальный набор стандартизованных графических примитивов (их всего 5), что не дает возможности эффективного использования современных интеллектуальных графических

устройств и увеличивает объем передаваемой информации. Возможность же использования обобщенного графического примитива или ESCAPE-механизма (предназначенных для нестандартных функций) не обеспечивает однозначность интерпретации графических данных при передаче их между различными системами.

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

CGM - Computer Graphics Metafile

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

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

DXF

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

WMF - Windows Metafile Format

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

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

изображений с различными параметрами визуализации (например, множество изображений пространственного объекта с различных точек зрения).

10.4.4 Проблемно-ориентированные протоколы

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

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

Растровые графические файлы

С появлением и широким распространением персональных ЭВМ, использующих растровые дисплеи и устройства документирования (лазерные и струйные принтеры и т.д.), для целей компактного хранения и транспортировки графической информации стали активно применяться различного рода растровые графические файлы. Используется более десятка различных типов растровых графических файлов. К наиболее известным относятся:

- TIFF (Tag Image File Format),
- GIF (Graphics Interchange Format),
- PIC,
- PCX,
- MAC (MacPaint),
- BMP (Bitmap).

TIFF (Tag Image File Format)

Форматы будут рассмотрены в отдельной главе.

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

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

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

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

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

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

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

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

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

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

Глава 11. Форматы графических файлов

Проблема сохранения изображений для последующей их обработки чрезвычайно важна. С ней сталкиваются пользователи любых графических систем. Изображение может быть обработано несколькими графическими программами прежде, чем примет свой окончательный вид. Например, исходная фотография сначала сканируется, затем улучшается её чёткость и производится коррекция цветов в программе **Adobe PhotoShop .** После этого изображение может быть экспортировано в программу рисования, такую как **CorelDRAW** или **Adobe Illustrator**, для добавления рисованных картинок. Если изображение создаётся для статьи в журнале или книги, то оно должно быть импортировано в издательскую систему **QuarkXPress** или **Adobe PageMaker**. Если же изображение должно появиться в мультимедиа-презентации, то оно, вероятнее всего, будет использовано в **Microsoft PowerPoint, Macromedia Director** или размещено на Web-странице.

Формат графического файла — способ представления и расположения графических данных на внешнем носителе.

Важно различать ***векторные*** и ***растровые*** форматы.

Изначально, в условиях отсутствия стандартов каждый разработчик изобретал новый формат для собственных приложений. Поэтому возникали большие проблемы обмена данными между различными программами (текстовыми процессорами, издательскими системами, пакетами иллюстративной графики, программами САПР и др.). Но с начала 80-х гг. официальные группы по стандартам начали создавать общие форматы для различных приложений. Единого формата, пригодного для всех приложений, нет и быть не может, но всё же некоторые форматы стали стандартными для целого ряда предметных областей.

Пользователю графической программы не требуется знать, как именно в том или ином формате хранится информация о графических данных. Однако умение разбираться в особенностях форматов имеет большое значение для эффективного хранения изображений и организации обмена данными между различными приложениями.

Большинство векторных форматов могут так же содержать внедрённые в файл растровые объекты или ссылку на растровый файл (технология OPI). Сложность при передаче данных из одного векторного формата в другой заключается в использовании программами различных алгоритмов, разной математики при построении векторных и описании растровых объектов.

OPI (Open Prepress Interface) — технология, разработанная фирмой Aldus, позволяющая импортировать не оригинальные файлы, а их образы, создавая в программе лишь копию низкого разрешения (эскиз) и ссылку на оригинал. В процессе печати на принтер, эскизы подменяются на оригинальные файлы. Применение OPI, вместо простого внедрения, (embedding) дает возможность экономить ресурсы компьютера (прежде всего, память), заметно повышая его производительность. OPI является основной работы с импортированными графическими файлами в таких программах, как FreeHand и QuarkXPress, широко применяется в других продуктах.

Растровый файл устроен проще (для понимания, по крайней мере). Он представляет из себя прямоугольную матрицу (bitmap), разделенную на маленькие квадратики - пикселы (pixel - picture element). Растровые файлы можно разделить на два типа: предназначенные для вывода на экран и для печати.

Разрешение файлов таких форматов как GIF, JPEG, BMP зависит от видеосистемы компьютера. В старых Маках на квадратный дюйм экрана приходилось 72 пиксела (экранное разрешение), на Windows единого стандарта не сложилось, но сегодня чаще всего употребляется значение 96 пикселов на квадратный дюйм экрана. Реально, однако, эти параметры теперь стали довольно условными, так как почти все видеосистемы современных компьютеров позволяют изменять количество отображаемых на экране пикселов. Растровые форматы, предназначенные исключительно для вывода на экран,

имеют только экранное разрешение, то есть один пиксел в файле соответствует одному экранному пикселу. На печать они выводятся так же с экранным разрешением.

Растровые файлы, предназначенные для допечатной подготовки изданий имеют, подобно большинству векторных форматов, параметр Print Size - печатный размер. С ним связано понятие печатного разрешения, которое представляет из себя соотношение количества пикселов на один квадратный дюйм страницы (ppi, pixels per inch или dpi - dots per inch, - термин не совсем верный, но часто употребимый). Печатное разрешение может быть от 130 dpi (для газеты) до 300 (высококачественная печать), больше почти никогда не нужно.

Растровые форматы, так же отличаются друг от друга способностью нести дополнительную информацию: различные цветовые модели, вектора, Альфа-каналы или каналы плашковых (spot)-цветов, слои различных типов, интерлиньяж (черезстрочная подгрузка), анимация, возможности сжатия и другое.

11.1 Векторные форматы

Файлы векторного формата содержат описания рисунков в виде набора команд для построения простейших графических объектов (линий, окружностей, прямоугольников, дуг и т. д.). Кроме того, в этих файлах хранится некоторая дополнительная информация. Различные векторные форматы отличаются набором команд и способом их кодирования.

Особенности некоторых векторных форматов приведены в Таблице 11. 1.

Таблица 11.1. Веторные форматы

Название формата	Программы, которые могут открывать файлы
WMF Windows MetaFile	Большинство приложений WINDOWS
EPS Encapsulated PostScript	Большинство настольных издательских систем и векторных программ, некоторые растровые программы
DXF Drawing Interchange Format	Все программы САПР, многие векторные редакторы,некоторые настольные издательские системы
CGM Computer Graphics Metafile	Большинство программ редактирования векторных рисунков, САПР и издательские системы

Краткое описание основных векторных форматов

WMF (Windows Metafile)

Векторный формат WMF использует графический язык Windows и, можно сказать, является ее родным форматом. Служит для передачи векторов через буфер обмена (Clipboard). Понимается практически всеми программами Windows, так или иначе связанными с векторной графикой. Однако, несмотря на кажущуюся простоту и универсальность, пользоваться форматом WMF стоит только в крайних случаях для передачи "голых" векторов. WMF искажает цвет, не может сохранять ряд параметров, которые могут быть присвоены объектам в различных векторных редакторах, не может содержать растровые объекты, не понимается очень многими программами на Макинтош. Из известных автору графических программ, найболее корректно создавать WMF-файлы может только CorelDRAW.

EPS (Encapsulated PostScript)

Формат Encapsulated PostScript можно назвать самым надежным и универсальным способом сохранения 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 векторов и растра в издательские системы, создается почти всеми программами, работающими с графикой. Использовать его имеет смысл только тогда, когда вывод осуществляется на PostScript-устройстве. EPS поддерживает все необходимые для печати цветовые модели, среди них такая, как Duotone, может записывать, так же, данные в RGB, обтравочный контур, информацию и треппинге и растрах, внедренные шрифты. В формате EPS сохраняют данные в буфере обмена (Clipboard) программы Adobe для обмена между собой.

Вместе с файлом можно сохранить эскиз (image header, preview). Это копия низкого разрешения в формате PICT, TIFF, JPEG или WMF, которая сохраняется вместе с файлом EPS и позволяет увидеть, что внутри, поскольку открыть файл на редакцию могут только Photoshop и Illustrator. Все остальные импортируют эскиз, подменяя его при печати на PostScript-принтере оригинальной информацией. На принтере, не поддерживающем PostScript, выводится на печать сам эскиз. Если вы работаете на Photoshop для Макинтош, сохраняйте эскизы в формате JPEG, остальные маковские программы сохраняют эскизы в формате PICT. Эти и JPEG-эскизы не могут использовать Windows-приложения. Если вы работаете на PC или не знаете, где будет использоваться файл, сохраняйте эскиз в формате TIFF (когда предоставляется выбор). CorelDRAW так же предлагает для эскиза векторный формат WMF, стоит очень осторожно пользоваться этим детищем Microsoft - до добра не доведет.

Photoshop позволяет сжимать растровые данные с помощью алгоритма JPEG. Adobe доработала этот способ сжатия. Теперь JPEG, в исполнении Photoshop, поддерживает CMYK и **сжимает лучше**, чем JPEG, полностью соответствующий первоначальным спецификациям. Другими словами, EPS-файлы без эскиза с JPEG-кодированием весят меньше, чем аналогичные файлы формата JPEG! **Драйверы принтеров и фотонаборных автоматов не могут выполнять цветоделение таких файлов.** То есть при выполнении цветоделения на вашем компьютере EPS-картинка с JPEG-сжатием полностью окажется на первой плате (Cyan, обычно). Тем не менее, в сервисном бюро рабочие станции Scitex (их большинство в Израиле) могут цветоделить страницы с JPEG EPS-иллюстрациями без всяких проблем. Системы других фирм, думаю, так же поддерживают JPEG EPS, в любом случае стоит поинтересоваться. В сервисных бюро и типографиях Тель-Авива мне часто рекомендовали использовать для записи растровых данных именно JPEG EPS вместо TIFF, так как он быстрее выводится.

EPS имеет много разновидностей, что зависит от программы-создателя. Самые надежные EPS создают программы производства Adobe Systems: Photoshop, Illustrator, InDesign. С 1996 года программы Adobe имеют встроенный интерпретатор PostScript, поэтому могут открывать EPS и редактировать их. Эта возможность представляется мне очень важной. Остальные графические редакторы открывать EPS не могут, мало того, создаваемые ими EPS-файлы иногда оказываются, мягко говоря, особенными. Среди самых проблемных Quark EPS, создаваемый функцией Save Page As EPS и FreeHand editable EPS, создаваемый функцией Save As. Не стоит особенно доверять Corel'овским EPS версии 6 и ниже и EPS из CorelXARA. У EPS-файлов из CorelDRAW 7 и выше сохраняется проблема добавления полей к Bounding Box (условный прямоугольник в PostScript, описывающий все объекты на странице). Прежде, чем экспортировать из CorelDRAW, CorelXARA и, в меньшей степени, из FreeHand'a EPS-файлы стоит конвертировать многие эффекты программ (полупрозрачные заливки, например) в

растровые или простые векторные объекты. Толстые контуры (более 2 pt), возможно, имеет смысл конвертировать так же в объекты, когда программа дает такую возможность.

11.2 Растровые форматы

В файлах растровых форматов запоминаются:

• размер изображения — количество видеопикселей в рисунке по горизонтали и вертикали

• битовая глубина — число битов, используемых для хранения цвета одного видеопикселя

• данные, описывающие рисунок (цвет каждого видеопикселя рисунка), а также некоторая дополнительная информация.

В файлах растровой графики разных форматов эти характеристики хранятся различными способами.

Поскольку размер изображения хранится в виде отдельной записи, цвета всех видеопикселей рисунка запоминаются как один большой блок данных. Так как растровое представление изображения кораблика достаточно громоздко, рассмотрим как сохраняется в растровом файле простое чёрно-белое изображение (рис. 11.1).

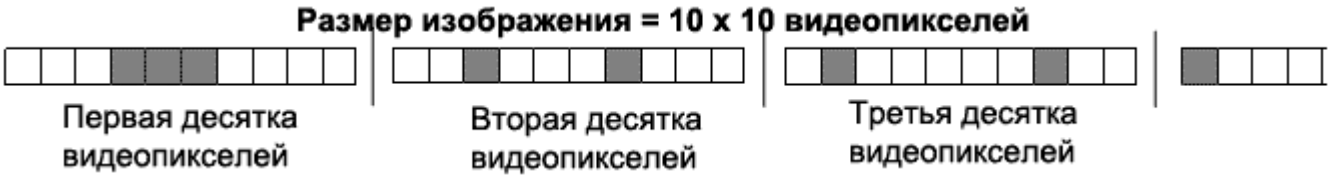


Рис. 11.1. В растровом файле сохраняется информация о цвете каждого видеопикселя

На рис. 11.2 показан результат восстановления изображения по информации, сохранённой в растровом файле, представленном на рис. 11.1. В изображении, восстановленном по файлу, видеопиксели располагаются согласно размеру изображения; а именно, сначала — первая десятка видеопикселей, в следующей строке — вторая десятка и т. д., в десятой строке — последние десять видеопикселей.

Рис. 11.2. Растровый рисунок, восстановленный по файлу растровой графики

Изображения фотографического качества, полученные с помощью сканеров с высокой разрешающей способностью, часто занимают несколько мегабайт. Например, если размер изображения 1766 x 1528, а количество используемых цветов — 16777216, то объём растрового файла составляет около 8 Мб (информация о цвете видеопикселей в файле занимает 1766 x 1528 x 24 / 8 / 1024 / 1024 Мб).

Решением проблемы хранения растровых изображений является сжатие, т. е. уменьшение размера файла за счёт изменения способа организации данных. Никому пока

не удалось даже приблизиться к созданию идеального алгоритма сжатия. Каждый алгоритм хорошо сжимает только данные вполне определённой структуры.

Методы сжатия делятся на две категории:

• сжатие файла с помощью программ — архиваторов;

• сжатие, алгоритм которого включён в формат файла.

В первом случае специальная программа считывает исходный файл, применяет к нему некоторый сжимающий алгоритм (архивирует) и создаёт новый файл. Выигрыш в размере нового файла может быть значительным. Однако этот файл не может быть использован ни одной программой до тех пор, пока он не будет преобразован в исходное состояние (разархивирован). Поэтому такое сжатие применимо только для длительного хранения и пересылки данных, но для повседневной работы оно неудобно. В системах DOS и WINDOWS наиболее популярными программами сжатия файлов являются ZIP, ARJ, RAR и другие.

Если же алгоритм сжатия включён в формат файла, то соответствующие программы чтения правильно интерпретируют сжатые данные. Таким образом, такой вид сжатия очень удобен для постоянной работы с графическими файлами большого размера. Например, пусть в **CorelDRAW** получен рисунок, который нужно разместить в документе, созданном в программе **Adobe PhotoShop. TIFF** — один из растровых форматов, с которыми может работать **Adobe PhotoShop.** При формировании файла формата **TIFF** выполняется сжатие графических данных. Именно это обстоятельство учитывается соответствующей программой чтения. Поэтому для достижения поставленной цели можно поступить следующим образом:

• сохранить рисунок, созданный в **CorelDRAW**, в файле формата **TIFF;**

• импортировать этот файл в программу **Adobe PhotoShop.**

Краткое описание основных растрвых форматов

BMP (Windows Device Independent Bitmap)

Еще один родной 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, больше ни на что не пригоден. Способен хранить как индексированный (до 256 цветов), так и RGB-цвет (более 16 млн. оттенков). Возможно применение сжатия по принципу RLE, но делать это не рекомендуется, так как очень многие программы таких файлов (они могут иметь расширение .rle) не понимают. Существует разновидность формата BMP для операционной системы OS/2.

Использование BMP не для нужд Windows является распространенной ошибкой новичков. Важно понимать - **использовать BMP не желательно** ни в веб, ни для печати (особенно), ни для простого переноса и хранения информации.

GIF (CompuServe Graphics Interchange Format)

Независящий от аппаратного обеспечения формат GIF был разработан в 1987 году (GIF87a) фирмой CompuServe для передачи растровых изображений по сетям. В 1989-м формат был модифицирован (GIF89a), были добавлены поддержка прозрачности и анимации. GIF использует LZW-компрессию, что позволяет неплохо сжимать файлы, в которых много однородных заливок (логотипы, надписи, схемы).

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 пикселов 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 файл поднимется весь. Черезстрочная запись незначительно увеличивает размер файла, но это, как правило, оправдывается приобретаемым свойством.

В GIF’е 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.

PNG (Portable Network Graphics)

PNG - разработанный относительно недавно формат для Сети, призванный заменить собой устаревший GIF. Использует сжатие без потерь Deflate, сходное с LZW (именно из-за патентования в 1995-м году алгоритма LZW возник PNG). Сжатые индексированные файлы PNG, как правило, меньше аналогичных GIF'ов, PNG RGB меньше соответствующего файла в формате TIFF.

Глубина цвета файлах PNG может быть любой, вплоть до 48 бит. Используется двумерный interlacing (не только строк, но и столбцов), который, так же, как и в GIF'е, несколько увеличивает размер файла. В отличие от GIF'а, где прозрачность как мед - либо есть, либо нет, PNG поддерживает также полупрозрачные пикселы за счет Альфа-канала с 256 градациями серого.

В файл формата PNG записывается информация о гамма-коррекции. Гамма представляет собой некое число, характеризующее зависимость яркости свечения экрана вашего монитора от напряжения на электродах кинескопа. Это число, считанное из файла, позволяет ввести поправку яркости при отображении. Нужно оно для того, чтобы картинка, созданная на Мас’е, выглядела одинаково и на Windows, и на различных UNIX'ах. Таким образом, эта особенность помогает реализации основной идеи WWW - одинакового отображения информации независимо от аппаратуры пользователя.

PNG поддерживается в Microsoft Internet Explorer начиная с версии 4 для Windows и с версии 4.5 на Макинтош. Netscape добавила поддержку PNG для своего браузера в версиях, начиная с 4.0.4 для обеих платформ. Тем не менее до сих пор не реализована поддержка таких важных функций формата, как плавно переходящая прозрачность и гамма-коррекция.

TIFF (Tagged Image File Format)

Аппаратно независимый формат TIFF появился как внутренний формат программы Aldus PhotoStyler. Его модульная архитектура оказалась настолько удачной, что, успешно пережив смерть родной программы, TIFF и в наши дни продолжает совершенствоваться и развиваться. Сегодня он является одним из самых распространенных и надежных, его поддерживают практически все программы на PC и Макинтош так или иначе связанные с графикой. Как правило, TIFF является лучшим выбором при импорте растровой графики в векторные программы и издательские системы. Ему доступен весь диапазон цветовых моделей от монохромной до RGB, CMYK и дополнительных плашковых цветов. TIFF может содержать обтравочные контуры, Альфа-каналы, слои, другие дополнительные данные.

Исключение, в какой-то мере, составляет FreeHand. Иногда TIFF-файлы в нем могут произвольно менять свое местоположение при создании PostScript-файла или прямо в документе при открытии. Чаще TIFF’ы "прыгают" находясь в обтравочном контуре. С FreeHand’ом, все же, предпочтительнее использовать EPS.

TIFF может сохраняться в двух порядках записи: Macintosh и PC. Это связано с тем, что процессоры Motorola читают и записывают числа слева направо, а процессоры Intel - наоборот. Современные программы могут без проблем использовать оба варианта формата.

В формате TIFF есть возможность сохранения с применением нескольких видов сжатия: JPEG, ZIP, но, как правило используется только LZW-компрессия. Ряд старых

программ (например, QuarkXPress 3.x, Adobe Streamline, многие программы-распознаватели текста) не умеют читать сжатые файлы TIFF, однако, если вы пользуетесь новым программным обеспечением, нет причины не использовать компрессию.

11.3 Методы сжатия графических данных

При **сжатии методом RLE** (**R**un — **L**ength **E**ncoding) последовательность повторяющихся величин (в нашем случае — набор бит для представления видеопикселя) заменяется парой — повторяющейся величиной и числом её повторений.

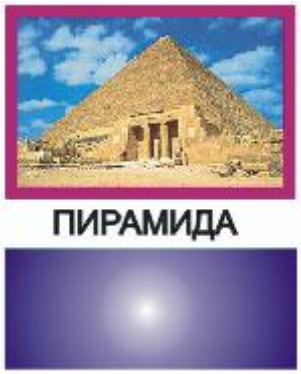
Метод сжатия RLE включается в некоторые графические форматы, например, в формат **PCX .**

Программа сжатия файла может сначала записывать количество видеопикселей, а затем их цвет или наоборот. Поэтому возможна такая ситуация, когда программа, считывающая файл, ожидает появления данных в ином порядке, чем программа, сохраняющая этот файл на диске. Если при попытке открыть файл, сжатый методом **RLE,** появляется сообщение об ошибке или полностью искажённое изображение, нужно считать этот файл с помощью другой программы или преобразовать его в иной формат.

Сжатие методом RLE наиболее эффективно для изображений, которые содержат большие области однотонной закраски, и наименее эффективно — для отсканированных фотографий, так как в них нет длинных последовательностей одинаковых видеопикселей .

Метод сжатия LZW основан на поиске повторяющихся узоров в изображении. LZW (Lempel-Ziv-Welch) разработан в 1978 году израильтянами Лемпелом и Зивом и доработан позднее в США. 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 пикселов, 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 LZW, так же, как и RLE, лучше действует на участках однородных, свободных от шума цветов, он действует гораздо лучше, чем RLE, при сжатии произвольных графических данных, но процесс кодирования и распаковки происходит медленнее.

Сильно насыщенные узорами рисунки могут сжиматься до 0,1 их первоначального размера. Метод сжатия **LZW** применяется для файлов форматов **TIFF** и **GIF;** при этом данные формата **GIF** сжимаются всегда, а в случае формата **TIFF** право выбора возможности сжатия предоставляется пользователю. Существуют варианты формата **TIFF,** которые используют другие методы сжатия. Из-за различных схем сжатия некоторые версии формата **TIFF** могут оказаться несовместимыми друг с другом. Это означает, что возможна ситуация, когда файл в формате **TIFF** не может быть прочитан в некоторой графической программе, хотя она должна «понимать» этот формат. Другими словами, не все форматы **TIFF** одинаковы. Но, несмотря на эту проблему, **TIFF** является одним из самых популярных растровых форматов в настоящее время.



Хорошо сжимаемое изображение Плохо сжимаемое изображение

Рис. 11.3**.** Сжатие методом RLE

Метод сжатия JPEG обеспечивает высокий коэффициент сжатия для рисунков фотографического качества. Формат файла **JPEG ,** использующий этот метод сжатия, разработан объединенной группой экспертов по фотографии (**J**oint **P**hotographic **E**xperts **G**roup). Сжатие по методу **JPEG** сильно уменьшает размер файла с растровым рисунком (возможен коэффициент сжатия 100 : 1). Высокий коэффициент сжатия достигается за счет сжатия с потерями, при котором в результирующем файле теряется часть исходной информации. Метод **JPEG** использует тот факт, что человеческий глаз очень чувствителен к изменению яркости, но изменения цвета он замечает хуже. Поэтому при сжатии этим методом запоминается больше информации о разнице между яркостями видеопикселей и меньше — о разнице между их цветами. Так как вероятность заметить минимальные различия в цвете соседних пикселей мала, изображение после восстановления выглядит почти неизменным. Пользователю предоставляется возможность контролировать уровень потерь, указывая степень сжатия. Благодаря этому, можно выбрать наиболее подходящий режим обработки каждого изображения: возможность задания коэффициента сжатия позволяет сделать выбор между качеством изображения и экономией памяти. Если сохраняемое изображение — фотография, предназначенная для высокохудожественного издания , то ни о каких потерях не может быть и речи, так как рисунок должен быть воспроизведён как можно точнее. Если же изображение — фотография, которая будет размещена на поздравительной открытке, то потеря части исходной информации не имеет большого значения. Эксперимент поможет определить наиболее допустимый уровень потерь для каждого изображения.

JPEG – это алгоритм сжатия, основанный не на поиске одинаковых элементов, как в RLE и LZW, а на разнице между пикселами. Кодирование данных происходит в несколько этапов. Сначала графические данные конвертируются в цветовое пространство типа LAB, затем отбрасывается половина или три четверти информации о цвете (в зависимости от реализации алгоритма). Далее анализируются блоки 8x8 пикселов. Для каждого блока формируется набор чисел. Первые несколько чисел представляют цвет блока в целом, в то время, как последующие числа отражают тонкие делали. Спектр деталей базируется на зрительном восприятии человека, поэтому крупные детали более заметны.

На следующем этапе, в зависимости от выбранного вами уровня качества, отбрасывается определенная часть чисел, представляющих тонкие детали. На последнем этапе используется кодирование методом Хафмана для более эффективного сжатия конечных данных. Восстановление данных происходит в обратном порядке.

Таким образом, чем выше уровень компрессии, тем больше данных отбрасывается, тем ниже качество. Используя JPEG можно получить файл в 1-500 раз меньше, чем BMP!

Формат аппаратно независим, полностью поддерживается на PC и Macintosh, однако он относительно нов и не понимается старыми программами (до 1995 года). JPEG не поддерживает индексированные палитры цветов. Первоначально в спецификациях формата не было и CMYK, Adobe добавила поддержку цветоделения, однако CMYK JPEG во многих программах делает проблемы. Лучшим решением является использование JPEG-сжатия в Photoshop EPS-файлах.

Существуют подформаты JPEG. Baseline Optimized - файлы несколько лучше сжимаются, но не читаются некоторыми программами. JPEG Baseline Optimized разработан специально для Интернета, все основные браузеры его поддерживают. Progressive JPEG так же разработан специально для Сети, его файлы меньше стандартных, но чуть больше Baseline Optimized. Главная особенность Progressive JPEG в поддержке аналога черезстрочного вывода.

Из сказанного можно сделать следующие выводы. JPEG'ом лучше сжимаются 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 с высокой печатной резолюцией (200-300 и более dpi), чем с низкой (72-150 dpi), т.к. в каждом квадрате 8x8 пикселов переходы получаются более мягкие, за счет того, что их (квадратов) в таких файлах больше. Нежелательно сохранять с JPEG-сжатием любые изображения, где важны все нюансы цветопередачи (репродукции), так как во время сжатия происходит отбрасывание цветовой информации. В JPEG'е следует сохранять только конечный вариант работы, потому что каждое пересохранение приводит ко все новым потерям (отбрасыванию) данных и превращении исходного изображения с кашу.

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 к красному, в то время как канал B - от синего к желтому. Средние значения для A и B соответствуют реальным оттенкам серого. Существует похожая цветовая модель YCC, используемая в форматах Kodak Photo CD и FlashPix, здесь не описываемых.

Метод сжатия Хаффмана (Huffman) разработан в 1952 году и используется как составная часть в ряде других схем сжатия, таких как LZW, Дефляция, JPEG. В методе Хаффмана берется набор символов, который анализируется, чтобы определить частоту каждого символа. Затем для наиболее часто встречающихся символов используется представление в виде минимально возможного количества битов. Например, буква "e" чаще всего встречается в английских текстах. Используя кодировку Хаффмана вы можете представить "e" всего лишь двумя битами (1 и 0), вместо восьми битов, необходимых для представления буквы "e" в кодировке ASCII.

Информация о методах сжатия, используемых в растровых форматах файлов, приведена в таблице 11.2.

Таблица 11.2. Растровые форматы графических файлов

Название формата	Программы, которые могут открывать файлы	Метод сжатия
BMP Windows Device Independent Bitmap	Все программы WINDOWS, которые используют растровую графику	**RLE** для 16- и 256- цветных изображений (по желанию)
PCX Z - Soft PaintBrush	Почти все графические приложения для PC	**RLE** (всегда)

GIF Graphic Interchange Format	Почти все растровые редакторы; большинство издательских пакетов; векторные редакторы, поддерживающие растровые объекты	**LZW** (всегда)
TIFF Tagged Image File Format	Большинство растровых редакторов и настольных издательских систем; векторные редакторы, поддерживающие растровые объекты	**LZW** (по желанию) и др.
TGA TrueVision Targa	Программы редактирования растровой графики	**RLE** (по желанию)
IMG Digital Research GEM Bitmap	Некоторые настольные издательские системы и редакторы изображений WINDOWS	**RLE** (всегда)
JPEG Joint Photographic Experts Group	Последние версии программ редактирования растровой графики; векторные редакторы, поддерживающие растровые объекты	**JPEG** (можно выбрать степень сжатия)

О некотоых других форматах

Adobe PostScript

PostScript - язык описания страниц (язык управления лазерными принтерами) фирмы Adobe. Был создан в 80-х годах для реализации принципа WYSIWYG (What You See is What You Get). Файлы этого формата представляют из себя программу с командами на выполнение для выводного устройства. Они имеют расширение .ps или, реже, .prn 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 так же другую информацию: платы цветоделения, дополнительные платы, линиатуру растра и форму растровой точки для каждой платы и другие данные для выводного устройства. Если файл создан правильно, не имеет значения на какой платформе он делался, были использованы шрифты True Type или Adobe Type 1 - все равно. Тем не менее нужно учитывать, что даже в том случае, когда вы сделали верные установки в окне печати, могут возникнуть проблемы связанные с некорректным переводом используемой вами программы ее графического языка на язык PostScript (например, внедрением информации о неиспользуемых шрифтах). Наиболее корректные PS-файлы создают программы Adobe.

Данные в PostScript-файле, как правило, записываются в двоичной кодировке (Binary). Бинарный код занимает вдвое меньше места, чем ASCII. Кодировка ASCII иногда требуется для передачи файлов через сети, для кроссплатформенного обмена, для печати через последовательные кабели. В приведенных случаях двоичная кодировка может исказиться (что сделает файл нечитаемым) или вызвать "странное" поведение файл-сервера. Эти проблемы давно изжиты в современных системах, но старые компьютеры и серверы бывают им подвержены. Сказанное относится ко всем форматам, основанным на языке PostScript: EPS и PDF, которые описываются ниже.

PDF (Portable Document Format)

PDF предложен фирмой Adobe как независимый от платформы формат для создания электронной документации, презентаций, передачи верстки и графики через сети. Используется как внутренний графический формат в Mac OS X.

PDF-файлы создаются путем конвертации из PostScript-файлов или функцией экспорта ряда программ. Для конвертации используется программа Adobe Acrobat Distiller, это лучший способ создания PDF. Создание PDF методом экспорта из программ дает, как правило худший результат - файлы получаются более тяжелыми, часто имеют проблемы со встраиванием шрифтов.

Для создания PDF так же существует программа PDFWriter, работающая как виртуальный принтер. PDFWriter не основан на PostScript и не может корректно обрабатывать графику. Он предназначен для быстрого изготовления простых текстовых документов. У него наблюдается та же проблема со встраиванием шрифтов, что и многих программ, умеющих экспортировать PDF. Самые надежные и максимально близкие к оригиналу PDF создает из PostScript и EPS-файлов программа Acrobat Distiller, поставляемая в пакете Adobe Acrobat.

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, RLE, CCITT, ZIP (похожее на LZW и известное еще как Deflate). Программа Acrobat Exchange 3 (которая в 4-й версии стала называться просто Acrobat 4.0) позволяет расставлять гиперссылки, заполняемые поля, включать в файл PDF видео и звук, другие действия.

Метод сжатия CCITT (International Telegraph and Telephone Committie) был разработан для факсимильной передачи и приема. Является более узкой версией кодирования методом Хаффмана. CCITT Group 3 идентичен формату факсовых сообщений, CCITT Group 4 - формат факсов, но без специальной управляющей информации.

Файл PDF может быть оптимизирован. Из него удаляются повторяющиеся элементы, устанавливается постраничный порядок загрузки страниц через web, с приоритетом сначала для текста, потом графика, наконец шрифты. Однако, когда повторяющихся элементов нет, файл, после оптимизации, как правило, несколько увеличивается.

PDF все больше используется для передачи по сетям в компактном виде графики и верстки. Он может сохранять всю информацию для выводного устройства, которая была в исходном PostScript-файле. Это касается PDF версий 1.2 (Acrobat 3) и выше. Однако, версия 1.2 не может включать сведения о треппинге, некоторые другие специфические данные (DSC, например), не использует цветовые профили. Все это реализовано в последующих вариантах формата.

Adobe Photoshop Document

Внутренний формат популярного растрового редактора Photoshop в последнее время стал поддерживаться все большим количеством программ. Он позволяет записывать изображение со многими слоями (до 1000), их масками, дополнительными Альфа-каналами и каналами простых (spot) цветов, корректирующими, векторными, текстовыми слоями, контурами и другой информацией - все, что может сделать Photoshop. В версии 3.0 появляются слои, контуры и RLE-компрессия, в 4-й версии алгоритм улучшается, файлы становятся еще меньше. В версии 5 реализован принципиально иной подход к управлению цветом. В программу была внедрена архитектура управления цветом, основанная на профилях для сканеров, мониторов и принтеров Международного консорциума по цвету (International Color Consortium, ICC). В шестой версии метод управления цветом переработан.

Несмотря на многочисленные дополнения функциями формат Photoshop'a имеет полную совместимость от 3-й до 7-й версии. В Photoshop'e 2.5 не было слоев и контуров, поэтому он выступает, как отдельный подформат.

Однослойный Photoshop Document понимают ряд программ, многослойные могут импортировать Illustrator и InDesidn. Corel Painter и Corel PHOTO-PAINT открывают на редакцию многослойные документы Photoshop.

Одной из простейших форм сжатия является **метод RLE** (Run Length Encoding - кодирование с переменной длиной строки). Действие метода RLE заключается в поиске одинаковых пикселов в одной строке. Если в строке, допустим, имеется 3 пиксела белого цвета, 21 - черного, затем 14 - белого, то применение RLE дает возможность не запоминать каждый из них (38 пикселов), а записать как 3 белых, 21 черный и 14 белых в первой строке.

Так же как и LZW, RLE хорошо работает с искусственными и пастеризованными картинками и плохо с фотографиями. В действительности, если фотография детализирована, RLE может даже увеличить размер файла.

Adobe Illustrator Document

Adobe Illustrator - самый первый продукт Adobe. Он был создан сразу же после выхода PostScript Level 1, его можно назвать интерфейсом для PostScript. До 9-й версии ядро формата основывалось на EPS, с 9-й в основе лежит ядро PDF. Это дает основание полагать, что в будущем появится, наконец, многостраничность. Формат Illustrator'a напрямую открывается Photoshop'ом, его поддерживают почти все программы Макинтош и Windows так или иначе связанные с векторной графикой и графикой вообще. Все, что создает Adobe Illustrator, совместимо PostScript (исключение составляют, разве что Gradient Meshes, которые нужно растеризовать перед закрытием на печать).

Формат Illustrator’ра является **наилучшим посредником** при передаче векторов из одной программы в другую, с PC на Macintosh и назад. Наиболее совместимыми можно назвать 3-ю и 4-ю версии. При передаче градиентных заливок между векторными редакторами в редактируемом виде (когда они не конвертируются в последовательность фигур) нужно использовать версии формата, начиная с 6-й. Внедренные или связанные с документом растровые файлы при обмене через формат Illustrator'a во всех программах, кроме FreeHand версии 9 и выше, теряются. Начиная с 9-й версии формат Illustrator'a может содержать внедренные шрифты (включая такие особенные шрифтовые форматы как Adobe Type 3 и Adobe Multiple Master) и ICC-профиль. Illustrator 9 позволяет сориентировать проект на цветовое пространство RGB или CMYK и сохранить это в файле.

CorelDRAW Document

Формат известен в прошлом низкой устойчивостью, плохой совместимостью файлов, искажением цветовых характеристик внедряемых битовых карт, тем не менее пользоваться CorelDRAW чрезвычайно удобно, он имеет неоспоримое лидерство на платформе PC. Многие программы на PC (FreeHand, Illustrator, PageMaker - среди них) могут импортировать файлы CorelDRAW.

В седьмой версии многие основные проблемы были решены. Ее, 8-ю и 9-ю версии CorelDRAW можно без натяжек назвать профессиональными. В файлах этих версий применяется компрессия для векторов и растра отдельно, могут внедряться шрифты, файлы CorelDRAW имеют огромное рабочее поле 45x45 метров; начиная с 4-й версии поддерживается многостраничность, начиная с 7-й - технология OPI.

Файлы формата CorelDRAW можно применять для переноса/передачи работ на PC, но нежелательно импортировать в программы верстки. На Макинтош файлы CorelDRAW для Windows открывают версия CorelDRAW для Макинтош и Adobe Illustrator 8 и выше.

PICT (Macintosh QuickDraw Picture Format)

PICT - собственный формат Mac OS Classic. Стандарт для буфера обмена, использует графический язык Mac OS. PICT способен нести растровую, векторную

информацию, текст и звук, использует RLE-компрессию. Поддерживается на Mac'е всеми программами. Чисто битовые PICT-файлы могут иметь любую глубину битового представления (от Lineart до CMYK). Векторные PICT-файлы, которые почти исчезли из употребления в наши дни, имели странные проблемы с толщиной линии и другими отклонениями при печати. Формат используется для потребностей Mac OS, и при создании определенных типов презентаций только для Макинтош. Вне Макинтош PICT имеет расширение .pic или .pct, читается отдельными программами, но работа с ним редко бывает простой и бесхитростной.

RTF (Microsoft Rich Text Format)

Текстовый формат RTF попал сюда за свои неординарные способности к переносу текстов из одной программы в другую. Он позволяет передавать **форматированный** текст из программ оптического распознавания символов или текстовых редакторов в графические программы или в любых других направлениях. RTF может оказаться хорошим решением (а, иногда, и единственным выходом) при переброске из программы в программу нелатинского, например, ивритского текста или русского в Windows 95/98 Hebrew Edition.

Секрет совместимости заключается в использовании специальных тегов форматирования RTF и Unicode. Именно Unicode (использованный как основа формата Microsoft Word 97/98 для Макинтош и PC) позволяет легко переносить русские тексты с PC на Мак и обратно в файлах MS Word 97/98 (верно и для более высоких версий Word'а). RTF используется как основной в поставляемом вместе с Mac OS X редакторе TextEdit и в прилагаемом к Windows программе WordPad.

О сохранении изображений в собственных и «чужих» форматах

Как правило, графические программы используют свои собственные форматы для сохранения изображений во внешней памяти. *Собственный файловый формат — частный и наиболее эффективный формат* для хранения файлов отдельного графического приложения. Например, «родной» формат **CorelDRAW — CDR , Adobe PhotoShop — PSD, Fractal Design Painter — RIFF, Paint** (стандартная программа WINDOWS) — **BMP**. При сохранении изображения в файле всегда нужно указывать тип формата. На рис. 11.4 показано диалоговое окно **(Export \Экспорт),** используемое в программе **CorelDRAW.**

Кроме того, для каждого «чужого» графического формата открываются дополнительные диалоговые окна, с помощью которых пользователь устанавливает параметры формата (количество используемых цветов, необходимость сжатия — для **BMP** и **TIFF**, коэффициент сжатия — для **JPEG** и др.).

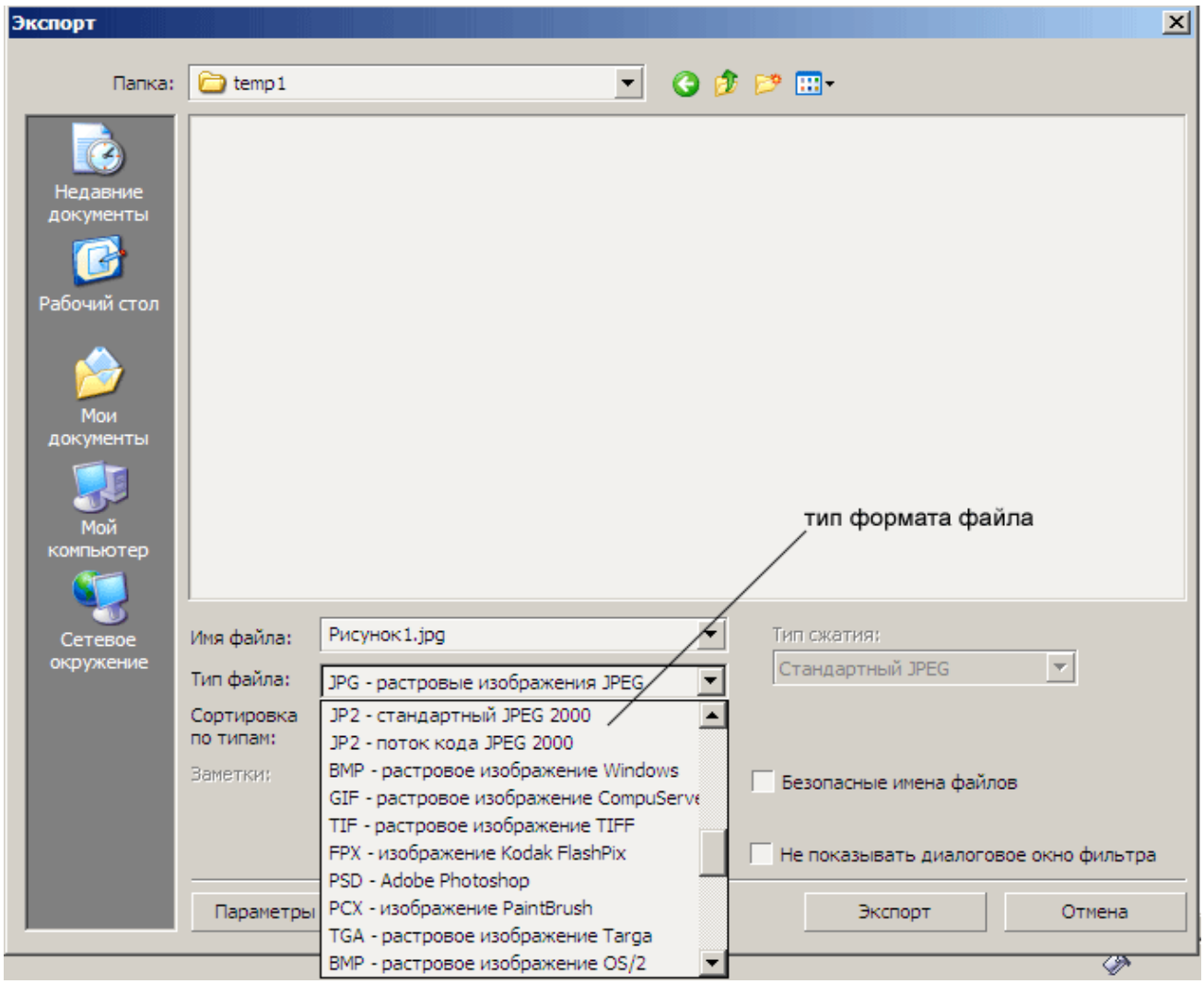


Рис. 11.4**.** Диалоговое окно для сохранения в **CorelDRAW** изображения в формате **JPEG**

11.4 Преобразование файлов из одного формата в другой

Необходимость преобразования графических файлов из одного формата в другой может возникнуть по разным причинам:

• программа, с которой работает пользователь, не воспринимает формат его файла;

• данные, которые надо передать другому пользователю, должны быть представлены в специальном формате.

Преобразование файлов из растрового формата в векторный

Существуют два способа преобразования файлов из растрового формата в векторный:

1) *преобразование растрового файла в растровый объект векторного изображения;*

2) *трассировка растрового изображения для создания векторного объекта.*

Первый способ используется в программе **CorelDRAW**, которая, как правило, успешно импортирует файлы различных растровых форматов. К примеру, если растровая картинка содержит 16 миллионов цветов, **CorelDRAW** покажет изображение, приближенное по качеству к телевизионному. Однако, импортируемый растровый объект может становиться довольно большим даже в том случае, если исходный файл невелик. В файлах растровых форматов информация хранится достаточно эффективно, так как часто используются методы сжатия. Векторные форматы такой способностью не обладают.

Поэтому растровый объект, хранящийся в векторном файле, может значительно превосходить по размерам исходный растровый файл.

Особенность второго способа преобразования растрового изображения в векторное заключается в следующем. Программа трассировки растровых изображений (например, **CorelTRACE)** ищет группы пикселей с одинаковым цветом, а затем создает соответствующие им векторные объекты. После трассировки векторизованные рисунки можно редактировать как угодно. На рис. 11.5 показано растровое изображение, которое хорошо преобразуется в векторное. Дело в том, что растровые рисунки, имеющие четко выраженные границы между группами пикселей одинакового цвета, хорошо переводятся в векторные. В то же время результат трассировки растрового изображения фотографического качества со сложными цветовыми переходами выглядит хуже оригинала.

Исходный растровый рисунок

Векторизованный рисунок

Рис. 11.5**.** Растровый рисунок с четкими границами, преобразованный в векторный формат

Преобразование файлов одного векторного формата в другой

Векторные форматы содержат описания линий, дуг, закрашенных полей, текста и т. д. В различных векторных форматах эти объекты описываются по-разному. Когда программа пытается преобразовать один векторный формат в другой, она действует подобно обычному переводчику, а именно:

- считывает описания объектов на одном векторном языке,
- пытается перевести их на язык нового формата.

Исходное растровыое изображение

Векторизованное изображение

Рис.11.6. Растровое изображение фотографического качества, преобразованное в векторный формат

Если программа-переводчик считает описание объекта, для которого в новом формате нет точного соответствия, этот объект может быть либо описан похожими командами нового языка, либо не описан вообще. Таким образом, некоторые части рисунка могут исказиться или исчезнуть. Всё зависит от сложности исходного изображения. На рис. 11.7 представлен один из возможных результатов преобразования файла из одного векторного формата в другие. Исходный рисунок создан в программе CorelDRAW и состоит из следующих элементов: импортированная растровая картинка в формате JPEG , рамка вокруг растровой картинки, текст, прямоугольник с конической заливкой.

При преобразовании рисунка 11.7а в формат CGM сохранились все исходные элементы. Формат DXF проигнорировал растровую картинку, исказил контур вокруг нее, коническую заливку, а также увеличил размер шрифта. Дело в том, что этот формат предназначен для конструкторских разработок и, следовательно, в нём отсутствуют команды для описания различных художественных эффектов.

а) Исходное изображение в формате **CDR**

б) Результат преобразования в векторный форамат **CGM**

в) Результат преобразования в векторный форамат **DXF**

Рис. 11.7**.** Результаты преобразования одного векторного формата в другой

Преобразование файлов из векторного формата в растровый

Преобразование изображений из векторного формата в растровый (этот процесс часто называют растрированием векторного изображения) встречается очень часто. Прежде, чем разместить рисованную (векторную) картинку на фотографии, её необходимо экспортировать в растровый формат. Например, изображение окна на рис. 11.8 было отсканировано и сохранено в файле формата **JPEG**. Рисунок котенка создан в векторной программе **CorelDRAW** и затем экспортирован в файл формата **TIFF**. Монтаж двух растровых изображений выполнен в программе **Adobe PhotoShop.**

Рис. 11.8. Рисованная картинка, вставленная в фотографию

Каждый раз, когда векторный рисунок направляется на устройство вывода (в частности, монитор или принтер), он подвергается растрированию — преобразованию в набор видеопикселей или точек.

При экспорте векторных файлов в растровый формат может быть потеряна информация, связанная с цветом исходного изображения. Это объясняется тем, что в ряде растровых форматов количество цветов ограничено (например, формат **GIF** использует не более 256 цветов).

Преобразование файлов одного растрового формата в другой

Этот вид преобразования обычно самый простой и заключается в чтении информации из исходного файла и записи ее в новом файле, где данные о размере изображения, битовой глубине и цвете каждого видеопикселя хранятся другим способом. Если старый формат использует больше цветов, чем новый, то возможна потеря информации. Преобразование файла с 24-битовым цветом (16777216 цветов) в файл с 8-битовым цветом (256 цветов) требует изменения цвета почти каждого пикселя. В простейшем случае это делается так: для каждого пикселя исходного файла ищется наиболее близкий к нему цвет из нового ограниченного набора цветов. При таком способе возможны нежелательные эффекты, когда часть рисунка, содержащая большое количество элементов, оказывается закрашенной одним цветом или когда плавные переходы цвета становятся резкими. На рис. 11.9 показано, к каким результатам может привести уменьшение количества цветов изображения.

исходное изображение

результат преобразования в новый формат с меньшим количеством цветов

Рис. 11.9. При уменьшении количества цветов появляются дефекты в изображении

Для преобразования файлов из одного формата в другой используются специальные программы — *преобразователи (конверторы) форматов*. Однако большинство графических программ (**CorelDRAW**, **Adobe Illustrator**, **Adobe PhotoShop** и др.) могут читать и создавать файлы различных форматов, т. е. являются преобразователями форматов.

Глава 12. Технические средства КГ (оборудование КГ)

Чаще всего, после того, как изображение возникло на мониторе, пользователь каким-либо образом должен взаимодействовать с ним: модифицировать, передвигать, управлять. Для этого существует ряд устройств, о которых будет рассказано в этой главе.

12.1 Видеоадаптеры

Важной чертой архитектуры персонального компьютера с позиций графики является то, что контроллер видеосистемы (видеоадаптер) расположен рядом с процессором и оперативной памятью и подключен к системной шине через быструю локальную шину. Это дает возможность быстро вести обмен данными между оперативной памятью и видеопамятью (для вывода графических изображений, в особенности в режиме анимации, нужна высочайшая скорость передачи данных). В отличие от этого, в больших компьютерах (мейн-фреймах) данные к дисплеям передавались через интерфейс канала ввода-вывода, который работает намного медленнее системной шины. Большие компьютеры, как правило, работали со многими дисплеями, расположенными на значительном расстоянии.

Первый компьютер IBM PC был оснащен видеоадаптером **MDA {Monochrome Display Adapter).** Видеосистема была предназначена для работы в текстовом режиме — отображались 25 строк по 80 символов в каждой строке.

Через год небольшая фирма Hercules выпустила видеоадаптер **Hercules Graphic Card.** Он поддерживал также и графический черно-белый режим 720x350.

Следующим шагом был видеоадаптер **CGA {Color Graphic Adapter).** Это первая цветная модель для IBM PC. Адаптер CGA позволял работать в цветных текстовом или в графическом режимах. Далее мы будем рассматривать только графические режимы видеоадаптеров. Графических режимов для CGA было два: черно-белый 640x200 и цветной 320x200. В цветном режиме можно было отображать одновременно только четыре цвета (2 бита на пиксел).

В 1984 году появился адаптер **EGA {Enhanced Graphic Adapter).** Это было значительное достижение для персональных компьютеров рассматриваемого типа. Появился графический 16-цветный видеорежим 640x350 пикселов. Цвета можно было выбирать из палитры 64 цветов. В это время начали распространяться компьютерные игры с более или менее качественной графикой и графические программы для работы. Однако шестнадцати цветов явно мало для отображения фотографий, а разрешающая способность недостаточна для графических пакетов типа САПР. Кроме того, видеорежим 640x350 имеет еще один недостаток — разная разрешающая способность по горизонтали и вертикали — "неквадратные пикселы".

В 1987 году появились видеоадаптеры **MCGA {Multi-Color Graphic Array)** и **VGA {Video Graphic Array).** Они обеспечивали уже 256-цветные видеорежимы.

Более популярным стал видеоадаптер VGA. Адаптер VGA имел 256-цветный графический видеорежим с размерами растра 320x200. Цвета можно было выбирать из палитры в 256 тысяч цветов. Это дало возможность полностью удовлетворить потребности отображения черно-белых (в 256 градациях серого) фотографий. Цветные фотографии отображались достаточно качественно, однако 256 цветов мало, поэтому в компьютерных играх и графических пакетах активно использовался дизеринг. Кроме того, режим 320x200 тоже имеет разную разрешающую способность по горизонтали и вертикали. Для мониторов, которые использовались в персональных компьютерах типа IBM PC, необходимо, чтобы количество пикселов по горизонтали и вертикали было в пропорции 4:3. То есть, не 320x200, а 320x240. Такого документированного видеорежима для VGA нет, однако в литературе приведен пример, как создать 256-цветный видеорежим 320x240 на видеосистеме VGA. Можно запрограммировать видеоадаптер, записав в его регистры соответствующие значения, и получить видеорежим "X" (не путать с XGA).

Видеоадаптер VGA также имеет 16-цветовой видеорежим 640x480. Это соответствует "квадратным пикселам". Увеличение разрешающей способности в сравнении с EGA не очень большое, но ощутимое, что дало новый толчок для развития графических программ на персональных компьютерах.

Дальнейшее развитие видеоадаптеров для компьютеров типа IBM PC связано с увеличением разрешающей способности и количества цветов. Можно отметить видеосистему IBM 8514, которая была предназначена для работы с пакетами САПР. Начали появляться видеоадаптеры разных фирм, которые обеспечивали сначала видеорежимы 800x600, а потом и 1024x768 при 16-ти цветах, а также видеорежимы 640x480, 800x600 и более — для 256 цветов. Эти видеоадаптеры стали называть **SuperVGA,** Чуть позже появился видеоадаптер IBM XGA.

Первой достигла глубины цвета в 24-6ит фирма Truevision с видеоадаптером **Targa 24,** что позволило получить на персональных компьютерах IBM PC видеорежим True Color. Такое достижение можно считать началом профессиональной графики на персональных компьютерах этого типа. Там, где раньше использовали графические рабочие станции или персональные компьютеры Apple Macintosh, отныне постепенно переходили на более дешевые компьютеры IBM PC. Одной из таких областей было компьютерное "настольное" издательство.

Сейчас на персональных компьютерах используется много типов видеоадаптеров. Все видеосистемы — растрового типа. Они позволяют устанавливать глубину цвета до 32 битов на пиксел при размерах растра 1600x1200 и больше. Существуют стандарты на видеорежимы, регламентированные VESA {Video Electronic Standards Association).

Параметры отображения обуславливаются не только моделью видеоадаптера, но и объемом установленной видеопамяти. Видеопамять персонального компьютера {VRAM— Video RAM) сохраняет растровое изображение, которое демонстрируется на экране монитора. Изображение на мониторе полностью соответствует текущему содержимому видеопамяти. Видеопамять постоянно сканируется с частотой кадров монитора. Запись новых данных в видеопамять мгновенно изменяет изображение на мониторе. Необходимый объем видеопамяти рассчитывается как площадь растра экрана в пикселах, умноженная на количество битов (или байтов) на пиксел. Например, для 24-битного видеорежима 1024x768 нужно видеопамяти: 24x1024x768 = 18.874.368 битов = 2.25 Мбайт.

В видеоадаптерах первых образцов количество видеопамяти исчислялось килобайтами, например, адаптер CGA имел 16 Кбайт. В современных видеоадаптерах счет идет на мегабайты. Обычно объем видеопамяти кратен степени двойки — 1, 2, 4, 8 Мбайт (в настоящее время — от 32 Мбайт и больше). Наблюдается тенденция увеличения объемов видеопамяти. Основным фактором здесь уже не является глубина цвета. Видеопамять сейчас используется не только как кадровый буфер — она может сохранять текстуры, Z-буфер и т. п.

Адреса, по которым процессор обращается к видеопамяти, находятся в общем адресном пространстве. Например, для многих видеорежимов VGA адрес первого байта видеопамяти равняется A0000. Для некоторых видеорежимов старых образцов используется другой адрес, например, B8000 для CGA 320x200. Современные видеоадаптеры обычно поддерживают видеорежимы, которые использовались ранее. Это делается для обеспечения возможности функционирование старых программ. Каждый видеорежим имеет собственный номер (код) согласно со стандартом VESA.

Кроме физической организации памяти компьютера — в виде одномерного вектора байтов в общем адресном пространстве, необходимо учитывать логическую организацию

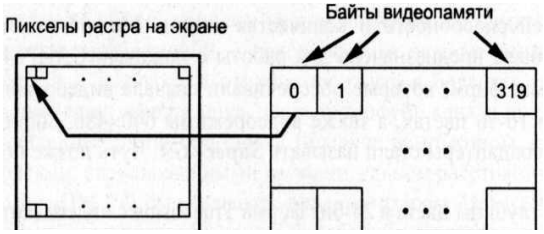


Рис.12.1. Один байт на пиксел для VGA 320x200

видеопамяти. Следует отметить, что названия "физическая" и "логическая" организация могут означать совсем разные вещи для разных уровней рассмотрения. Например, если говорить о физической организации памяти, то она в микросхемах выглядит совсем не как одномерный вектор байтов, а как матрица битов. Логическая организация видеопамяти зависит от видеорежима. В качестве примера на рис. 1.52 приведена логическая организация для видеорежима VGA 256 цветов 320x200 (его код 13h).

Намного сложнее логическая организация видеопамяти для видеорежима

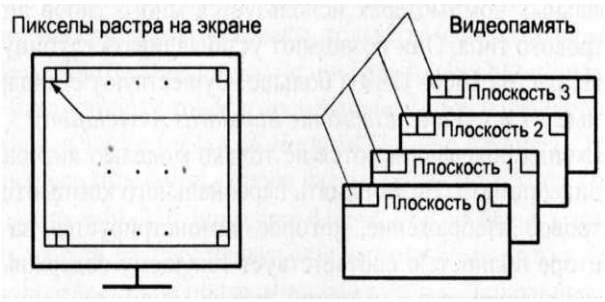


Рис. 12.2. Четыре битовых плоскости для видеорежима 16 цветов 40x480

VGA 16 цветов 640x480 (код 12h), которая показана на рис. 12.2.

Каждая битовая плоскость имеет 80 байтов в одном строке. Плоскости имеют одинаковый адрес в памяти, для доступа к отдельной плоскости необходимо устанавливать индекс плоскости в соответствующем регистре видеоадаптера. Подобный способ организации видеопамяти используется во многих других видеорежимах, он позволяет, например, быстро копировать массивы пикселов.

В этом видеорежиме используются четыре массива байтов памяти. Каждый массив назван битовой плоскостью, для любого пиксела используются одинаковые биты данных разных плоскостей. Для хранения нескольких кадров изображения в некоторых видеорежимах предусматриваются отдельные страницы видеопамяти с одинаковой логической организацией. Тогда можно изменять стартовый адрес видеопамяти — это приводит к сдвигу изображения на экране. Во всех графических видеорежимах стартовый адрес видеопамяти соответствует левому верхнему пикселу на экране. Поэтому координатная система с центром координат (0, 0) в левом верхнем углу растра часто используется в качестве основной (или устанавливается по умолчанию) во многих графических интерфейсах программирования, например, в GDI API Windows.

Обмен данными по системной шине для видеосистемы обеспечивают процессор, видеоадаптер и контроллер локальной шины. До недавнего времени для подключения видеоадаптеров использовалась локальная шина PCI {Peripheral Component Interconnect local bus). Шина PCI предназначена не только для графики, она является стандартом присоединения разнообразнейших устройств, например, модемов, сетевых контроллеров, контроллеров интерфейсов. Эта шина — 32-битная, работает на частоте 33 Мгц, скорость обмена до 132 Мбайт/с.

В настоящее время видеоадаптеры подключаются через локальную шину **AGP {Accelerated Graphics Port).** Разрядность — 64 бит. На частоте 66 Мгц обеспечивала скорость обмена 528 Мбайт/с. Сейчас работает и на более высоких скоростях. Шина AGP была

разработана для повышения скорости обмена данными между видеоадаптером и оперативной памятью по сравнению с возможностями шины PCI. Это позволяет достичь большей частоты кадров при работе 3D-акселераторов. Наличие AGP-порта также приводит к росту быстродействия компьютера в целом благодаря уменьшению нагрузки на шину PCI, что дает возможность более эффективно использовать последнюю для работы с другими внешними устройствами.

Современные видеоадаптеры представляют собой сложные графические устройства. На плате видеоадаптера (сейчас его часто называют видеокартой) располагается мощный специализированный графический процессор **(GPU — Graphic Processor Unit),** который по сложности приближается к центральному процессору. Кроме визуализации кадрового буфера графический процессор видеоадаптера выполняет как относительно простые растровые операции (копирование массивов пикселов, манипуляции с цветами пикселов), так и более сложные. Там, где раньше использовался исключительно центральный процессор, теперь все чаще применяется графический процессор видеоадаптера, например, для выполнения операций графического вывода линий, полигонов. Первые графические процессоры видеоадаптеров выполняли преимущественно операции рисования плоских элементов. Современные графические процессоры выполняют уже много базовых операций 3D-графики, например, поддержку Z-буфера, наложение текстур и т. п. Видеоадаптер выполняет эти операции аппаратно, что позволяет намного ускорить их в сравнении с программной реализацией данных операций центральным процессором. Так появился термин **графические акселераторы.** Быстродействие таких видеоадаптеров часто измеряется количеством графических элементов, которые рисуются за одну секунду. Современные графические акселераторы способны рисовать миллионы треугольников за секунду. Этим "интеллектуальность" видеоадаптеров не ограничивается. Недавно появились модели, которые, кроме относительно простых неизменных базовых операций, способны сами выполнять небольшие программы, которые могут составлять пользователи. Эти программы называются **"шейдерами"** (shaders). Такие возможности графических акселераторов сейчас активно используются разработчиками компьютерных игр.

На рис.12.3 представим общую структуру современного видеоадаптера.

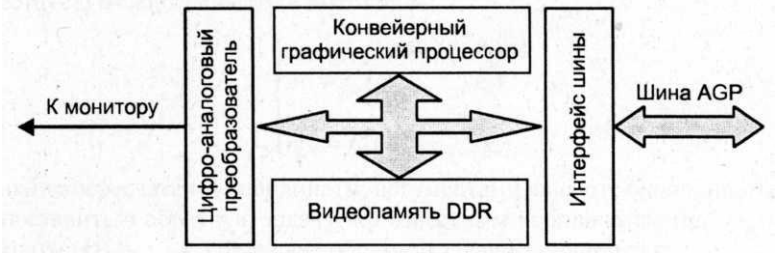


Рис. 12.3. Общая структура видеоадаптера

Номенклатура видеоадаптеров для персональных компьютеров широка. Несколько примеров: видеоадаптеры Matrox (качественная двумерная графика), ATI Radeon, NVidia (профессиональные и игровые 3D-акселераторы).

Использование программистами графических возможностей видеосистемы может осуществляться по-разному. Во-первых, простейшие операции, такие, как определение графического видеорежима, вывод пиксела на экран и некоторые другие, поддерживаются BIOS. Во-вторых, можно использовать функции операционной системы. Разные операционные системы могут предоставлять разные возможности. Например, в MS-DOS графических функций почти не было, однако программисту был разрешен свободный доступ ко всем аппаратным ресурсам компьютера. В быстродействующих графических программах часто использовался непосредственный доступ к видеопамяти. В отличие от этого, операционная система Windows запрещает прикладным программам непосредственный доступ к аппаратным ресурсам, однако можно применять несколько сотен графических функций операционной системы — интерфейс GDI API. В-третьих,

можно использовать специализированные графические интерфейсы, которые поддерживают аппаратные возможности современных графических процессоров.

Один из известнейших графических интерфейсов — OpenGL. Этот интерфейс в виде библиотеки графических функций был разработан Silicon Graphics и поддерживается многими операционными системами (в том числе Windows) и производителями графических акселераторов. Интерфейс OpenGL для графического отображения использует взаимодействие типа клиент-сервер.

Другим известным графическим интерфейсом является DirectX. Этот интерфейс разработан Microsoft и предназначен только для ОС Windows.

12.2 Манипуляторы

Первые персональные компьютеры располагали для ввода информации и управления работой компьютера единственным устройством — клавиатурой. Для реализации более простого управления нужно было создать дополнительную, параллельную клавиатуре, систему. Эту задачу решил Дуглас Энджелбарт из Стенфордского исследовательского института (США). В 1970 году им был получен патент на манипулятор. Вначале такой манипулятор назывался «индикатор позиции X-Y». Он явился прообразом современной мыши. Позже были созданы и другие типы манипуляторов — трекболы и джойстики.

Мышь

Мышь является важнейшим средством ввода графической информации в компьютер. В современных программных продуктах, имеющих сложную графическую оболочку, мышь (см. рис. 12.4) является основным инструментом управления программой.

В наиболее распространенных конструкциях мыши в качестве элемента, следящего за ее движением, используется шарик, сделанный из плотного резинопластика. В процессе перемещения мыши по поверхности шарик вращается и передает вращение двум металлическим валикам, которые также вращаются: один вдоль направления движения мыши, а другой — поперек. Вращение валиков регистрируется специальными устройствами, позволяющими выделять направления вдоль оси X и вдоль оси Y. Таким образом, в каждый момент времени положение мыши фиксируется с помощью координат X и Y в условной координатной плоскости. Эти координаты передаются в компьютер, после чего электроника компьютера устанавливает курсор на экране в соответствии с этими координатами. Для обеспечения оптимального функционирования мышь необходимо перемещать по ровной поверхности — специальному *коврику* (mouse pad). При этом указатель мыши передвигается по экрану синхронно с движением мыши по коврику. Устройством ввода мыши являются кнопки (клавиши). Большинство манипуляторов этого типа имеют две кнопки (рис. 12.4). Существуют также 3-кнопочные мыши и мыши, имеющие большее количество кнопок (Бывают также и мыши, имеющие всего одну кнопку, и при этом неплохо справляющиеся со своими функциональными обязанностями. — *Примеч. Ред.*).

Одной из важных характеристик мыши является ее разрешение, измеряемое в dpi (*dpi* — dots per inch — количество точек на дюйм. — *Примеч. Ред.*). Эта характеристика определяет минимальное перемещение, которое способен почувствовать контроллер мыши. Чем больше разрешение, тем точнее позиционируется мышь, тем с более мелкими объектами можно работать. Нормальное разрешение мыши лежит в диапазоне от 300 до 900 dpi. В усовершенствованных мышах используют переменный баллистический эффект скорости, заключающийся в том, что при небольших перемещениях скорость смещения курсора небольшая, а при значительных перемещениях существенно увеличивается. Это позволяет эффективнее работать в графических пакетах, когда приходится обрабатывать мелкие детали.

Рис. 12.4. Манипулятор мышь

В настоящее время разработано несколько разновидностей бесхвостых мышей, то есть не связанных кабелем с компьютером. Бесконтактные мыши используют инфракрасную связь, аналогично пультам дистанционного управления (требует визуального контакта с приемником), либо радиосвязь.

Художнику-дизайнеру удобнее работать с профессиональным вариантом мыши, называемой 4D-MOUSE (например, A4TECH 4-Way Scroll). Ее вертикальное и горизонтальное колесики удобны для перемещений по осям X-Y, боковая кнопка воспроизводит режим экранной лупы, верхняя третья кнопка позволяет задавать режим выхода из приложения, закреплять за кнопкой комбинацию альтернативных клавиш любой команды и т. д. Более подробную информацию об A4TECH можно получить по адресу www.a4tech.com.tw.

Трекболы

Трекбол — это устройство ввода информации, которое можно представить в виде перевернутой мыши с шариком большого размера (рис. 12.5). Принцип действия и способ передачи данных трекбола такой же, как и мыши. Наибольшее распространение получил оптико-механический принцип регистрации положения шарика. Трекбол чаще всего используют в компактных компьютерах типа Laptop или Notebook. Подключение трекбола, как правило, осуществляется через последовательный порт.

Рис. 12.5. Манипулятор трекбол

Джойстики

Джойстик является координатным устройством ввода информации и наиболее часто применяется в области компьютерных игр и компьютерных тренажеров. В последнем случае обычно используются аналоговые джойстики, тогда как в игровых компьютерах — цифровые. Аналоговые джойстики обеспечивают более точное управление, что очень важно для программных приложений, в которых объекты должны точно позиционироваться. Для удобства работы конструкция джойстика должна быть достаточно прочной и устойчивой. Джойстик подключают к внешнему разъему карты расширения, имеющей соответствующий порт.

Рис. 12.6. Манипулятор джойстик

Дигитайзер

Дигитайзер или планшет, как его тоже называют, состоит из двух основных элементов: основания и курсора, двигающегося по его поверхности. Это устройство, изначально предназначенное для оцифровки изображений. При нажатии на кнопку курсора его местоположение на поверхности планшета фиксируется основанием, а его координаты передаются в компьютер.

Рис. 12.7. Манипулятор дигитайзер

Сейчас дигитайзер также часто ассоциируют с управлением командами в «Автокаде» и аналогичных системах при помощи накладных меню. Команды в меню расположены на разных местах на поверхности дигитайзера. При выборе курсором одной из команд специальный программный драйвер интерпретирует координаты указанного места, посылая соответствующую команду на выполнение. Не последнее место занимает применение планшетов для создания на компьютере рисунков и набросков. Художник создает изображение на экране, но его рука водит пером по планшету. Наконец, дигитайзер можно использовать просто как аналог манипулятора «мышь».

Перчатки и виртуальный шлем

Специальные перчатки чаще всего используют вкупе с виртуальным шлемом. Они позволяют «взять» некий виртуальный камень (или другой предмет), который пользователь видит перед собой в шлеме, повернуть его, не выпуская из рук, и даже почувствовать его неправильную, угловатую или, наоборот, обтекаемую форму. Специальные датчики, закрепленные на перчатке, считывают положение пальцев пользователя, компьютер обрабатывает эти сигналы, преобразует их в картинку для виртуального шлема, а также посылает управляющие сигналы выполненным из специального материала подушечкам перчатки, которые принимают ту или иную форму и давят с разной силой на различные участки ладони пользователя, создавая иллюзию сжатия предмета.

Рис. 12.8. Виртуальный шлм

В виртуальном шлеме имеется два экрана, каждый из которых выводит свое изображение для левого и правого глаза пользователя. Ни левый, ни правый глаз не видит, что происходит на соседнем экране, так как этому мешает перегородка виртуального шлема. В мозгу смотрящего принятые изображения складываются в единую трехмерную картину.

Штурвал и педали

Интересными разновидностями джойстиков можно считать штурвал и педали. Такие вещи больше всего используются в симуляторах различных полетов.

Рис. 12.9 Педали и штурвал

Штурвал представляет из себя контроллер, имеющий 3 оси (тангаж, крен, газ), а также целую россыпь кнопок: 3 кнопки, два 2-позиционных качающихся переключателя, 4-позиционный хэт, а также два рычажка (закрылки и шасси), один из которых 1-позиционный, а другой — 2-позиционный.

Особенностью педалей является наличие дополнительной функции педальных тормозов: помимо скользящего движения, характерного для управления рулем поворота в воздухе, каждая педальная площадка имеет качающееся движение, для имитации подтормаживания одной или обеих стоек шасси, используемого при рулении на земле.

12.3 Оборудование мультимедиа

Что такое мультимедиа? Мультимедиа — это комплексное представление информации — вывод данных в текстовом, графическом, видео-, аудио- и мультипликационном видах.

Мультимедийный набор- это:

- звуковая карта;
- звуковые колонки;
- микрофон;
- CD-ROM;
- кабели

Рис. 12.10. Оборудование для мультимедиа

И многое, многое другое:

- чувствительный экран (touch-панель)
- световое перо
- контактный, акустический, лазерный щуп (перо)
- считыватель-обработчик штрих-кодов
- цифровая камера
- dvd-проигрыватель
- спутниковая плата
- программируемая клавиатура
- ультразвуковая мышка («Сова»)

Текст

Под текстом понимается любой набор символов из той или иной кодовой страницы. Текст использовался в компьютерах еще задолго до того, как появилось само слово «мультимедиа». Но и сейчас, и в будущем текст останется важным компонентом мультимедиа, так как он является простым, но чрезвычайно эффективным средством для представления и передачи информации.

Текст может быть представлен различными *кодовыми страницами*. Кодовая страница — это взаимно однозначное соответствие между изображением символа и его порядковым номером (кодом) в кодовой таблице.

Первоначально кодовые страницы состояли 128 символов, в число которых входили только строчные и прописные латинские символы, цифры, управляющие символы и символы псевдографики. По мере распространения персональных компьютеров стали появлятся кодовые страницы с символами национальных алфавитов (в том числе и с символами кириллицы). Разные кодовые таблицы имеют свои названия. Старейшая кодовая страница с русскими символами — KOI8-R, которая используется в операционных системах Unix. В ОС DOS использовалась кодировка cp866; в ОС Windows

используется кодировка Windows-1251. Последняя из разрабатываемых кодовых страниц — Unicode (UTF-8), которая содержит 64 тыс. символов всех национальных алфавитов, математические, химические и другие знаки; Unicode в той или иной степени поддерживется многими современными операционными системами.

Графика

По принципу представления графика далится на растровую и векторную. Изображение в растровой графике строится как набор элементарных точек, раскрашенных тем или иным цветом. Векторная графика строится по правилам векторной алгебры из точек, линий, поверхностей.

Растровая графика характеризуется следующими параметрами:

- размер картинки (измеряется в пикселах, миллиметрах, дюймах и т. д.);
- разрешение — количество точек на единицу (обычно дюйм);
- количество передаваемых цветов или глубина цвета. Чем большее количество информации отводится для запоминания каждой отдельной точки, тем красочнее картинка и больше размер файла. Стандартные значения:
- 2 цвета (1 бит на точку);
- 16 цветов (4 бита на точку);
- 256 цветов (8 бит на точку);
- 16777216 цветов (24 бита на точку);
- 4294967296 цветов (32 бита на точку);
- формат записи (BMP, PCX, GIF, TIF, JPG, TGA и др.) — способы хранения графической информации с элементами (или без них) сжатия.

Векторную графика подразделяется на двумерную и трехмерную. Она имеет характеристики аналогичные математическим, а именно: координаты (декартовы, сферические, цилиндрические и др.), системы отсчета, размеры... Векторная графика может быть преобразована в растровую путем получения плоского изображения одной из проекций. Обратное преобразование невозможно или крайне сложно.

Видео

Видео-изображение — это последовательность растровых картинок, сменяющихся с большой скоростью аналогично принципу, используемому в кинематографе или телевидении. С помощью специальных аппаратных средств обычные видеозаписи переводятся в компьютерный формат. Это дает возможность производить нелинейный монтаж и применять к изображениям различные компьютерные эффекты. После этого видео снова может быть выведено на пленку.

Компьютерное видео характеризуется следующими параметрами:

- количество кадров в секунду (15, 24, 25...);
- поток данных (килобайт/с);
- формат файла (avi, mov...);
- способ сжатия (Microsoft Video for Windows, MPEG, MPEG-I, MPEG-2, Moution JPEG).

Видео кодируется двумя основными способами: сжимается каждый кадр (картинка) в отдельности и составляется видео фильм либо создаются опорные кадры, а затем записываются изменения между этими опорными кадрами.

Компьютерное видео создается редакторами 3D анимации, монтажными пакетами, оцифровыванием видео-изображения.

Анимация

Отличается от видео тем, что получается чисто компьютерным способом. Может быть записана в тех же форматах, что и видео, и выведена на видеопленку. Анимация делится на двумерную и трехмерную. Анимация создается редакторами двумерной и трехмерной графики, сканированием и оцифровыванием изображения.

Цифровой звук

Аналоговый звуковой сигнал непрерывен по амплитуде и времени. Простейшая звуковая волна представляется обычно напряжением или током, изменяющимся во времени по синусоидальному закону. Амплитуда соответствует громкости звука, частота — высоте звука. Для представления в цифровом виде аналоговый сигнал перекодируют, запоминая параметры звука через определенные промежутки времени в структуре данных определенного размера.

Качество записи характеризуется: частотой дискретизации (Гц), размером структуры данных (бит), количеством каналов (стерео, моно, квадро), обобщающим параметром — потоком (бит/с).

Наиболее часто звук записывается в формате PCM (Pulse Code Modulation). Такие звуковые файлы еще называют WAV-файлами. Основные частоты дискретизации: 8, 11, 22, 44 кГц, основные размеры: 8, 16, 32, 64 бит. Сочетая эти параметры различным образом, можно широко варьировать как качество звука, так и размеры получаемых файлов.

Для воспроизведения цифрового звука применяют обратное преобразование в аналоговый сигнал из цифрового или синтез аналогового сигнала на основе цифровой записи. Для уменьшения размера звукового файла используют специальные форматы записи звука (DPCM, ADPCM) с дополнительной компрессией. В последнее время огромную популярность получил звук в формате MP3 (MPEG 1 Layer 3). Это схема сильного сжатия аудиоинформации с потерями качества звучания. Популярность этого формата объясняется тем, что при относительно высоком качестве звучания размер звукового фрагмента для наиболее часто используемого потока 128 килобит/с на порядок ниже исходного звукового фрагмента. Однако качество Audio-CD при записи в MP3 достигается на гораздо более высоких потоках, и лишь плохая воспроизводящая аппаратура не позволяет заметить артефактов MP3 на потоках от 128 килобит/с и ниже. Основная идея, на которой основана данная методика сжатия — отказ от кодирования тонких деталей звучания, лежащих вне пределов возможностей человеческого слуха. В общем случае объем и степень ощутимости потерь определяются, с одной стороны, потоком, а с другой — психоакустической моделью возможностей слуха, использованной в каждом конкретном кодере.

Запись мелодий в формате MIDI

Для записи звучания инструментальных композиций используется формат MIDI, позволяющий описывать звучание того или иного инструмента с помощью нотной грамоты и заранее заданных характеристик этого инструмента. Плюсом является то, что выходной файл получается небольшим: десятки, редко — сотни килобайт. Большой недостаток в использовании этой методики заключается в том, что нигде заранее не было оговорено, как должен звучать, к примеру, орган или клавесин. Поэтому производители музыкальных плат настраивали звучание того или иного инструмента так, как считали нужным. Поэтому одна и та же MIDI-мелодия может звучать абсолютно по-разному на звуковых платах разных производителей.

12.4 Мониторы

Монитор компьютера (рис. 12.11) предназначен для вывода на экран текстовой и графической информации. Это практически единственный элемент компьютера, который нельзя в дальнейшем модернизировать. Он покупается для долговременного использования. От его качества и безопасности напрямую зависит ваше здоровье, прежде всего зрение. В то же время монитор — один из самых «консервативных» компонентов компьютерной системы. Производительность процессоров, емкость винчестеров и банков оперативной памяти увеличиваются в несколько раз в течение года, но монитор практически не меняет своего облика, поэтому к покупке монитора для своего домашнего ПК следует подходить очень серьезно. Ибо, подключив к высокопроизводительной

системе маленький и некачественный монитор, вы лишитесь всей радости от приобретения нового компьютера. Что должен уметь монитор?

Прежде всего он должен нормально работать на разрешении 1280 x 1024 при частоте вертикальной развертки хотя бы 85 Гц. Хороший монитор должен поддерживать частоту обновления как минимум 85 Гц. Лучше, конечно, больше — 100-120 Гц, так как многие **на** частоте 85 Гц все еще ощущают мерцание. Кроме всего прочего, запас частоты говорит о классе монитора — у дорогих и качественных моделей частотные характеристики лучше.
Второй немаловажный фактор — это размер точки или ширина ячейки апертурной решетки. Он должен быть не более 0,24-0,25 мм.

Самый распространенный на сегодня тип дисплеев — это CRT (Cathode Ray Tube), или ЭЛТ-мониторы. Свое название они получили от их основного компонента — электронно-лучевой трубки (ЭЛТ), которая может иметь различные варианты конструкции.

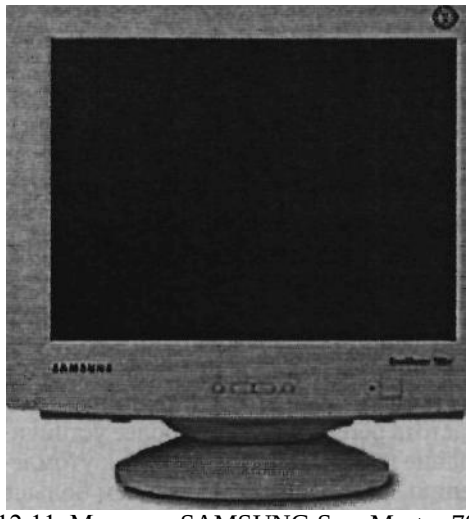


Рис. 12.11. Монитор SAMSUNG SyncMaster 700IFT

В некоторых моделях ЭЛТ используется *апертурная решетка* (aperture grill). Этот тип маски применяется в трубках Trinitron от Sony, Diamondtron от Mitsubishi и других. Существует также вариант ЭЛТ еще с одним типом маски — щелевой.

Практически в каждой развитой стране существует достаточное количество национальных стандартов, регламентирующих уровень излучений монитора. Но особую популярность во всем мире завоевали стандарты, разработанные в Швеции, — MPR II и TCO.

Характеристики мониторов

В настоящее время существует большое разнообразие типов мониторов. Их можно охарактеризовать следующими основными параметрами.

Тип экрана:

электронно-лучевая трубка или ЭЛТ (CRT);
жидкокристаллический дисплей (ЖКД);
плазменный дисплей.

Размер по диагонали (обычно от 14'до 2 Г).

Цветность (цветные или монохромные).

Размер зерна (обычно от 0,24 до 0,31 мм).

Частота кадров (обычно от 50 до 100 Гц),

Видеосигнал (цифровой или аналоговый).

Прочие характеристики (функции управления растром, система энергосбережения, защита от излучения, вес, габариты, потребляемая мощность).

Размер монитора связан *с разрешением. Разрешение* выражается в количестве точек (пикселов) по горизонтали и по вертикали отображаемого изображения. Например, если говорят, что монитор имеет разрешение 640 x 480 — это означает, что на экране можно

целиком разместить изображение, состоящее из 640 x 480 - 307 200 точек. Возможность использования конкретного разрешения зависит от различных факторов, среди которых в первую очередь следует отметить размер по диагонали и размер точки самого монитора, характеристики видеокарты и объем доступной видеопамяти, которая ограничивает число отображаемых цветов. *Максимальная разрешающая способность* — одна из основных характеристик монитора. Чем больше разрешение, тем больше информации умещается на экране.

Важной характеристикой также является *частота регенерации экрана.* Этот параметр определяет частоту обновления (перерисовывания) изображения на экране. Частота регенерации измеряется в Гц (Герцах, Hz). У мониторов на основе ЭЛТ время свечения люминофорных элементов очень мало, поэтому электронный луч должен проходить через каждый элемент люминофорного слоя достаточно часто, чтобы не было заметно мерцания изображения. Если частота такого обхода экрана становится меньше 70 Гц, то инерционности зрения будет недостаточно для того, чтобы изображение не мерцало. Чем выше частота регенерации, тем более устойчивым выглядит изображение на экране. Мерцание изображения приводит к утомлению глаз, головным болям и даже к ухудшению зрения. Заметим, что чем больше экран монитора, тем более заметно мерцание, особенно периферийным зрением, так как угол обзора изображения увеличивается. Значение частоты регенерации зависит от используемого разрешения, от электрических параметров монитора и от возможностей видеоадаптера. Минимально безопасной частотой кадров считается 75 Гц. *Мониторы бывают с чересстрочной разверткой и построчной разверткой. Чересстрочные и построчные развертки* — два способа регенерации изображения на экране монитора. Монитор с чересстрочной разверткой регенерирует изображение на экране за два прохода электронного луча. Первый проход воспроизводит нечетные строки, а второй — четные. Монитор с построчной разверткой воспроизводит полное изображение на экране за один проход электронного луча. Мониторы с построчной разверткой обладают лучшими характеристиками, так как они воспроизводят изображение на экране быстрее и без мерцания. Они также имеют более резкие и четкие изображения. Все мониторы высокого качества отображают изображения во всех режимах разрешения с построчной разверткой, Мониторы, имеющие «штатные» режимы с чересстрочной разверткой, ни одной из ведущих фирм-производителей не выпускаются.

Рекомендация: в первом приближении для графических работ подойдет монитор с размером экрана от 17" и выше, частотой кадров 85 Гц, разрешением не ниже 1024x768 dpi.

Аналоговые мониторы

Электронно-лучевая трубка мониторов данного типа управляется аналоговыми сигналами, поступающими от видеоадаптера. Принцип работы электронно-лучевой трубки монитора такой же, как у телевизионной трубки. Аналоговые мониторы способны поддерживать разрешение стандарта VGA (640 x 480 пикселов) и выше.

Все современные аналоговые мониторы условно можно разделить на следующие типы:

традиционные с фиксированной частотой развертки,

с несколькими фиксированными частотами и многочастотные *(мулътичастотные).*

Эти мониторы обладают способностью настраиваться на произвольные значения частот синхронизации из некоторого заданного диапазона, например30-64 кГц для строчной и 50-100 Гц для кадровой развертки. Разработчиком мониторов данного типа является фирма NEC. В названии таких мониторов присутствует слово Multisync. Эти мониторы относятся к наиболее распространенному типу мониторов с электронно-лучевой трубкой (CRT).

Жидкокристаллические дисплеи

Экран *жидкокристаллического дисплея* (ЖКД) состоит из двух стеклянных пластин, между которыми находится масса, содержащая жидкие кристаллы. Кристаллы

изменяют свои оптические свойства в зависимости от прилагаемого электрического заряда. Жидкие кристаллы сами не светятся, поэтому ЖКД нуждаются в подсветке или во внешнем освещении. Основным достоинством ЖКД являются их габариты (экран плоский, толщиной 5-6 см). К недостаткам можно отнести низкое быстродействие при изменении изображения на экране, что особенно заметно при перемещении курсора мыши, а также зависимость резкости и яркости изображения от угла зрения.

Газоплазменные мониторы

Газоплазменные мониторы состоят из двух пластин, между которыми находится газовая смесь, светящаяся под воздействием электрических импульсов. Такие мониторы не имеют недостатков, присущих ЖКД, однако их нельзя использовать в переносных компьютерах с аккумуляторным и батарейным питанием, так как они потребляют большой ток.

Пример. Модель монитора SAMSUNG SyncMaster 700IFT

Модель SyncMaster 700IFT предназначена для решения профессиональных задач. В ней используется ЭЛТ DynaFlat с абсолютно плоским экраном. Эта модель отвечает требованиям стандарта безопасности TCO 99. Ее электроника обеспечивает работу монитора с максимальным экранным разрешением 1600 x 1200. Помимо стандартного 15-штырькового VGA-входа имеются разъемы BNC. Дизайн дисплея, в общем, классический, со скругленными углами сзади корпуса. Элементы управления представлены квадратной кнопкой включения, двумя колесиками под нижним торцом спереди для настройки яркости и контрастности и выезжающей панелью с клавишами для операций с экранным меню. Одна из семи кнопок на панели служит для получения информации о рабочем режиме. Четыре клавиши предназначены для навигации по экранному меню. Последние две кнопки — для вызова меню и выхода из него. Среди функций экранного меню можно отметить регулирование линейности, фокусировку и муар по горизонтали-вертикали. Среди возможных языков экранного меню есть русский. В комплекте с монитором поставляется компакт-диск с программой Colorific.

LCD-панель (жидкокристаллическая индикаторная панель), как правило, работает во взаимодействии с эпидиаскопическим проектором или диапроектором. Панель помещается поверх проектора, который проецирует компьютерное изображение на экран. В последние годы качество изображения LCD-панелей значительно улучшилось. Панели более старых моделей не могли обеспечить резкое и чистое цветное изображение. Среди наиболее известных производителей жидкокристаллических панелей можно выделить компании 3M и Polaroid.

Проектор, Если вы создаете мультимедиа-презентацию и не хотите быть связанным возможностями эпидиаскопического проектора, то есть вариант с использованием проектора на жидких кристаллах (ЖК-проектор). Это самодостаточное устройство, способное проецировать не только видео-, но и аудиоданные из компьютера. ЖК-проекторы обычно стоят от 3000 до 8000$. Производителями таких проекторов являются фирмы Focus Systems и Proxima.

Внешний декодер. Если вам необходимо провести презентацию на телевизионном мониторе, то стоимость подключения будет намного ниже, чем покупка LCD-панели или проектора. Целый ряд фирм-производителей изготавливают декодирующие сканирующие преобразователи (encoding scan converters), которые преобразуют сигнал вашего компьютера в сигнал, подходящий для чтения обычным телевизором. Более подробную информацию можно найти в рекламных каталогах торгующих компьютерным оборудованием фирм.

Видеокарта

Видеокарта (графическая карта, видеоадаптер) реализует вывод информации на монитор. От ее качества зависят:

скорость обработки информации;

четкость изображения и размеры;

цветность (количество воспроизводимых цветов) рабочего поля экрана.

В зависимости от количества поддерживаемых цветовых оттенков различают следующие режимы работы видеоадаптеров:

16 цветов;

256 цветов;

High Color (16 бит);

True Color (24 бит);

True Color (32 бит).

Основными параметрами видеоадаптеров являются величина *разрешения экрана* и *тип развертки монитора,* которые они способны поддерживать. В настоящее время преимущественно используются видеоадаптеры стандарта SVGA и другие более современные, предложенные фирмой IBM (включая Enhanced VGA, XGA и VESA 1.2).

По выполняемым функциям *видеокарта* представляет собой небольшой компьютер, собранный на одной плате (рис. 12.12). У него, как и у основного компьютера, есть свой кварцевый генератор рабочей частоты, собственный BIOS (в последних моделях — чаще всего перепрограммируемый), центральный процессор, или чипсет, составляющий основу видеокарты, память и RAMDAC — конвертер цифрового сигнала, вырабатываемого картой, в аналоговый сигнал, подаваемый на монитор. На многих видеокартах имеются дополнительные разъемы, которые используются для размещения дополнительной памяти, тюнера, MPEG- или 3D-ускорителя или еще какого-нибудь прибора.

Основные компоненты видеокарты изображены на рис. 12.12. Цифрами обозначены: 1 — видеопроцессор. Он настолько интенсивно и жарко работает, что закрыт радиатором для охлаждения. А на самых современных графических ускорителях приходится использовать даже вентиляторы; 2 — набор микросхем видеопамяти; 3 — микросхема ПЗУ, в котором зашита программа управления видеопроцессором; 4 — специальный разъем для передачи информации из ОЗУ в видеопамять; 5 — разъем для передачи информации из процессора в монитор.

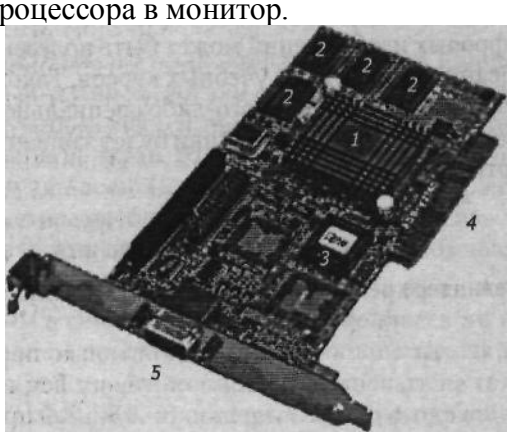


Рис.12.12. Внешний вид и основные компоненты видеокарты

При комплектовании компьютера в первую очередь следует обратить внимание на следующие важные характеристики видеокарты:

чипсет;

RAMDAC;

частота регенерации;

разрешение экрана;

память видеокарты.

Чипсет (chipset) — набор микросхем, фактически главный компонент видеокарты и от того, насколько хорошо он реализован, зависят и ценность, и возможности вашей карты. Одной из главных характеристик чипсета является его *пропускная способность* (bandwidth). Сегодня на рынке комплектующих можно встретить 64-битные и 128-битные видеокарты. Естественно, что последние более быстрые и более дорогие.

RAMDAC (Random Access Memory Digital to Analog Converter) — частота преобразования цифровых данных в аналоговые. Нормальной частотой сегодня следует считать 250 МГц, то есть произведение системой 250 миллионов операций в секунду. Что это означает на практике? Каждый раз, когда генерируется картинка, она обрабатывается RAMDAC и выводится на монитор. Даже если картинка не меняется, все равно изображение на мониторе нужно постоянно и с очень большой скоростью обновлять. Поэтому чем выше частота RAMDAC, тем изображение качественнее, то есть более четкое и устойчивое, а производительность системы в целом — более высокая. От значения частоты RAMDAC зависит также частота регенерации. Нужную частоту RAMDAC можно вычислить по следующей формуле:

Частота = вертикальное разрешение х горизонтальное разрешение х частоту регенерации х 1,32 (коэффициент, связанный с той временной задержкой, с которой лучи доходят от катода до поверхности люминофора).

Например, чтобы получить разрешение 1600 х 1200 с частотой регенерации экрана 85 МГц, нам потребуется RAMDAC с частотой 215424 000 Гц, или 215,4 МГц.

Частота регенерации (refresh rate) показывает, сколько раз в секунду будет обновляться картинка на экране дисплея. Чем выше его значение — тем лучше. От него зависит, как сильно будет мерцать монитор и, соответственно, насколько будут уставать ваши глаза при длительном сидении за компьютером. Минимальная приемлемая частота для 14"-15" монитора при максимально возможном разрешении 75 Гц. Лучше, если монитор способен работать при частоте 80-85 Гц. Видеокарта должна давать солидный запас прочности, то есть при разрешении 1024 х 768 она должна обеспечивать 120 Гц (если вы не собираетесь использовать при этом монитор размером больше 21 дюйма по диагонали). Но лучше еще больше — 150-170 Гц.

Разрешение экрана (resolution) выше 1024 х 768 используют сегодня только профессионалы, занимающиеся графикой, и владельцы больших мониторов. Тем не менее хорошо, если видеокарта в силах выдерживать разрешение 1600 х 1200 при нормальной (75 Гц) частоте регенерации. Правило: больше — лучше.

Для обычной, двухмерной графики достаточно памяти *видеокарты* 8 Мбайт. Это позволит обеспечить максимально возможное разрешение 1600 х 1200 при 24-битовой глубине (16,7 млн. цветов). Однако в последних моделях видеокарт со встроенными 3D-ускорителями используется дополнительная память — как минимум еще 8 Мбайт. На картах следующего поколения установлены уже 32 Мбайт памяти, и это, видимо, не предел целесообразности. Память на современных видеокартах встречается в основном двух видов — SDRAM и SGRAM, из которых последняя выполнена по более сложной технологии и поэтому стоит дороже, хотя характеристики и быстродействие у них очень схожие.

Все видеоданные должны быть доставлены к месту назначения по широкой (64-128-разрядной) и быстрой магистрали. За пропускную способность шин, на которых работают видеокарты, отвечают спецификации ISA, PCI, AGP.

Функции графического ускорителя

Графический ускоритель нужен для ускорения прорисовки экрана. Это связано с тем, что при работе с изображениями (особенно в векторной графике) перерисовка занимает значительную часть ресурсов компьютера. Когда речь идет о работе с двухмерной графикой, то практически любые графические карты последнего поколения, а также и многие карты, созданные уже довольно давно, демонстрируют превосходные эксплуатационные показатели и прекрасную работу. Карты типа ATI Rage Pro, Intel i740 и nVidia Riva 128 — это вполне приемлемый выбор для тех, кто планирует пользоваться Adobe Photoshop или CorelDRAW. Данный вывод обусловлен тем, что все эти карты имеют 8-32 Мбайт оперативной видеопамяти, которой вполне достаточно для обработки большого количества цветов с высокой разрешающей способностью. Можно посоветовать самое последнее поколение графических карт и ускорителей, которое включает 3Dfx Voodoo Banshee и VoodooS, nVidia Riva

TNT и TNT2, ATI Rage 128, Matrox G200 и G400 или S3 Savage и Savage 4. Они будут также нуждаться по крайней мере в 16 Мбайт оперативной видеопамяти, чтобы обрабатывать текстурные файлы с нормальным уровнем качества и достаточно быстро. У каждого компьютерного фотохудожника могут быть свои собственные предпочтения. Например, чип SDfx's 3-D-only Voodoo2 имеет ярко выраженную специализацию для работы с трехмерной графикой высокого уровня. Поэтому его могут использовать те пользователи, которые работают с AutoCAD 2000 и 3D-MAX.

Выбор видеокарты под монитор

Для нового поколения игр необходимы видеокарты, чипы которых поддерживают стандарты 3D-ускорения, На данный момент с этой задачей лучше других, на наш взгляд, справляются видеокарты на чипах RivaTNT/TNT2/TNT2 Vanta. Для систем средней производительности они имеют оптимальный объем видеопамяти — 16 Мбайт. Кроме того, упомянутые карты прекрасно справляются с профессиональной 20-графикой и являются идеальным решением для офисных и домашних систем. Если вы купите такую видеокарту сейчас, вам еще очень долго не придется задумываться об очередной модернизации.

Для профессионалов компьютерной графики, дизайнеров и художников лучшим выбором может стать только Matrox — модели G200 или более новые — G400.
Пользователям, не нуждающимся в мощных игровых SD-ускорителях и сверхкачественной обработке двухмерной графики, можно порекомендовать поставить видеокарту Intel 740/752 или одну из дешевых моделей ATI.

Абсолютное большинство современных видеоадаптеров выполняется для AGP-слота. Поэтому, покупая PCI-видеокарту, убедитесь, что в вашем компьютере есть свободное гнездо шины PCJ.

Сегодня практически все видеокарты выпускаются с 4 Мбайт (и более) видеопамяти. Они поддерживают миллионы цветов почти для всех размеров мониторов. Тем не менее надо быть готовым, что вам может попасться видеокарта с меньшим объемом видеопамяти. В этом случае полезно иметь представление о количестве цветов, обеспечиваемых такими видеокартами для различных мониторов:

- 1 Мбайт видеопамяти предоставляет миллионы цветов на 14-дюймовом мониторе и тысячи — на 15-дюймовом;
- 2 Мбайт видеопамяти дают миллионы цветов на 14- и 17-дюймовых мониторах, тысячи цветов — на 19-дюймовом мониторе или на мониторе размером в 21 дюйм.

12.5 Видеобластеры

Видеобластер (видеокарта ввода-вывода) — это устройство, предназначенное для ввода в персональный компьютер видеоинформации, а также для вывода из компьютера видеоинформации на внешние устройства (например, видеомагнитофон). Видеобластер не заменяет видеоадаптер, служащий для вывода информации на экран монитора, а устанавливается дополнительно.

Частным случаем видеобластеров являются *TV-конвертеры.* Наиболее простые и дешевые TV-конвертеры, или платы видеовывода, подключаются между видеоадаптером и монитором. Они стоят примерно $300 и позволяют вести запись на обычный бытовой видеомагнитофон типа VHS. Более профессиональные устройства (они стоят дороже) вставляются в компьютер в виде отдельной платы. Они рассчитаны на подключение профессионального «видика» с качеством Super-VHS и могут управлять видеомагнитофоном.

12.6 Периферия

Периферийные устройства служат для расширения функциональных возможностей персонального компьютера, удобства управления им и представления информации в различных формах в процессе ее обработки, хранения и отображения. К периферийным устройствам относятся: принтеры, модемы, сканеры, CD-ROM, магнитооптические диски,

стримеры, графические планшеты, плоттеры, устройства мультимедиа (видеобластеры, звуковые платы и акустические системы), трекболы, джойстики и другие желательные, но не обязательные устройства. Подсоединение периферийных устройств к компьютеру производится через устройства *сопряжения* (адаптеры), на которых реализованы стандартные или специальные интерфейсы. Обычно адаптеры выполняются в виде отдельных плат ввода/вывода, вставляемых в разъемы расширения на системной плате. Интерфейс определяет тип и вид соединителя (вилка или розетка, male или female), протоколы обмена, уровни и длительности электрических сигналов.

Последовательный и *параллельный интерфейсы* называют также *портами ввода-вывода.* Последовательные порты используются для подключения мыши, удаленного принтера, внешнего модема, плоттера и т. п. Параллельные порты используются для подключения принтера, сканера, плоттера.

12.6.1 Принтеры

Кроме мониторов к устройствам вывода графических данных относятся и *принтеры*. *Принтер* (printer), или печатающее устройство, предназначен для вывода информации на бумагу. Все современные принтеры могут выводить текстовую информацию, а также рисунки и другие изображения. В настоящее время известно несколько тысяч моделей принтеров, которые могут быть разделены на четыре основных типа — матричные, струйные, лазерные и светодиодные.

Матричные принтеры

До недавнего времени *матричные* (игольчатые) *принтеры* являлись основным стандартным устройством вывода информации для персональных компьютеров, поскольку струйные принтеры работали еще неудовлетворительно, а цена лазерных была достаточно высока. В настоящее время матричные принтеры применяются все реже. Достоинства этих принтеров: удовлетворительная скорость печати и универсальность, заключающаяся в способности работать с любой бумагой, а также низкая стоимость печати. Недостаток: низкое качество печатной продукции, особенно графической. Другой недостаток: игольчатый принтер — механическое устройство, а работа механических узлов всегда сопровождается шумом.

Производительность работы матричных принтеров оценивают но количеству печатаемых знаков в секунду *(cps* — characters per second). Обычными режимами работы матричных принтеров являются:

draft — режим черновой печати;

normal — режим обычной печати;

NLQ (Near Letter Quality) — обеспечивает качество печати, близкое к качеству пишущей машинки.

Матричный принтер формирует знаки несколькими иголками, расположенными в головке принтера. Бумага втягивается с помощью вала, а между бумагой и головкой принтера располагается красящая лента. При ударе иголки но этой ленте на бумаге остается закрашенная точка. Иголки, расположенные внутри головки, обычно активизируются электромагнитным методом. Головка движется по горизонтальной направляющей и управляется шаговым двигателем.

Так как напечатанные знаки внешне представляют собой матрицу, а воспроизводит эту матрицу игольчатый принтер, то часто его называют матричным принтером. Среди матричных принтеров существуют 9-игольчатые и 24-игольчатые. В головке 9-игольчатого принтера находятся 9 иголок, которые, как правило, располагаются вертикально в один ряд. Благодаря горизонтальному движению головки принтера и активизации отдельных иголок напечатанный знак образует как бы матрицу, причем отдельные буквы, цифры и знаки «заложены» внутри принтера в виде бинарных кодов.

В 24-игольчатом принтере используется технология последовательного расположения иголок в два ряда по 12 штук. Вследствие того что иголки в соседних рядах

сдвинуты по вертикали, точки на распечатке перекрываются таким образом, что их невозможно различить. Имеется возможность перемещения головки дважды по одной и той же строке, чтобы знаки пропечатывались еще раз с небольшим смещением. Такое качество печати обозначают как LQ (Letter Quality — машинописное качество), в этом режиме скорость печати уменьшается незначительно, так как головка печатает при движении слева направо и справа налево. Изготовители обычно указывают теоретическую скорость печати, то есть максимально возможную скорость чернового (draft) режима. При этом качество печати не играет роли. LQ-печать у игольчатых принтеров длится дольше.

Режим печати графики является самым медленным. Это связано с тем, что в данном случае набор знаков не читается из внутренней памяти (ROM) принтера, а рассчитывается для каждой печатаемой точки. Игольчатые принтеры оборудованы внутренней памятью (буфером) до 64 Кбайт и более, который принимает данные от персонального компьютера.

Струйные принтеры

В *струйных принтерах* изображение формируется микроскопическими каплями специальных чернил, вылетающих на бумагу через маленькие отверстия. В качестве элементов, выталкивающих струи чернил, используются пьезокристаллы. В основе их работы лежит эффект расширения под действием электричества. По сравнению с матричными принтерами этот способ печати обеспечивает лучшее качество печати и более высокую производительность. К тому же он очень удобен для реализации цветной печати. Цветное изображение формируется с помощью использования (наложения друг на друга) четырех основных цветов. Уровень шума струйных принтеров значительно ниже, чем игольчатых, поскольку его источником является только двигатель, управляющий перемещением печатающей головки.

При черновой печати скорость струйного принтера значительно выше, чем игольчатого. Скорость печати в режиме LQ составляет 3-4 (до 10) страницы в минуту. Качество печати зависит от количества сопел в печатающей головке — чем их больше, тем выше качество. Большое значение имеют качество и толщина бумаги. Для струйных принтеров выпускается специальная бумага, но можно печатать и на обычной бумаге плотностью от 60 до 135 г/м^2. В некоторых моделях для быстрого высыхания чернил применяется подогрев бумаги. Разрешение струйных принтеров при печати графики составляет от 300 x 300 до 720 x 720 dpi. Основные недостатки струйного принтера — большая стоимость расходных материалов и возможность засыхания чернил внутри сопла, что приводит к необходимости замены печатающей головки.

Лазерные и светодиодные принтеры

Лазерные принтеры обеспечивают в настоящее время наилучшее качество печати. В них для печати используются лазерный луч, управляемый компьютером. В лазерном принтере имеется валик, покрытый полупроводниковым веществом, которое электризуется от попадания лазерного света. Луч при помощи поворотного зеркала направляется в то место валика, где должно быть изображение. Это место электризуется и к нему «прилипают» мельчайшие частицы сухой краски, которая находится в контейнере под валиком. После этого валик прокатывается по листу бумаги и краска переходит на бумагу. Для закрепления на бумаге красящего порошка ее пропускают через нагревательный элемент, что приводит к спеканию краски.

К основным параметрам лазерных принтеров относятся:

разрешающая способность, dpi (dots per inch — точек на дюйм);

производительность (страниц в минуту);

формат используемой бумаги;

объем собственной оперативной памяти.

Стоимость цветного лазерного принтера значительно выше, чем черно-белого, а скорость печати — ниже. Лазерные принтеры со средними возможностями печатают 4—8 страниц в минуту. Высокопроизводительные сетевые лазерные принтеры могут печатать до 20 и более страниц в минуту. Печать сложных графических изображений занимает больше времени.

Разрешение по вертикали соответствует шагу барабана и составляет от 1/300 до 1/600 дюйма. Разрешение по горизонтали определяется точностью наведения лазерного луча и количеством точек в строке и составляет, как правило, от 1/300 до 1/1200 дюйма.
Лазерный принтер обрабатывает целые страницы, что связано с большим количеством вычислений. Минимальный объем памяти лазерного принтера не менее 1 Мбайт. Наиболее часто используется память от 2 до 4 Мбайт. Цветные принтеры требуют для работы еще большей памяти.

Память лазерного принтера может быть увеличена путем установки специальных карт с DRAM или SIMM модулями. Большинство лазерных принтеров могут печатать на бумаге формата А4, реже — А3. Некоторые принтеры могут печатать на обеих сторонах листа, но они стоят существенно дороже.

В *светодиодных принтерах* вместо лазера имеется полоса, состоящая из большого количества светодиодов, свет которых электризует полупроводниковый барабан. Все остальное происходит так же, как в лазерном принтере. Светодиоды — это полупроводниковые элементы, которые излучают свет при подаче на них напряжения. Разрешающая способность как светодиодных, так и лазерных принтеров находится в диапазоне от 300 до 1200 точек на дюйм.

12.6.2 Имиджсеттеры

Фотонаборные машины с цифровым формированием изображения, или *имиджсеттеры,* производят вывод на печать с высоким разрешением — от 1000 до 3000 точек на дюйм (dpi). Для черно-белой издательской продукции имиджсеттеры обеспечивают вывод печати и графики с максимальным количеством оттенков серого — 256. Для цветного вывода имиджсеттеры создают четыре пленочных негатива, каждый из которых передает свой цвет (C+M+Y+K). Контрольные оттиски могут быть выведены с негативов, при этом полученные цвета будут полностью соответствовать тем, что выйдут из типографии в конечном варианте. После того как процесс получения пробного оттиска завершен, негативы пересылают на коммерческий принтер, на котором создаются печатные формы с негативов.

12.6.3 Плоттеры

Плоттер (plotter), или *графопостроитель,* — это устройство для вывода различных чертежей, географических карт, плакатов и других изображений на бумагу большого формата. Плоттеры бывают монохромными и цветными. По технологии нанесения изображения плоттеры делятся на перьевые и струйные.
Большинство плоттеров имеют пишущий узел перьевого типа (pen-plotter). Они используют специальные фломастеры или ручки с возможностью их автоматической замены.

Существуют разновидности плоттеров с пишущим узлом струйного типа, а также использующие эффект притягивания частиц краски электростатическим зарядом. Большинство струйных аппаратов обеспечивают печать графических файлов формата TIFF, BMP, PCX. Стандартным языком управления для плоттеров является HP-GL (Hewlett-Packard Grafics Language), а типовым интерфейсом — последовательный RS232 (скорость передачи данных — до 38,4 Кбайт/с). На базе перьевых плоттеров было создано еще одно периферийное устройство — *cutter,* в котором пишущий узел заменен на режущий инструмент. Такое устройство использует специальную полимерную пленку или

бумагу на самоклеющейся основе и применяется для создания рекламно-информационной продукции (ярлыки, наклейки и т. п.).

12.7 Модемы

Модем (модулятор-демодулятор, modem) — устройство для подключения компьютера к глобальной компьютерной сети или для связи с другими компьютерами по телефонной линии. Модем состоит из двух частей — передатчика (модулятора) и приемника (демодулятора). Модулятор передает в низкочастотную телефонную сеть цифровую информацию от компьютера в виде аналоговых сигналов звукового диапазона частот. Демодулятор преобразует эти аналоговые сигналы л цифровые значения, которые может интерпретировать компьютер. Таким образом, модем преобразует аналоговый телефонный сигнал в цифровой компьютерный и наоборот.

Основной характеристикой модема является его скорость работы или скорость передачи данных. Для работы в Интернете рекомендуются модемы со скоростью, на которой работает ваш провайдер (обычно это 56 Кбит в секунду). Модемы бывают внутренними (плата, вставляемая в гнездо расширения) и внешними, подключаемыми к одному из портов компьютера.

12.8 Звуковые карты

Звуковая карта вставляется в свободный слот расширения компьютера и позволяет осуществлять запись, воспроизведение и синтез звука. Встроенный синтезатор помогает воспроизводить сложные звуковые эффекты, не загружая при этом центральный процессор. К таким картам обычно можно подключить микрофон, колонки, наушники, джойстик и привод компакт-диска. При записи звука в PC звуковая карта осуществляет преобразование непрерывного электрического аналогового сигнала, несущего информацию о звуке, в двоичный цифровой код. Преобразование аналогового сигнала в цифровую форму осуществляется с помощью аналого-цифрового преобразователя, основными характеристиками которого являются *частота* и *разрядность квантования:*

Частота квантования показывает, сколько раз в секунду происходит измерение сигнала. Обычно она лежит в пределах от 4 до 48 кГц.

Разрядность определяет точность преобразования. Так, например, 8-разрядные звуковые карты могут измерять сигнал с точностью 1/256 от его максимальной величины. Наибольшее распространение получили 16-разрядные звуковые карты.

На звуковой карте размещены два принципиально различных устройства воспроизведения звука — *синтезатор* и *плеер:*

плеер воспроизводит звук аналогично цифровой аудиотехнике, используя обратное преобразование цифрового кода в аналоговый сигнал с помощью цифро-аналогового преобразователя;

синтезатор служит для синтеза звуков и может использовать либо FM-синтез, либо WT-синтез. Характеристики синтезатора можно определить по тому, какому стандарту соответствует звуковая карта: GM, GS или XG.

12.9 Сканеры

Сканер (scaner) — устройство для копирования графической и текстовой информации и ввода ее в компьютер. Персональные сканеры бывают трех типов — *ручные, планшетные* и *барабанные.* Основными элементами сканера являются полупроводниковый лазер и полупроводниковый фотоприемник. Когда сканер ведут по тексту или изображению, лазерный луч пробегает по листу, сканирует его и отражает на светочувствительный полупроводниковый элемент. Фотоэлемент преобразует световой сигнал в электрический, который затем по шине передается в компьютер. В нем сигнал преобразуется в цифровую форму, содержащую информацию о координатах и цвете

каждого пиксела изображения. И наконец, на последней стадии полученная об изображении информация записывается на диск в виде файла.

Основным элементом сканера является массив фоточувствительных кремниевых ячеек (приборов с зарядовой связью, ПЗС), выполняющих функции датчиков для измерения интенсивности светового потока, отраженного от сканируемого оригинала или прошедшего сквозь него. С помощью аналого-цифрового преобразователя результаты измерений подразделяются на яркостные (цветовые) уровни (например, 256 уровней для каждого из основных цветов при использовании сканера с глубиной цвета 24 бит) и сохраняются в виде последовательности двоичных символов, которые вы впоследствии можете рассматривать и преобразовывать с помощью вашего компьютера.

Цветные сканеры работают по принципу сложения цветов, при котором цветное изображение создается путем смешения трех цветов: красного, зеленого и синего. Технически это реализуется одним из двух способов:

в процессе сканирования цветной оригинал освещается белым цветом, а отраженный свет попадает на ПЗС-матрицу через систему специальных фильтров, разлагающих его на три компонента: красный, зеленый, синий, каждый из которых улавливается своим набором фотоэлементов;

цветной оригинал освещается не белым светом, а последовательно красным, зеленым и синим, для каждого из которых осуществляется своя процедура сканирования. Полученная таким образом информация предварительно обрабатывается и передается в компьютер.

Планшетные сканеры

Планшетные сканеры предназначены для ввода графической информации с прозрачного или непрозрачного листового материала, Принцип действия этих устройств состоит в том, что луч света, отраженный от поверхности материала (или прошедший сквозь прозрачный материал), фиксируется специальными элементами, называемыми приборами с зарядовой связью (ПЗС). Обычно элементы ПЗС конструктивно оформляют в виде линейки, располагаемой по ширине исходного материала. Перемещение линейки относительно листа бумаги выполняется механическим протягиванием линейки при неподвижной установке листа или протягиванием листа при неподвижной установке линейки. Планшетный сканер изображен на рис.12.13. Под крышку такого сканера «лицом» на стекло может закладываться не только отдельный лист или страница, а целая развернутая книга или журнал. Этим планшетные сканеры напоминают копировальные аппараты.

Рис. 12.13. Планшетный сканер «бегемот»

Основными эксплуатационными параметрами планшетных сканеров являются:

- разрешающая способность;
- производительность сканера;
- динамический диапазон;
- максимальный размер сканируемого материала.

Значение разрешающей способности зависит от плотности размещения ячеек ПЗС на линейке, а также от точности механического позиционирования линейки при сканировании. Типичный показатель для офисного применения — 600-1200 dpi (dpi — dots per inch — количество точек на дюйм). Для профессионального применения характерны показатели 1200-3000 dpi.

Производительность сканера характеризуется продолжительностью сканирования листа бумаги стандартного формата и зависит как от совершенства механической части устройства, так и от типа интерфейса, использованного для сопряжения с компьютером.

Динамический диапазон определяется логарифмом отношения яркости наиболее светлых участков изображения к яркости наиболее темных участков. Типовой показатель для сканеров офисного применения составляет 1,8-2,0, а для сканеров профессионального применения — от 2,5 (для непрозрачных материалов) до 3,5 (для прозрачных материалов),

Ручные сканеры

Ручные сканеры — это относительно недорогие устройства небольшого размера (рис. 12.14). Они удобны для оперативного сканирования изображений из книг и журналов. Ширина полосы сканирования обычно не превышает 105 мм, стандартное разрешение 300-400 dpi. К недостаткам ручного сканера можно отнести зависимость качества сканирования от навыков пользователя и невозможность сканирования относительно больших изображений целиком.

Рис. 12.14. Ручной сканер

Барабанные сканеры

В *барабанных сканерах* исходный материал закрепляется на цилиндрической поверхности барабана, вращающегося с высокой скоростью (рис.12.15). Устройства этого типа обеспечивают наивысшее разрешение в диапазоне 2400-5000 dpi. Это связано с использованием в них в качестве чувствительных элементов фотоэлектронных умножителей (ФЭУ вместо ПЗС). В настоящее время барабанные сканеры используются только в типографском производстве.

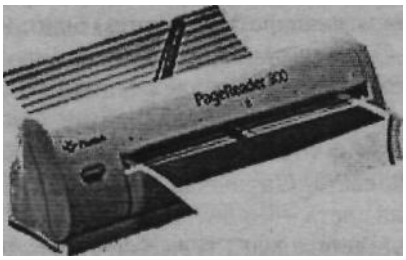


Рис.12.15. Барабанный сканер

Сканирование слайдов (слайд-адаптер)

Существуют специальные сканеры, предназначенные для оцифровки слайдов и негативов. Основными производителями таких устройств являются компании Nicon и Polaroid. Слайдовые сканеры намного дороже планшетных или ручных сканеров. Поэтому их применение оправдано только при необходимости получения большого количества снимков.

Стандарт TWAIN

Этот стандарт разработан для осуществления обмена данными между прикладной программой и внешним устройством (в частности, сканером). Он привлекателен тем, что любая прикладная программа, поддерживающая TWAIN, будет работать с любым TWAIN-совместимым сканером независимо от его конструкции, типа используемого аппаратного интерфейса, параметров сканирования и прочего. Поэтому производитель TWAIN-совместимого сканера может не беспокоиться о программной поддержке своего сканера. Главная его задача — качественно написать TWAIN-совместимый драйвер сканера. Сделать это не так сложно, поскольку TWAIN — открытый и качественно документированный стандарт.

В настоящее время стандарт TWAIN поддерживается большинством современных программных средств, включая:

популярные графические пакеты, такие как Adobe PageMaker, CorelDRAW, Adobe Photoshop, PhotoStyler, PicturePubliher и т. п.;

программы оптического распознавания символов;

факс-приложения (так как некоторые сканеры могут выполнять еще и функции копировального аппарата и факса).

12.10 Секреты графических планшетов (дигитайзеров)

Графический планшет (дигитайзер) предназначен для выполнения профессиональных графических работ. Это практически основной инструмент художников. Рисовать мышкой неудобно, а планшеты позволяют дизайнерам и художникам создавать экранные изображения привычными приемами (рис. 12.16), характерными для традиционных инструментов (карандаш, перо, кисть). С помощью специального программного обеспечения планшет позволяет преобразовывать движение руки оператора в формат векторной графики. В отличие от мыши дигитайзер способен точно определять и обрабатывать абсолютные координаты.

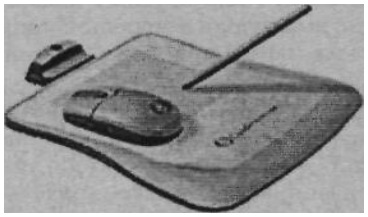


Рис. 12.16. С помощью графического планшета вы сможете освоить компьютерную живопись. Внешний вид графического планшета A4 Tech Tablet 12x12"

Планшет может быть настроен с учетом требований и привычек своего владельца. Для этого используются программируемые кнопки на корпусе пера и макрокнопки, расположенные на самом планшете.

Наличие у планшета функции Erasing означает, что при переворачивании пера включается его противоположный конец, который выполняет функцию стирания. Последнее свойство поддерживается большинством современных популярных редакторов, включая MS Word, WordPerfect, QuarkXPress, PageMaker и т. п. Для того чтобы удалить лишний текст, нужно опустить перо «ластиком» вниз, выделить ненужный фрагмент текста и затем поднять перо. Выделенный текст будет уничтожен.

Достоинства и недостатки графических планшетов

Недостатки

Высокая стоимость, особенно серьезных, солидных моделей.

У доступных широкому кругу покупателей моделей относительно маленькоерабочее пространство (от 7,5 x 10 до 30 x 30 см).

Весьма часто встречается проводное подключение пера или мыши, что довольно неудобно при рисовании.

Достоинства

Это незаменимое и очень удобное в работе средство для профессиональных художников, пользователей, работающих с САПР или занимающихся дизайном и рекламой.

Оцифровка изображения происходит очень точно.

Планшет можно использовать в качестве средства идентификации по электронной подписи, а также в качестве устройства ввода рукописного текста.

Дигитайзеры поддерживаются большинством серьезных графических редакторов и средствами проектирования, так что отпадает необходимость в каком-либо специализированном программном обеспечении.

12.11 Цифровые фотоаппараты и фотокамеры

Цифровая фотокамера — это еще один тип устройства оцифровывания графики и ввода изображений в ПК. В отличие от обычного фотоаппарата в его цифровом аналоге изображение проецируется не на фотопленку, а на полупроводниковую светочувствительную матрицу из ПЗС-ячеек. После этого изображение переводится в цифровую форму и записывается в память фотокамеры. Главным достоинством и основным преимуществом цифровой фотографии является оперативность. Снятый вами кадр буквально через минуту может быть помещен в компьютер и отправлен через Интернет на край света, что очень важно для событийных съемок. Главным же недостатком цифровых фотоаппаратов (с точки зрения фотографа-профессионала) является невозможность (пока!) получения отпечатка приемлемого качества и большого размера на обычной бумаге.

Каков в самом упрощенном виде принцип действия цифровой камеры? Свет, прошедший через объектив, попадает на светочувствительную матрицу (занимающую место пленки), представляющую собой совокупность сенсоров — ПЗС (CCD) или КМОП (CMOS), которые, в свою очередь, и выполняют оцифровку изображения. Светочувствительная матрица (сенсоры) является одним из главных (и самых дорогих) компонентов цифровой камеры. Качество последующей картинки во многом определяется характеристиками сенсоров. Наиболее простые на сегодняшний момент цифровые камеры дают разрешение 640 x 480 пикселов. Более «продвинутые» — 800 x 600. Кроме того, есть еще класс гибридных камер, представляющих собой взятые за основу известные пленочные прототипы с пристегнутой на месте задней крышки матрицей. Как правило, это кодаковские приставки к камерам Nikon и Canon. В результате мы имеем камеру, совместимую со всей данной системой (включая оптику и аксессуары), но, тем не менее, являющуюся цифровой. Такие камеры имеют разрешение порядка 1500 x 1200 пикселов.

После того как мы получили фотокартинку, ее необходимо записать в память. Для этого чаще всего используются форматы JPEG или TIFF. Для фотографа не столько важен формат записи, сколько возможности разных режимов сжатия (естественно, с потерей качества), а также количество памяти в камере. Указанные форматы являются наиболее распространенными в компьютерном мире, а стало быть, обычно совместимы со множеством программ. Что касается памяти, то она может быть встроенной (например, жесткий диск в 20 или 40 Мбайт у камеры Polaroid PDC-2000) либо это могут быть обычные съемные дискеты (1,44 Мбайт) или карты емкостью 2, 4, 8, 10 Мбайт и т. д. Чем больше у вас с собой «памяти», тем большее количество кадров вы можете сиять и сохранить «без перезарядки» (то есть без перекачивания изображений из камеры в компьютер).

Еще одним из достоинств цифровых камер является наличие жидкокристаллического дисплея (экрана), на котором можно посмотреть то, что вы уже сняли, а в некоторых случаях использовать его в качестве видоискателя. Не понравившуюся вам картинку вы можете тут же стереть, освободив часть памяти.

Напоминаем: большое количество энергопотребляющих частей камеры приводит к быстрому истощению батарей или аккумуляторов, но за все надо платить.

Ниже приведена классификация цифровых камер.

Экономичные. Разрешение: в пределах 480 x 640 пикселов. Демонстрируют хорошее качество при размере фотографии примерно до 4 x 5 см.

Любительские. Разрешение: в пределах 600 x 800 пикселов. Хорошее качество при размере фотографии примерно до 5 x 7 см.

Полупрофессиональные. Разрешение: от 1280 x 960 пикселов. Хорошее качество при размере фотографии примерно до 10 x 15 см.

Профессиональные. Разрешение: от 1500 x 1200 пикселов и выше. Хорошее качество при размере фотографии примерно до 15 x 20 см.

Литература

1. М. Н. Петров, В. П. Молочков. Компьютерная графика. Учебник. Серия: Учебник для вузов. Издательство Питер, 2004 г. 812 стр. ISBN 5-94723-758-X
2. Марк Кэмпбелл. Компьютерная графика. The Complete Idiot's Guide ti Computer Illustration. Серия: The Complete Idiot's Guide. Периодическое издание. Издательство АСТ, 2007 г.
3. П. Я. Пантюхин, А. В. Быков, А. В. Репинская. Компьютерная графика. В 2 частях. Серия: Профессиональное образование. Издательство Инфра-М, 2007 г. 88 стр. ISBN 978-5-8199-0284-4, 978-5-16-002734-0
4. В. П. Каминский, Е. И. Иващенко. Инженерная и компьютерная графика для строителей. Серия: Высшее образование. Издательство Феникс, 2008 г. 288 стр.
5. Н. В. Максимов, И. И. Попов, Т. Л. Партыка. Архитектура ЭВМ и вычислительные системы. Серия: Профессиональное образование. Издательство Форум, 2008 г. 512 стр. ISBN 978-5-91134-230-2
6. Шикин Т.В., Борисков А.В. "Компьютерная графика: динамика и реалистические изображения". - М.: "Диалог-МИФИ", 1996 г.
7. Смородинский А.В., Моисеев В.А.. "Библия ACAD 2004". – М.: "Компьютер", 2005 г.
8. Стевол К.,Вильямс Н. Графический пакет 3D Studio и правила работы в нем. - М.: "Диалог-МИФИ", 2003 г.

Постнов К.В. Компьютерная графика. – Москва, 2009. – 247 стр.

В конспекте лекций приведены основные понятия и методы компьютерной графики; подробно рассматриваются модели трехмерной математики, растровые алгоритмы, непосредственная работа с графическими устройствами, вычислительная геометрия, удаление невидимых линий и поверхностей, текстурирование, построение графических интерфейсов. Конспект дает представление об основных направлениях компьютерной графики и позволяет освоить базовые приемы реализации ее алгоритмов на персональных компьютерах. Большое внимание уделено геометрическому моделированию, получению трехмерных моделей, способам работы с ними. Рассмотрены современные подходы к визуализации графической информации и получению реалистичных изображений сложных трехмерных сцен.