МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ ДОНЕЦКОЙ НАРОДНОЙ РЕСПУБЛИКИ

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

КОМПЬЮТЕРНОЕ ОБЕСПЕЧЕНИЕ ИНЖЕНЕРНОЙ ДЕЯТЕЛЬНОСТИ В ЭНЕРГОМЕХАНИЧЕСКОЙ СФЕРЕ

КУРС ЛЕКЦИЙ

(для студентов направления подготовки 15.04.02 и специальности 21.05.04)

Р а с с м о т р е н о
на заседании кафедры
«Энергомеханические системы».
Протокол № 8 от 27.04.2017 г.

ДОНЕЦК – 2017

УДК 681.3.06

Федоров О. В. Компьютерное обеспечение инженерной деятельности в энергомеханической сфере. Курс лекций (д для студентов направления подготовки 15.04.02 и специальности 21.05.04) — Донецк: ДонНТУ, 2017. – 112 с.

Изложены основные понятия, принципы и методы компьютерной графики. Освещены вопросы передачи цвета, кодирования и обработки растровых изображений. Приведены основные методы и алгоритмы двумерной графики. Рассмотрены основы трехмерного моделирования и способы визуализации трехмерных объектов.

Составитель: О. В. Федоров

Рецензент

Компьютерный набор и оформление О. В. Федоров

Ответственный за выпуск Н. Г. Бойко

Оглавление

Введение 4

1. Основные понятия. Визуализация. Графические устройста 5
   1.1 Сущность, виды и задачи компьютерной графики 5
   1.2 Способы визуализации 7
   1.3 Типы графических устройств 9

2. Передача цвета. Цветовые модели 15
   2.1 Физическая и психофизиологическая природа цвета 15
   2.2 Трехкомпонентная теория цвета 18
   2.3 Цветовые модели 21

3. Кодирование растрового изображения. Особенности растра 25
   3.1 Растровые изображения и их основные характеристики 25
   3.2 Способы растровой развертки 27
   3.3 Кодирование цветов и полутонов 30
   3.4 Основные форматы растровых графических файлов 33

4. Методы и алгоритмы двумерной графики 36
   4.1 Координатный метод. Преобразование координат 36
   4.2 Алгоритмы вывода линий 39
   4.3 Кривые Безье и NURBS 42
   4.4 Алгоритмы закрашивания фигур 46
   4.5 Стиль линии. Перо. Алгоритмы вывода толстой линии 49

5. Обработка растровых изображений 52
   5.1 Масштабирование 52
   5.2 Цветовая коррекция 55
   5.3 Фильтрация 60
   5.4 Палитризация и псевдотонирование 63
   5.5 Дефекты растровых изображений и их устранение 72

6. Основы трехмерной графики 74
   6.1 Преобразование координат в пространстве 74
   6.2 Проекции 77
   6.3 Модели описания поверхностей 84

7. Визуализация трехмерных объектов 91
   7.1 Способы и уровни визуализации 91
   7.2 Имитация мелких деталей и микрорельефа 98
   7.3 Освещение и тени 103
   7.4 Метод трассировки лучей 106

Список литературы 111

ВВЕДЕНИЕ

Способность человека создавать художественные образы всегда считалась одной из основных и удивительных черт, отличающих его от животных. Первые изображения, нарисованные на скалах охрой, углем, глиной, появились десятки тысяч лет назад. Со временем техника создания изображений менялась, но неизменной оставалась тяга человека к творчеству, то удивление и восхищение, которое вызывали у современников и потомков произведения мастеров.

Создавая прекрасное, человек экспериментировал с материалами. Что только не шло в ход — холст и краски, бумага и грифель, глина, мрамор, бронза. Стремительный технический прогресс последнего века не прошел для искусства незамеченным. Появились новые способы и технологии создания прекрасного, от гигантских скульптур со стальным каркасом и алюминиевой обшивкой до голографических изображений, полученных с помощью лазера.

Электронные вычислительные машины, впервые появившиеся в 40-х и получившие распространение в 50-х годах, казалось бы, не имели к искусству никакого отношения. Скорее наоборот, их сухая математика и жесткая логика казались чем-то, абсолютно противоположным художественному творчеству. К тому же, первые ЭВМ не имели никаких средств для создания изображений.

Тем не менее, с развитием компьютерной техники, увеличением производительности и объемов памяти, вычислительные машины стали привлекать к решению задач, близких к художественным. С помощью компьютеров стали проектировать корпуса автомобилей и самолетов, здания — появились первые системы автоматизированного проектирования. Их появление поставило перед инженерами и программистами новую задачу — научить компьютер создавать изображения, сделать их наиболее похожими на настоящие, реалистичными. Для этого, во-первых, нужно было создать устройства, способные отображать такие картинки — цветные мониторы и принтеры с высоким качеством печати, а во-вторых, разработать методы и алгоритмы, позволяющие создавать эти изображения.

Успех в этой области был достигнут не сразу. Потребовалось пять десятилетий поисков, исследований и разработок, чтобы мы научились создавать художественные фильмы, в которых не снимался ни один живой актер, путешествовать по выдуманным мирам компьютерных игр, проектировать самолеты и подводные лодки, ни разу не прикоснувшись к карандашу. Сейчас то, что мы называем ***компьютерной графикой***, окружает нас повсюду. В телепередачах и художественных фильмах, рекламных плакатах, книгах журналах и газетах — повсюду мы видим изображения и видео, созданные с помощью компьютера. Но мало кто представляет себе, насколько сложные алгоритмы скрыты за этими «картинками», какие объемы вычислений необходимо выполнить, чтобы их создать. Далее мы рассмотрим те идеи, методы и алгоритмы, которые положены в основу компьютерной графики.

1. ОСНОВНЫЕ ПОНЯТИЯ. ВИЗУАЛИЗАЦИЯ. ГРАФИЧЕСКИЕ УСТРОЙСТА

1.1 Сущность, виды и задачи компьютерной графики

Компъютерная графика включает визуализацию, обработку и распознавание изображений с помощью ЭВМ.

Визуализация — 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 — графики, диаграммы, схемы, карты. Или наоборот, имитация трехмерной реальности — изображение сцен в компьютерных играх, художественных фильмах, тренажерах, в системах архитектурного проектирования.

Обработка это преобразование готовых изображений. Примерами обработки изображений могут служить: повышение контраста, четкости, коррекция цветов, сглаживание, устранение дефектов, ретуширование, реставрация фотоснимков, создание коллажей и так далее. В качестве материала для обработки могут использоваться сканированные изображения, цифровые фотографии, кинокадры, искусственно синтезированные изображения.

Для ***распознавания*** изображений основная задача — получение описания объектов, представленных изображением (обычно растровым). 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*.

Назначение компьютерной графики весьма разнообразно. Это:

— ***художественная графика*** (оформление изданий, создание рекламных проспектов, интернет-сайтов и т.д.);

— ***инженерная графика*** (создание чертежей и 3D-моделей);

— ***кинематографическая графика*** (компьютерные спецэффекты в фильмах, компьютерные мультфильмы, видеоролики и заставки);

— ***интерактивная анимированная графика*** (2D – и 3D – компьютерные игры, тренажеры и симуляторы);

— ***графика систем управления*** (отображение информации на дисплеях диспетчеров, операторов, пультах управления машин, станков и т.д., графические операционные системы и интерактивные программы);

Довольно популярным до недавнего времени было словосочетание ***интерактивная компьютерная графика***, обозначавшее способность компьютерной системы создавать графику в режиме нормального времени и вести диалог с человеком. В настоящее время почти любую программу можно считать интерактивной системой компьютерной графики.

Исторически первыми интерактивными системами считаются ***системы автоматизированного проектирования*** (САПР — английская аббревиатура CAD — Computer Aided Design), которые появились в 60-х годах. Они представляют собой значительный этап в эволюции компьютеров и программного обеспечения. В системе интерактивной компьютерной графики пользователь воспринимает на дисплее изображение, представляющее некоторый сложный объект, и может вносить изменения в описание (модель) объекта. 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.

Системы автоматизированного проектирования активно используются во многих областях, например, в машиностроении и электронике. Одними из первых были созданы САПР для проектирования самолетов, автомобилей, объектов архитектуры, разработки микроэлектронных интегральных схем и т.п. Такие системы сначала функционировали на больших компьютерах и были доступны лишь крупным фирмам и проектным организациям. Затем получили распространение быстродействующие компьютеры среднего класса с развитыми графическими возможностями — графические рабочие станции. С возрастанием мощностей персональных компьютеров все чаще 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. Назовем несколько популярных САПР: многоцелевая система для выполнения проектных работ в разных областях — *AutoCAD* и его российский аналог — система *KOMPAS-3D*, для архитекторов — *ArchiCAD*, для проектирования электронных схем — *ORCAD*, *PCAD* и т.д.

Широко используются графические системы в дизайне, рекламе. Кроме традиционных ***2D-редакторов*** — *Adobe Photoshop*, *CorelDraw* и подобных им — используются и ***3D-пакеты***. Анимационные ролики создаются средствами систем моделирование трехмерных сцен, таких как *Maya*, *3D Studio Max*, *Light Wave*, *Bryce 3D* и других. Одно из направлений исследований и разработок для современной компьютерной графики — анимация движения людей и животных, изучение мимики.

Важным этапом развития систем компьютерной графики стали так называемые ***системы виртуальной реальности*** (virtual reality). Наращивание мощности компьютеров, повышение реалистичности трехмерной графики и усовершенствование способов диалога человека с компьютером позволяют

создавать иллюзию вхождения человека в виртуальное пространство. Это пространство может быть моделью существующего пространства или выдуманного. Первоначально такие системы использовались как тренажеры для пилотов, однако наибольшее распространение получили разнообразные компьютерные игры-3D. Следует отметить, что во многих компьютерных играх реализованы идеи и методы, которые ранее были воплощены в профессиональных дорогих системах.

Широко используется компьютерная графика в кино. До недавнего времени технологии компьютерной графики использовались для спецэффектов, создания изображений экзотических чудовищ, имитации стихийных бедствий и других элементов, служащих лишь фоном для игры живых актеров. В 2001 году вышел на экраны полнометражный кинофильм "Финальная фантазия", в котором все, включая изображения людей, синтезировано компьютером — живые актеры только озвучили роли за кадром.

Как наука, компьютерная графика включает в себя ***разделы***:

1. Способы кодирования и отображения графической информации (растровая и векторная графика, способы цветопередачи).
2. Алгоритмы растровой двумерной графики (отображение отрезков, кривых, литер, заполнение цветом, обработка растровых изображений).
3. Алгоритмы векторной двумерной графики (создание, отображение и редактирование векторных изображений).
4. Алгоритмы трехмерной графики (создание и отображение 3D-моделей, визуализация, отсечение невидимых частей, создание реалистических изображений).
5. Способы анимации двух- и трехмерных изображений.

1.2 Способы визуализации

Наиболее известны два способа визуализации: ***растровый*** и ***векторный***.

Растровая визуализация основывается на представлении изображения на экране или бумаге в виде совокупности отдельных точек (пикселов). Вместе пикселы образуют растр.

Векторная визуализация — это создания изображения на экране или бумаге посредством рисования сплошных линий (векторов) — прямых или кривых.

Качество векторной визуализации обуславливается точностью вывода и номенклатурой ***базовых графических примитивов*** — линий, дуг, эллипсов и других. При этом в памяти компьютера хранится информация об изображении в виде набора данных об этих примитивах — их форме, цвете, размерах, взаимном расположении и порядке отображения. Для вывода векторного изображения на экран или печатающее устройство система каждый раз заново «рисует» изображение по этому набору данных.

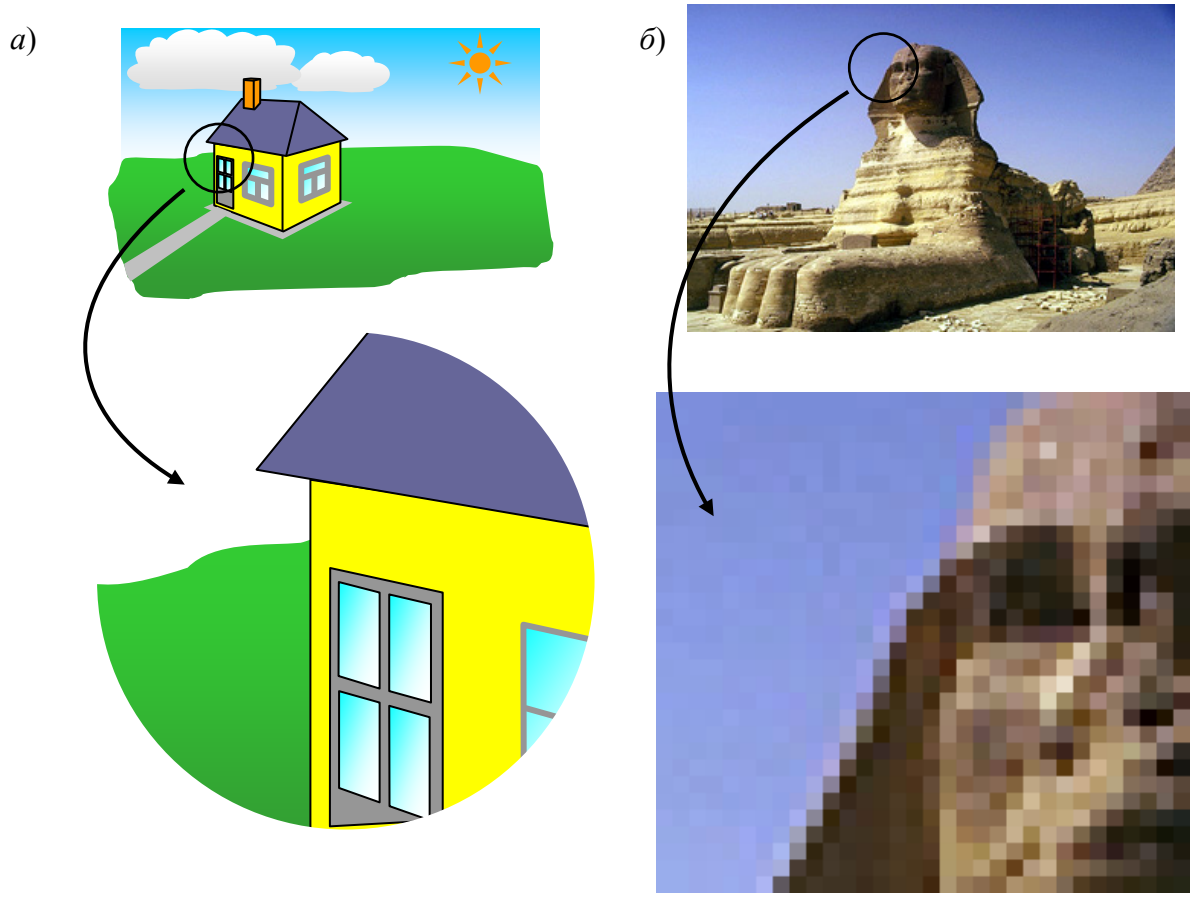


Рисунок 1.1 – Примеры векторного (*а*) и растрового (*б*) способов визуализации

К *преимуществам* векторного способа визуализации относятся:

— возможность *редактировать* изображение путем добавления новых примитивов, их удаления, а также изменения параметров уже существующих примитивов; при этом остальная графическая информация остается неизменной;

— возможность *масштабировать* изображение (увеличивать или уменьшать) без потери качества, рис. 1.1 *а*;

— относительно небольшой объем памяти, занимаемый информацией об изображении.

Недостатками векторного способа являются:

— относительно малая скорость вывода изображений на растровые устройства отображения, что связано с необходимостью выполнения больших объемов вычислений при ***растеризации*** векторного изображения (преобразовании его в растр);

— ограниченные возможности в отображении — векторный способ визуализации хорош для схем, чертежей, рисунков и других «искусственных»

объектов, но практически непригоден для хранения реалистических изображений, например, фотографий;

— поскольку информация о составе изображения закодирована, изображение может быть расшифровано только устройством или программой, «понимающими» этот код; это приводит к проблемам совместимости при переносе векторных изображений в другие программные пакеты.

В настоящее время доминирует растровый способ визуализации. Это обусловлено большей распространенностью растровых дисплеев и принтеров. Растровое изображение значительно быстрее, чем векторное, выводится на дисплей, поскольку для этого не нужно выполнять сложные вычисления. Растровый способ визуализации удобен для хранения сложных изображений, в том числе и многоцветных фотореалистических, рис. 1.1 *б*, но обладает существенными недостатками:

— дискретность изображения (разложение на отдельные точки) ухудшает его качество и ограничивает возможность масштабирования, см. рис. 1.1 *б*;

— растровые изображения значительно сложнее редактировать, поскольку в памяти компьютера хранится информация об отдельных точках (пикселах), а не об объектах, которые мы хотим изменить;

— растровые изображения (особенно полноцветные) занимают большие объемы памяти.

В современных компьютерных системах получила распространение смешанная технология, при которой на растровых устройствах отображения (дисплеях) реализуется «виртуальная» векторная графика. Пользователь создает и редактирует векторное изображение, которое выводится на экран растрового дисплея с помощью алгоритмов растеризации. Примерами могут служить графический редактор *CorelDraw*, встроенный графический редактор *Microsoft Word*, а также CAD-пакеты: *AutoCAD*, *KOMPAS-3D* и др.

1.3 Типы графических устройств

Существует много разнообразных устройств для **вывода изображений**, построенных с помощью машинной графики:

— перьевые графопостроители, точечно-матричные, струйные и лазерные печатающие устройства;

— векторные дисплеи на запоминающей трубке;

— векторные дисплеи с регенерацией изображения;

— растровые дисплеи на электронно-лучевой трубке;

— жидкокристаллические дисплеи.

Печатающие устройства используются для вывода на бумагу результатов обработки изображений. Для непосредственной работы с изображением используются различные устройства на электронно-лучевых трубках и ЖК-дисплеи.

Наиболее часто в качестве устройства отображения выступает ***видеомонитор***, основу которого составляет ***электронно-лучевая трубка*** (***ЭЛТ***).

На рис. 1.2 схематично показана ***ЭЛТ***, используемая в видеомониторах. ***Катод*** (отрицательно заряженный) нагревают до тех пор, пока возбужденные электроны не создадут расширяющегося облака (электроны отталкиваются друг от друга, так как имеют одинаковый заряд). Эти электроны притягиваются к сильно заряженному положительному ***аноду***, для чего между катодом и анодом создается разность потенциалов в несколько десятков киловольт.

На внутреннюю сторону расширенного конца ***ЭЛТ*** нанесен ***люминофор*** — вещество, способное испускать видимый свет при облучении пучком электронов.

Облако электронов с помощью ***электронных линз*** фокусируется в узкий, строго параллельный пучок, который дает яркое пятно в центре ЭЛТ. Луч отклоняется влево или вправо, выше или ниже центра с помощью ***усилителей горизонтального и вертикального отклонения***. Перемещаясь по экрану, луч оставляет след — линию.

Для отклонения луча могут использоваться как ***электроды***, на которые подается напряжение (ЭЛТ осциллографов), так и ***электромагнитные катушки*** (ЭЛТ телевизоров). Во втором случае длина ЭЛТ значительно уменьшается, но усложняется система управления лучом.

Для регулирования яркости линии используется управляющий электрод, на который подается отрицательный заряд. Чем выше напряжение на управляющем электроде, тем меньше интенсивность пучка электронов и яркость светового пятна.

В цветной ЭЛТ находятся ***три электронные пушки***, по одной на каждый основной цвет: ***красный***, ***зеленый*** и ***синий***. Электронные пушки часто объединены в блок, соответствующий подобному блоку точек красного, зеленого и синего люминофоров на экране ЭЛТ, рис. 1.3

Для того, чтобы электронные пушки возбуждали только соответствующие им точки люминофора (например, красная пушка возбуждала только точку красного люминофора), между электронными пушками и поверхностью экрана помещена перфорированная металлическая решетка — ***теневая маска***.

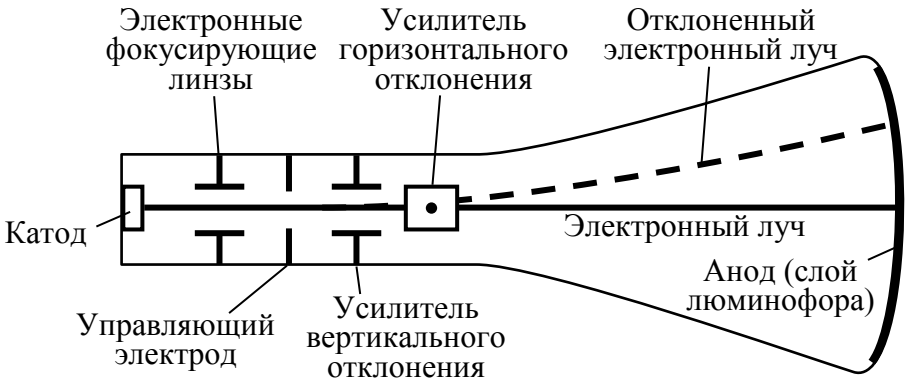


Рисунок 1.2 – Схема устройства электронно-лучевой трубки

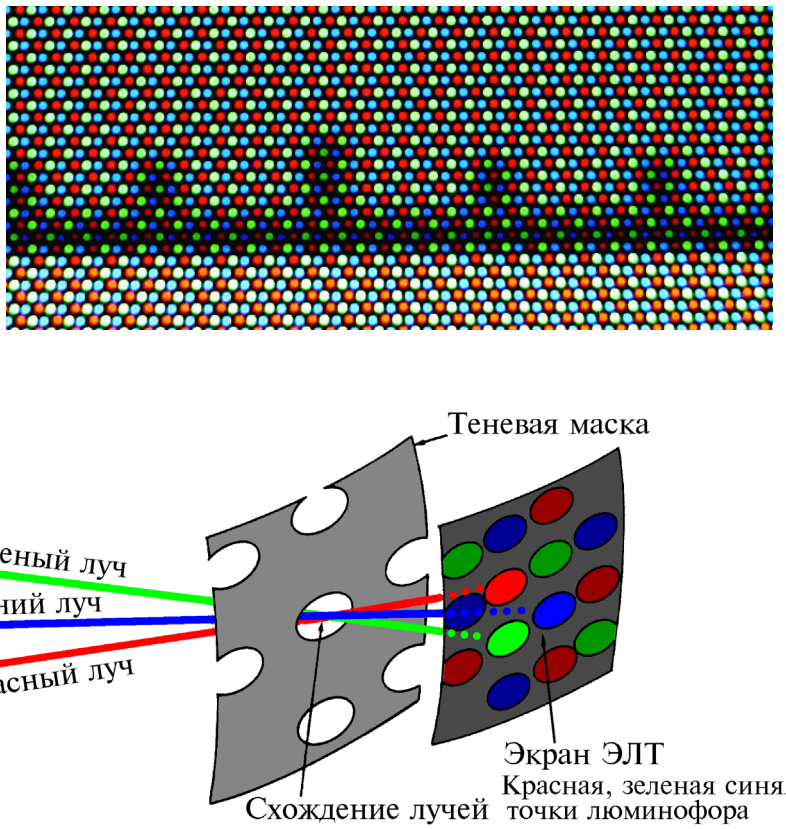


Рисунок 1.4 – Расположение электронной пушки и теневой маски цветной ЭЛТ

Цветовые пушки расположены таким образом, что их лучи сходятся и пересекаются в плоскости теневой маски, рис. 1.4. После прохождения через отверстие красный луч, например, защищен от попадания на зеленую или синюю точки люминофора. Он может попасть лишь на красную точку. Изменяя интенсивность электронного луча для каждого основного цвета, можно получить различные оттенки. Комбинация этих оттенков дает большое количество цветов для каждой точки.

Из всех дисплеев на ЭЛТ наиболее просто устроен дисплей на запоминающей ЭЛТ с прямым копированием изображения. ***Запоминающая ЭЛТ*** это ЭЛТ, покрытая люминофором с длительным временем послесвечения.

Линия или литера остаются на ней видимыми в течение длительного времени, прежде чем станут окончательно неразличимыми, рис. 1.5 *а*. Чтобы ***нарисовать*** отрезок на дисплее, интенсивность электронного луча увеличивают до такой величины, которая вызывает запоминание следа луча на люминофоре.

Для ***стирания*** изображения на всю трубку подают специальное напряжение, снимающее свечение люминофора. При этом стираются все отрезки и литеры. Таким образом, стереть отдельные линии и литеры нельзя, и изображение динамического движения или анимация невозможны.

Дисплей на запоминающей трубке — это ***векторный дисплей***, или дисплей с произвольным сканированием. Это означает, что отрезок (вектор)

а) *б*)

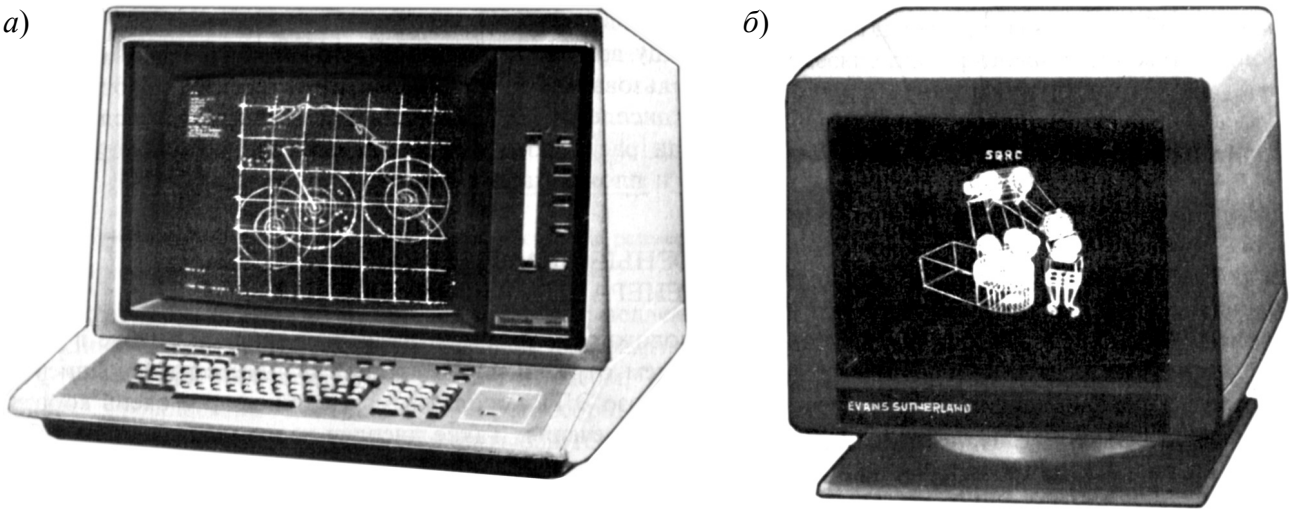


Рисунок 1.5 – Векторные дисплеи на запоминающей трубке (*а*) и с регенерацией (*б*)

может быть нарисован непосредственно из одной адресуемой точки в любую другую. ЭВМ обрабатывает изображение в виде набора ***координат узловых точек***, по которым движется электронный луч.

В противоположность дисплею на запоминающей трубке в ***векторном дисплее с регенерацией изображения***, рис. 1.5 *б*, используется люминофор с очень коротким временем послесвечения. Изображение на ЭЛТ за секунду должно многократно перерисовываться — ***регенерироваться***. Частота регенерации должна составлять не менее 30 Гц, в противном случае изображение будет мерцать.

Для векторного дисплея с регенерацией требуется кроме ЭЛТ еще два элемента: дисплейный буфер и дисплейный контроллер. ***Дисплейный буфер*** — это непрерывный участок памяти, содержащий информацию, необходимую для вывода изображения на ЭЛТ (набор координат узловых точек).

Функция ***дисплейного контроллера*** заключается в том, чтобы циклически обрабатывать эту информацию со скоростью регенерации.

Сложность рисунка (число изображаемых векторов) ограничивается двумя факторами — размером дисплейного буфера и скоростью дисплейного контроллера.

Широко используемые в 60-е и 70-е годы, векторные дисплеи не получили дальнейшего развития в силу ограниченных возможностей отображения — невозможности передачи полутоновых и цветных изображений.

Значительно больше возможностей для создания и обработки изображений дали ***растровые графические дисплеи*** на электронных лучевых трубках, к которым относится большинство используемых в настоящее время компьютерных дисплеев.

Растровое устройство можно рассматривать как матрицу дискретных точек — ***пикселей***, каждая из которых может быть подсвечена. Информация о состоянии каждой точки содержится в специальной памяти — ***буфере кадра***.

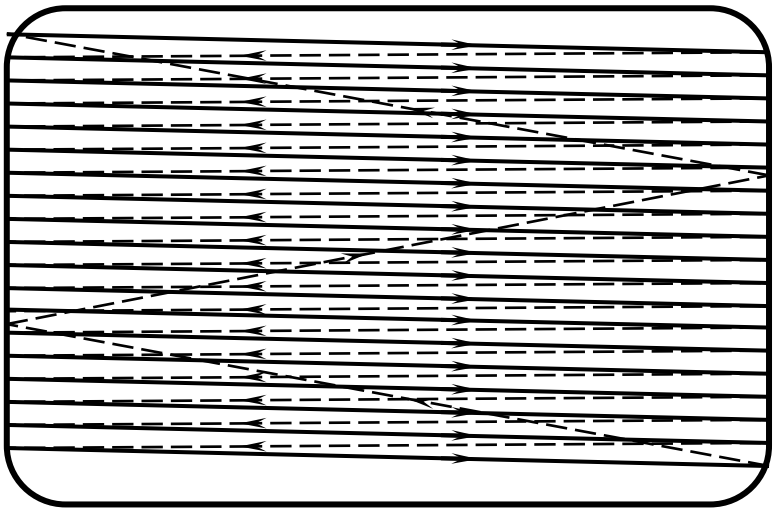


Рисунок 1.6 – Схема растровой развертки

Процесс преобразования хранящейся в буфере кадра растровой картинки в упорядоченный набор точек на телеэкране называется ***растровой разверткой***, рис. 1.6. Электронный луч движется по экрану горизонтально слева направо, затем гасится и быстро возвращается влево, переходя при этом на строку, расположенную ниже. После прохода всего экрана луч быстро возвращается вверх.

При движении луча ***дисплейный контроллер*** управляет его интенсивностью в соответствии с содержимым буфера кадра. При этом на экране создается изображение, состоящее из светлых или темных точек в черно-белых дисплеях, или из точек различных цветов в цветных.

Для устранения мерцания изображения частота его регенерации должна быть весьма высокой — в современных дисплеях — до 100 Гц и выше.

Качество растрового изображения определяется его ***разрешением*** — количеством горизонтальных строк и пикселей в каждой строке. Возможность увеличения разрешения ограничивается объемом памяти буфера кадра и максимальной частотой развертки дисплея.

Обладая рядом преимуществ, дисплеи на ЭЛТ имеют недостатки — большие габариты (что особенно заметно при их расположении на обычном рабочем столе) и высокое энергопотребление. Поэтому в последнее время все большее распространение получают ***жидкокристаллические дисплеи***.

Жидкие кристаллы — это органические вещества, способные под напряжением изменять величину пропускаемого света. Жидкокристаллический монитор представляет собой две стеклянных или пластиковых пластины, между которыми находится суспензия. Кристаллы в этой суспензии расположены параллельно по отношению друг к другу, тем самым они позволяют свету проникать через панель. Под действием электрического напряжения расположение кристаллов изменяется, и они начинают препятствовать прохождению света.

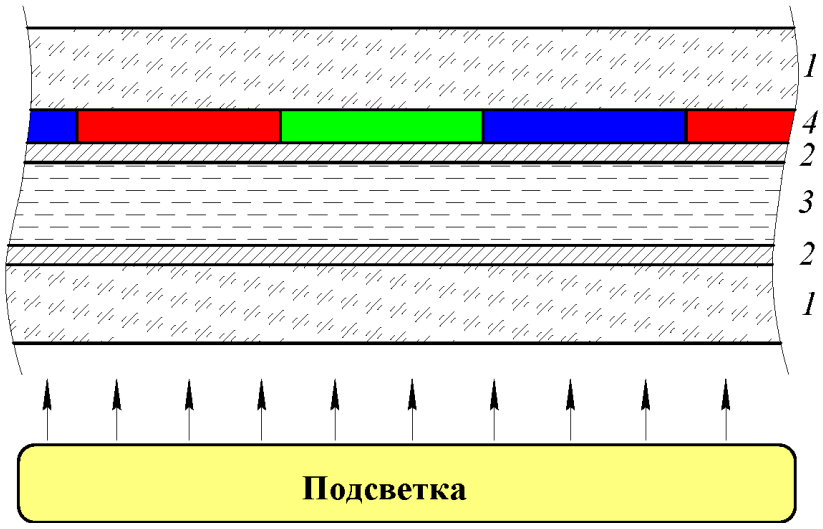


Рисунок 1.6 – Схема устройства ЖК-монитора: *1* – стекло;
2 – управляющие электроды; *3* – слой жидких кристаллов; *4* – цветной фильтр

Жидкокристаллический дисплей состоит из нескольких слоев, рис. 1.6 Крайний слой любой из сторон выполнен из стекла. Между этими слоями расположены тонкопленочные управляющие электроды, панель цветного фильтра, обеспечивающая нужный цвет — красный, синий или зеленый, и слой жидких кристаллов. Изнутри экран освещается флуоресцентной подсветкой.

Верхний и нижний электроды состоят из набора горизонтальных и вертикальных проводников. При нормальных условиях, когда нет напряжения, жидкие кристаллы находятся в аморфном состоянии и пропускают свет. Для затенения некоторого пикселя подается напряжение на соответствующие горизонтальный и вертикальный проводники. В месте их пересечения создается разность потенциалов, которая поляризует жидкие кристаллы и слой в этом месте теряет прозрачность, рис. 1.7.

Как и в традиционных электроннолучевых трубках, пиксель в цветном ЖК-мониторе формируется из трех участков — красного, зеленого и синего. Различные цвета получаются в результате изменения величины степени затенения каждого из цветов.

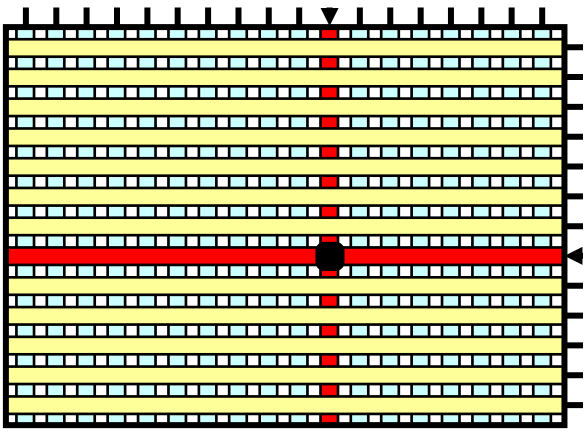


Рисунок 1.7 – Схема засветки точки на ЖК-мониторе

2. ПЕРЕДАЧА ЦВЕТА. ЦВЕТОВЫЕ МОДЕЛИ

2.1 Физическая и психофизиологическая природа цвета

Восприятие ***цвета*** человеком имеет как *физическую*, так и *психофизиологическую*, природу. Оно зависит от физических свойств ***света***, т. е. электромагнитной энергии, от его взаимодействия с физическими веществами, а также от их интерпретации зрительной системой человека.

Зрительная система человека воспринимает как ***видимый свет*** электромагнитную энергию с длинами волн от 380 до 760 нм (1 нм = 10^{-9} м). Излучения с длинами волн от 380 до 470 нм имеют фиолетовый и синий цвет, от 470 до 500 нм — сине-зеленый, от 500 до 560 нм — зеленый, от 560 до 590 нм — желто-оранжевый, от 590 до 760 нм — красный, рис. 2.1.

Система цветового зрения человека включает два типа *светочувствительных фоторецепторов*, расположенных на *сетчатке* глаза: колбочки, чувствительные к некоторому цвету, и палочки, не обладающие преимущественной чувствительностью к какому-либо цвету и играющие главную роль в создании ахроматических зрительных образов. В каждом глазу 6 … 8 млн. колбочек и 100 … 120 млн. палочек.

Существуют три типа колбочек, пики чувствительности которых приходятся примерно на 420 нм, 534 нм и 564 нм, которые называют соответственно «синими», «зелеными» и «красными». Термины «красный» и «зеленый» применительно к колбочкам весьма условны, поскольку пиковые значения 534

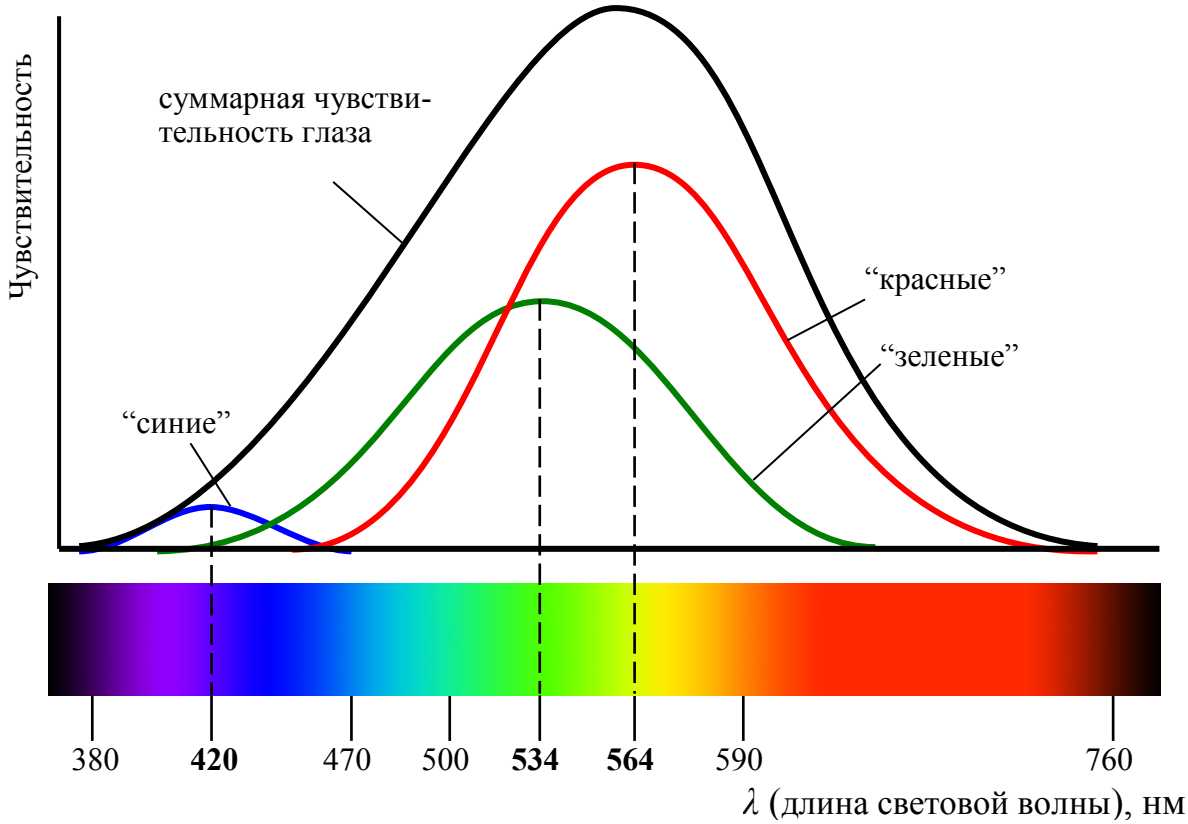


Рисунок 2.1 – Восприятие световых волн фоторецепторами глаза

и 564 нм лежат в желтом диапазоне. Чувствительность глаза к синему цвету существенно ниже, чем к зеленому и красному.

Суммарная чувствительность глаза максимальна при длине волны порядка 550 нм, а на краях видимого диапазона спектра она резко падает. Кривая суммарной чувствительности на рис. 2.1 называется *функцией спектральной чувствительности глаза*. Это мера световой энергии или интенсивности с учетом свойств глаза — источники излучения одинаковой интенсивности но с различным спектральным составом будут восприниматься глазом как имеющие разную яркость.

Свет воспринимается либо *непосредственно от источника*, например электрической лампочки, экрана ЭЛТ, либо *косвенно* при *отражении* от поверхности объекта или *преломлении* в нем.

Различают понятия *яркости* и *светлоты*. ***Яркость*** является свойством самосветящихся или *излучающих* объектов и определяет интенсивность излучения. ***Светлота*** — свойство несветящихся или *отражающих* объектов отражать некоторую часть падающего на них света.

Интенсивность отраженного света удобно рассматривать в диапазоне от 0 до 1, где 0 соответствует черному, 1 — белому, а промежуточные значения — серому цвету.

Источник или объект является ***ахроматическим***, если наблюдаемый свет содержит все видимые длины волн в приблизительно равных количествах. Ахроматический источник кажется *белым*, а отраженный или преломленный ахроматический свет — *белым*, *черным* или *серым*. *Белыми* выглядят объекты, ахроматически отражающие более 80% света белого источника, а *черными* — менее 3%. Промежуточные значения дают различные *оттенки серого*.

Если воспринимаемый свет содержит длины волн в *произвольных неравных количествах*, то он называется ***хроматическим***. Если длины волн сконцентрированы у верхнего края видимого спектра, то свет кажется *красным* или красноватым, т. е. доминирующая длина волны лежит в красной области видимого спектра. Если длины волн сконцентрированы в нижней части видимого спектра, то свет кажется *синим* или голубоватым, т. е. доминирующая длина волны лежит в синей части спектра.

Однако сама по себе электромагнитная энергия определенной длины волны не имеет никакого цвета. *Ощущение цвета* возникает в результате преобразования физических явлений в глазу и мозге человека. *Цвет* объекта зависит от распределения длин волн источника света и от физических свойств объекта. Объект кажется *цветным*, если он отражает или пропускает свет лишь в узком диапазоне длин волн и поглощает все остальные. При взаимодействии цветов падающего и отраженного или пропущенного света могут получиться самые неожиданные результаты. Например, при отражении зеленого света от белого объекта, объект кажется зеленым, а если зеленым светом освещается

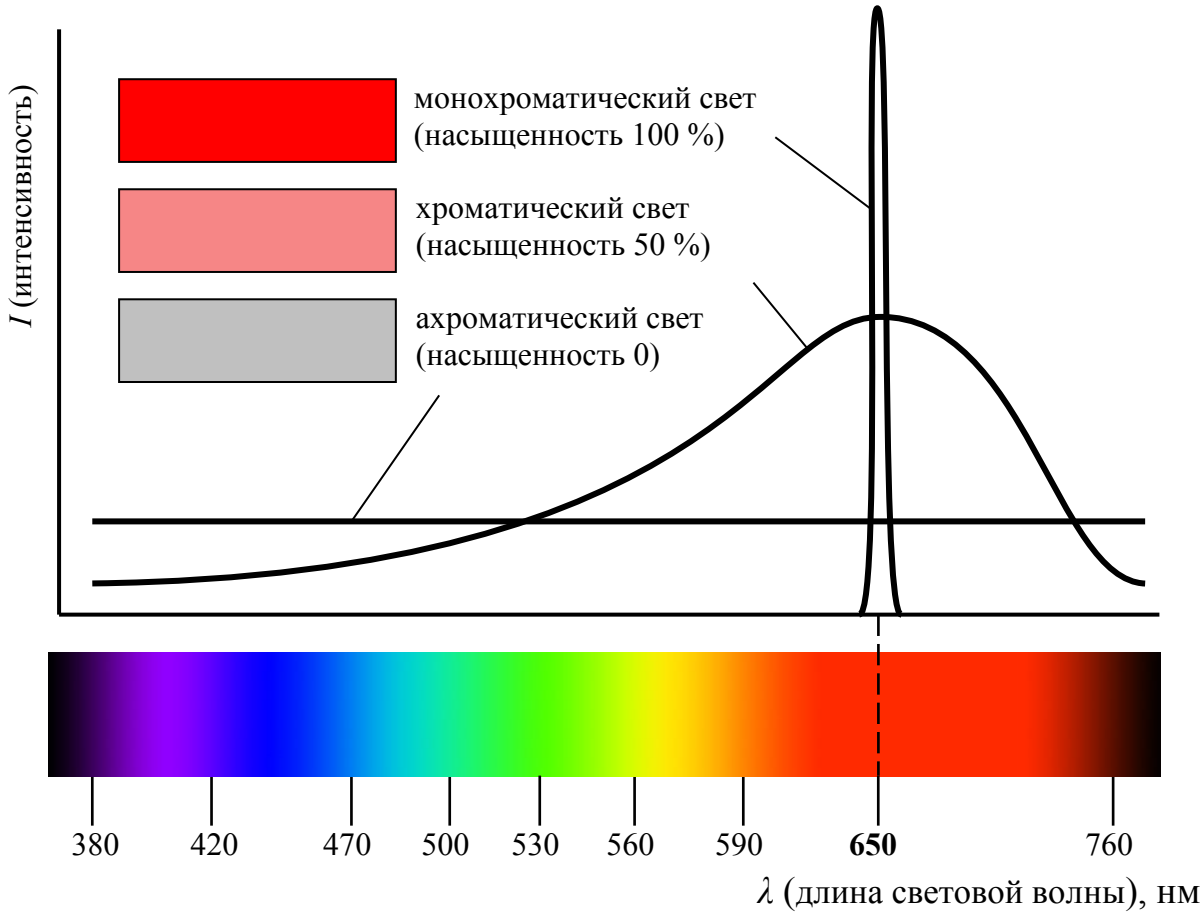


Рисунок 2.2 – Хроматический и ахроматический свет

красный объект, то он будет черным, так как от него свет вообще не отражается.

Психофизиологическое представление света определяется *цветовым тоном*, *насыщенностью* и *светлотой*. ***Цветовой тон*** позволяет различать цвета, а ***насыщенность*** — определять степень ослабления (разбавления) данного цвета белым цветом. У чистого цвета она равна 100% и уменьшается по мере добавления белого. Насыщенность ахроматического цвета составляет 0%, а его светлота равна интенсивности этого света.

Физическими эквивалентами цветового тона, насыщенности и светлоты являются *доминирующая длина волны*, *чистота* и *яркость*. Электромагнитная энергия одной длины волны в видимом спектре дает ***монохроматический*** цвет. На рис. 2.2 изображено распределение энергии ахроматического («белого») света, хроматического света с доминирующей длиной волны 650 нм и насыщенностью 50 %, и монохроматического света с такой же длиной волны.

Яркость пропорциональна энергии света и рассматривается как суммарная энергия волн всех длин. Графически яркость света определяется площадью под кривой спектрального состава, рис. 2.2. Однако субъективное ощущение яркости при восприятии света человеческим глазом зависит от его спектральной чувствительности. Свет зеленого или желтого тона будет казаться значительно ярче, чем свет с такой же энергией, но синего тона.

2.2 Трехкомпонентная теория цвета

Обычно встречаются не чистые монохроматические цвета, а их смеси. В основе ***трехкомпонентной теории цвета*** служит предположение о том, что в центральной части сетчатки находятся три типа чувствительных к цвету колбочек. Первый воспринимает длины волн, лежащие в середине видимого спектра, т. е. зеленый цвет; второй — длины волн у верхнего края видимого спектра, т. е. красный цвет; третий — короткие волны нижней части спектра, т. е. синий.

Если на все три типа колбочек воздействуют волны одинаковой интенсивности (яркости), то свет кажется *белым*. Естественный белый свет содержит все длины волн видимого спектра, однако *ощущение белого света* можно получить, смешивая любые три цвета, если ни один из них не является *линейной комбинацией* двух других. Такие три цвета называются ***основными***.

В машинной графике применяются две системы смешения основных цветов, рис. 2.3:

— ***аддитивная*** — красный, зеленый, синий (*RGB*);

— ***субтрактивная*** — голубой, пурпурный, желтый (*CMY*).

Аддитивная цветовая система RGB удобна для *светящихся поверхностей*, например экранов ЭЛТ. Поверхность экрана ЭЛТ, ЖК-дисплея, цветной панели или другого устройства отображения цветной графической информации состоит из участков, излучающих свет трех основных цветов — обычно красного, зеленого и синего, см. п. 1.3. Поскольку эти участки расположены рядом и весьма малы, при наблюдении такого экрана с некоторого расстояния они сливаются, т.е. в один светочувствительный рецептор глаза наблюдателя попадает свет из нескольких участков различных цветов. Глаз «*суммирует*» эти излучения и воспринимает их как единый световой поток с некоторым спектральным составом — суммой спектральных составов излучений расположенных рядом источников света, рис. 2.3 *а*.

а)

б)

Рисунок 2.3 – Цветовые системы: аддитивная *RGB* (*а*) и субтрактивная *CMY* (*б*)

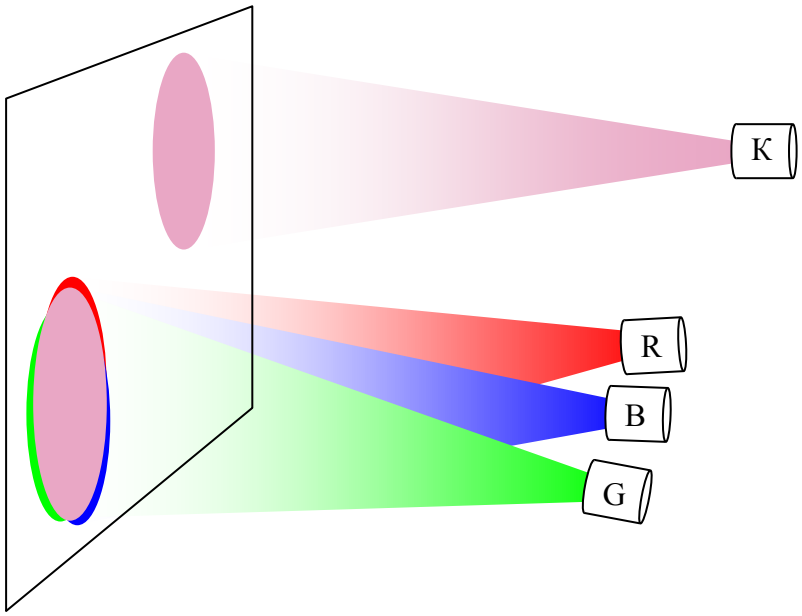


Рисунок 2.4 – Схема уравновешивания цветов: R, G, B — источники красного, зеленого и синего монохроматического света; К — контрольный источник

Субтрактивная система CMY применяется для *отражающих поверхностей*, например типографских красок, цветных фотоизображений, а также прозрачных пленок и несветящихся экранов. В субтрактивных системах из спектра белого света, падающего на изображение или проходящего сквозь пленку, поглощаются (*вычитаются*) волны некоторого цвета, рис. 2.3 *б*. Например, чтобы получить красный цвет, из белого нужно вычесть голубой (сумму синего и зеленого). При отражении или пропускании света сквозь пурпурный объект поглощается зеленая часть спектра. Такие цвета называются *дополнительными*.

Дополнительный цвет — это разность белого и данного цвета: голубой это белый минус красный, пурпурный — белый минус зеленый, желтый — белый минус синий. Цвета одной системы являются дополнительными к другой: голубой — к красному, пурпурный — к зеленому, желтый — к синему. Интересно, что в спектре радуги пурпурного цвета нет, т. е. он порождается зрительной системой человека.

Передача произвольного цвета с помощью трех основных цветов производится следующим образом. Пусть на некоторый фон падает произвольный контрольный свет. Наблюдатель пытается опытным путем уравнять на фоне рядом с контрольным светом его цветовой фон, насыщенность и светлоту при помощи монохроматических потоков света разной интенсивности, рис. 2.4.

Если используется только один инструментальный (уравнивающий) цвет, то длина волны у него должна быть такой же, как у контрольного. В противном случае, если не принимать в расчет *цветовой тон* и *насыщенность* контрольного света, можно уравнять цвета по *светлоте*. Эта процедура называется ***фотометрией***. Таким способом создаются монохроматические репродукции цветных изображений.

Если в распоряжении наблюдателя есть два монохроматических источника, то он может уравнять ограниченное количество контрольных образцов. Добавляя третий инструментальный цвет, можно получить почти все контрольные варианты, при условии, что эти три цвета достаточно широко распределены по спектру и ни один из них не является линейной комбинацией других, т. е. что это *основные цвета*. Желательно, чтобы первый цвет лежал в области спектра с большими длинами волн (*красный*), второй — со средними (*зеленый*) и третий — с малыми длинами волн (*синий*). Объединение этих трех цветов для уравнивания контрольного цвета математически выражается как

$$C = rR + gG + bB$$

где C — цвет контрольного света; R, G, B — красный, зеленый и синий инструментальные потоки света; r, g, b — относительные количества соответствующих потоков света. Изучением вопросов цветопередачи занимается ***колориметрия***.

Результаты колориметрических исследований обобщаются в трех ***законах Грассмана***:

1. Глаз реагирует на три различных стимула, что подтверждает *трехмерность* природы цвета. В качестве стимулов можно рассматривать, например, доминирующую длину волны (цветовой фон), насыщенность и яркость или красный, зеленый и синий цвета.

2. Цвета всегда *линейно зависимы*, т. е. $C = rR + gG + bB$. Следовательно, для смеси двух цветов C_1, и C_2 имеет место равенство

$$C_1 + C_2 = r_1R+g_1G+b_1B + r_2R +g_2G+b_2B = (r_1+r_2)R + (g_1+g_2)G + (b_1+b_2)B$$

При этом структура спектров цветов C_1 и C_2 значения не имеет.

3. Если в смеси трех цветов один непрерывно изменяется, а другие остаются постоянными, то цвет смеси будет меняться непрерывно, т. е. *трехмерное цветовое пространство непрерывно*.

Известно, что зрительная система способна различать примерно 350 000 цветов. Если цвета различаются только по тонам, то в центральной сине-желтой части спектра различными оказываются цвета, у которых доминирующие длины волн отличаются на 1 нм, в то время как у краев спектра — на 10 нм. Четко различимы примерно 130 цветовых тонов. Если меняется только насыщенность, то зрительная система способна выделить уже не так много цветов. Существует 16 степеней насыщенности желтого и 23 — красно-фиолетового цвета.

2.3 Цветовые модели

Трехмерная природа света позволяет представить совокупность цветов в виде некоторого пространства, причем каждый из компонентов будет представлен координатой этого пространства. Такое представление цвета называется ***цветовой моделью***.

Простейшими цветовыми моделями являются системы ***RGB*** и ***CMY***, представленные в виде «цветовых кубов». Любой цвет *C* можно представить как вектор в трехмерном пространстве, в котором координатные оси соответствуют интенсивности основных цветов *RGB* или *CMY*. Проекциями этого вектора на оси будут интенсивности его составляющих *r, g* и *b*, рис. 2.5.

На рисунке максимальная интенсивность основного цвета равна 1, а меньшие значения интенсивности выражаются дробными значениями в диапазоне 0…1. Такое представление удобно для расчетов и анализа, но на практике, при передаче цветных цифровых изображений не применяется, поскольку хранение чисел с плавающей запятой требует больших объемов памяти, чем целых чисел. Поэтому компоненты задаются либо в процентах (0…100), либо в диапазоне 0…255, что соответствует одному байту информации.

Началом координат в цветовом кубе *RGB* служит черный цвет, а в *CMY*— белый. *Основные* цвета в обоих случаях расположены по осям, а *дополнительные* цвета лежат в противоположных вершинах.

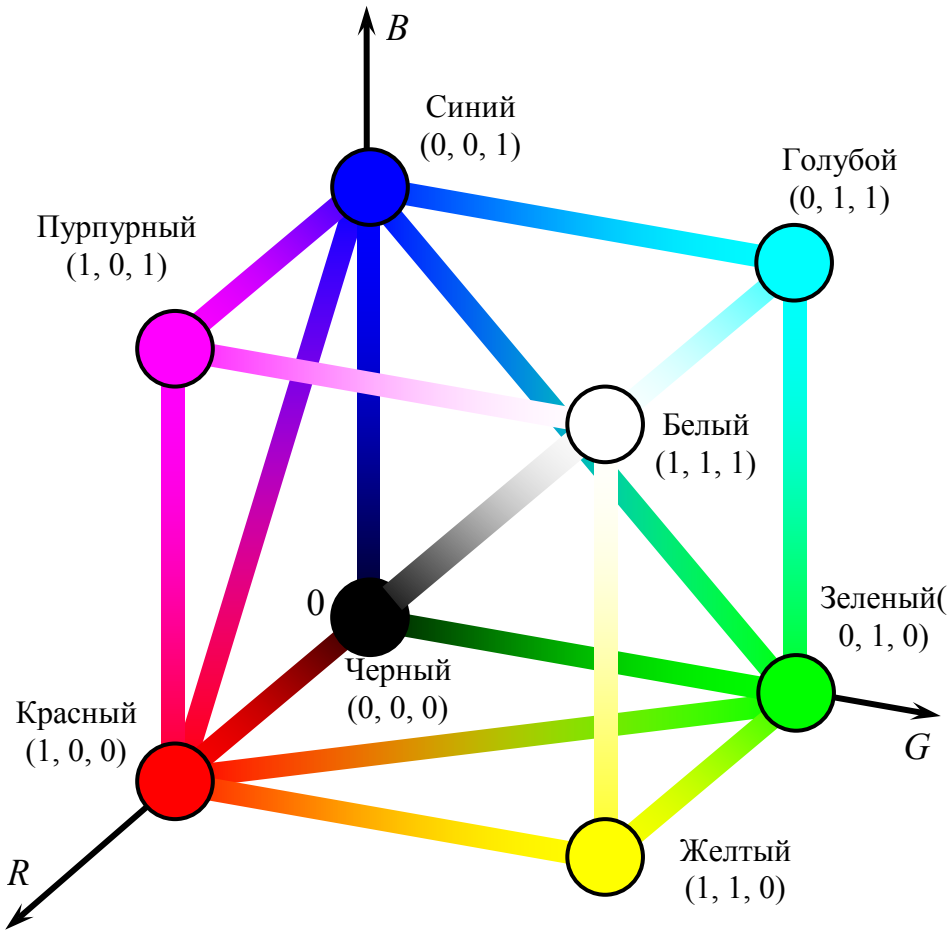


Рисунок 2.5 – Цветовой куб *RGB*

Преобразование между пространствами *RGB* и *CMY* выражается следующим образом:

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

Для того, чтобы выполнить это преобразование, нужно «перевернуть» куб, изображенный на рис. 2.5.

Ахроматические, т. е. серые цвета в обеих моделях расположены по главной диагонали куба — от черного до белого, — поскольку для них должно выполняться условие равенства интенсивностей трех составляющих:

$$r = g = b.$$

Плоскости, проведенные перпендикулярно главной диагонали куба, называются *плоскостями равной интенсивности*. Для всех точек, лежащих на такой плоскости, выполняется равенство

$$r + g + b = Br,$$

где *Br* — яркость (светлота), одинаковая для всех точек плоскости. На рис. 2.5 показана плоскость, проведенная через середину главной диагонали. Такая плоскость пересекает ребра куба в их серединах и образует *медианное сечение*.

Медианное сечение обладает весьма важными свойствами:

1. *Яркость* света в любой точке медианного сечения равна половине яркости белого света.

2. В центре такого сечения (точке, лежащей на главной диагонали) *насыщенность* цвета равна нулю — цвет ахроматический. По мере удаления от центра сечения *насыщенность* цвета возрастает и становится максимальной на его гранях.

3. На медианном сечении присутствуют все три основных и три дополнительных цвета, расположенных по его окружности в такой последовательности: красный, желтый, зеленый, голубой, синий, пурпурный.

Модели *RGB* и *CMY* удобны для машинной обработки, отображения и хранения изображений, поскольку в них явно заданы интенсивности компонент. Однако описывать субъективное восприятие цвета людьми в этих системах неудобно. Например, как в обозначениях *RGB* или *CMY* задать пастельный красновато-оранжевый цвет?

Художники характеризуют цвет с помощью таких понятий, как *разбелы*, *оттенки*, *тона*. Разбелы получают, добавляя в чистый цвет белый, оттенки — черный, тона — добавляя обе эти краски. Это можно представить в виде треугольника, рис. 2.6.

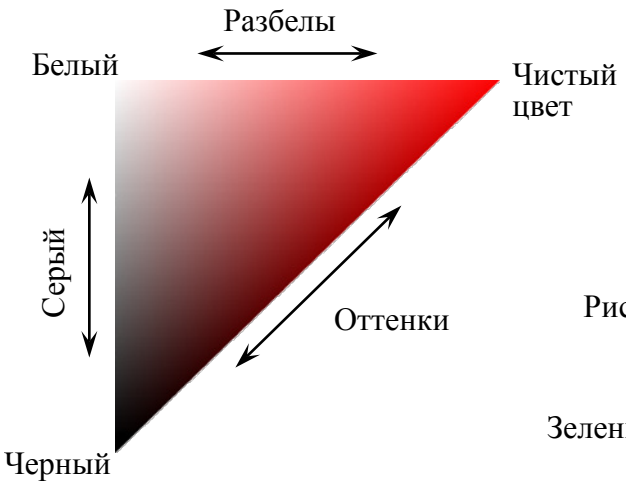


Рисунок 2.6 – Разбелы и оттенки чистого цвета

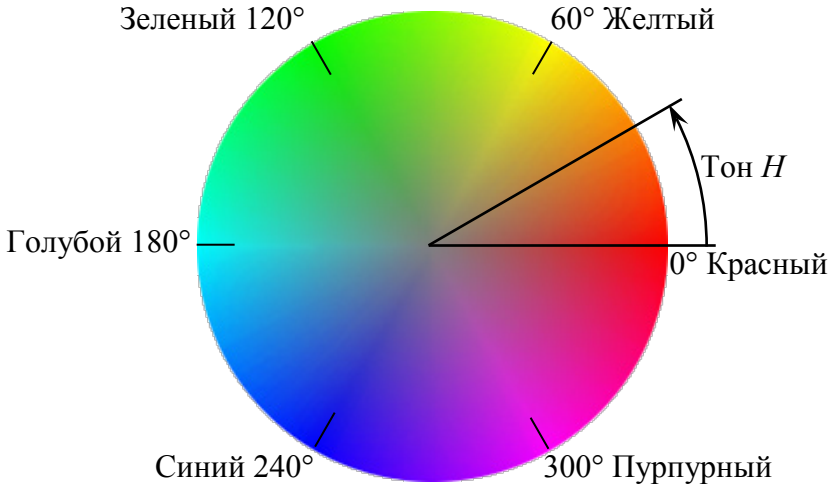


Рисунок 2.7 – Цветовые тона

Тоновую шкалу можно получить на основе медианного сечения цветового куба. Последнее представляет собой правильный шестиугольник, вершины которого — основные и дополнительные к ним цвета, а ребра — переходные тона, получаемые смешением основных в различных пропорциях. Заменив ребра шестиугольника дугами, получим цветовой круг, рис. 2.7. Приняв один из основных цветов (красный) за 0, получим шкалу, в которой цветовой тон задается углом в диапазоне 0 … 360°.

Если собрать треугольники разбелов и оттенков для всех чистых цветов вокруг центральной черно-белой оси, получим трехмерную модель субъективного представления цвета — **цветовой конус *HSB***, рис. 2.8 *а*. В этой модели цветовой тон H задается в градусах в соответствии со шкалой рис. 2.7, насыщенность S определяется расстоянием до оси конуса, а светлота B — расстоянием до вершины; величины S и B обычно задаются в диапазоне 0 … 100 %. Вершина конуса соответствует черному цвету (светлота $B = 0$), ось конуса — ахроматическим цветам от черного до белого (насыщенность $S = 0$). Поверхность конуса — чистые цвета ($S = 100$ %).

Модель *HSB* соответствует тому, как составляют цвета художники. Чистым пигментам отвечают значения $S = 100\%$, $B = 100\%$; разбелам — цвета с увеличенным содержанием белого, т. е. с меньшим S; оттенкам — цвета с уменьшенным B, которые получаются при добавлении черного. Поэтому модель *HSB* целесообразно применять для отражающих предметов, например типографских изображений.

а)

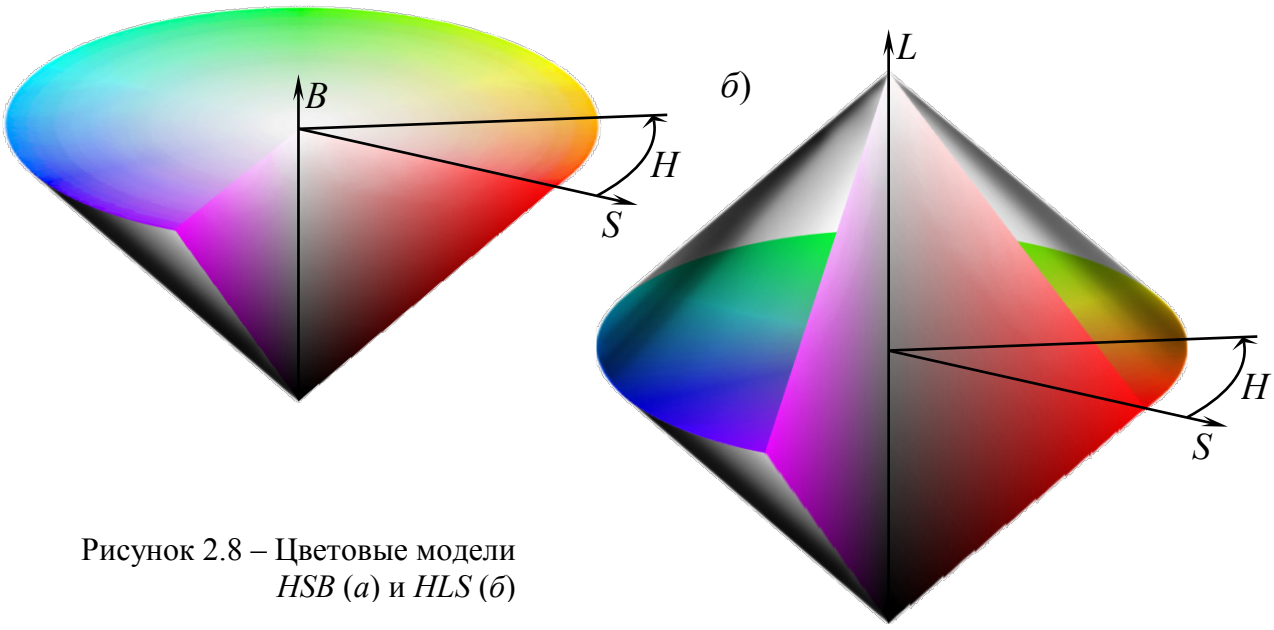


Рисунок 2.8 – Цветовые модели *HSB* (*а*) и *HLS* (*б*)

Цветовая модель ***HLS*** в виде двойного конуса является расширением одиночного конуса *HSB* и применяется для *самосветящихся* объектов, рис. 2.8 *б*. Светлота здесь обозначает яркость L источника света. Нижняя вершина двойного конуса соответствует нулевой яркости ($L = 0$), а верхняя — максимальной яркости источника ($L = 100$ %). Тон H и насыщенность S задаются так же, как и в *HSB*. Основное отличие *HLS* от *HSB* заключается в том, что чистый цвет в *HSB* получается при максимальной светлоте ($B = 100$ %), а в *HLS* — при средней яркости ($L = 50$ %). В этом модель *HLS* ближе к цветовому кубу *RGB*.

Существуют другие цветовые модели, рассчитанные на применение в различных областях техники. В телевидении системы NTSC применяется цветовая модель ***YIQ***, обеспечивающая возможность передачи цвета с помощью частотной модуляции и совместимая с черно-белым телевидением. Сигнал Y содержит информацию об уровне яркости и полностью аналогичен сигналу черно-белого телевидения. Сигналы I и Q несут информацию о цветовом тоне и насыщенности.

При подготовке изображений для цветной печати используется четырехкомпонентная система *CMYK*, отличающаяся от *CMY* тем, что параметры C, M и Y задают относительные содержания голубого, пурпурного и желтого цветов, а параметр K — светлоту изображения. Такая система соответствует принципу цветной печати с помощью красителей четырех цветов: голубого, пурпурного, желтого и черного.

3. КОДИРОВАНИЕ РАСТРОВОГО ИЗОБРАЖЕНИЯ. ОСОБЕННОСТИ РАСТРА

3.1 Растровые изображения и их основные характеристики

Растр — это матрица ячеек (пикселов). Любой ***пиксел*** имеет свой цвет. Совокупность пикселов различного цвета образует ***изображение***.

Для описания расположения пикселов используют разнообразные *системы координат*. Чаще всего используется система целых координат — номеров пикселов с (0, 0) в левом верхнем углу.

Растр имеет такие *характеристики*:

Тип растра — расположение пикселов в пространстве. Различают квадратный, прямоугольный, гексагональный и др. типы растра, рис. 3.1. В компьютерной технике наиболее распространен квадратный растр. Прямоугольный растр использовался в старых дисплейных системах (CGA, VGA и др.). Гексагональный растр используется в полиграфии.

Размер растра обычно измеряется количеством пикселов по горизонтали и вертикали, рис. 3.2.

Форма пикселов растра определяется особенностями устройства графического вывода (рис. 3.3). 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).

Разрешающая способность растра 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).

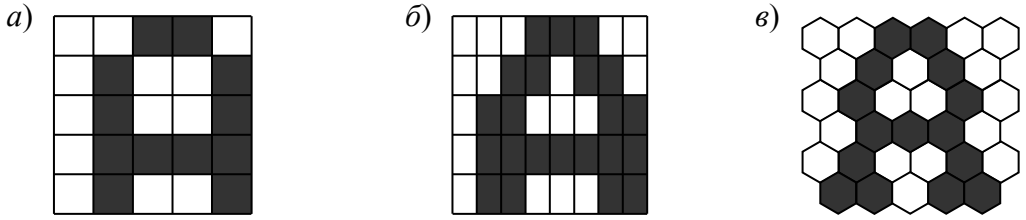


Рисунок 3.1 – Примеры квадратного (*а*), прямоугольного (*б*) и гексагонального (в) растров

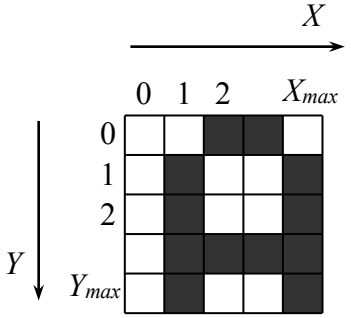


Рисунок 3.2 – Система координат и размер растра

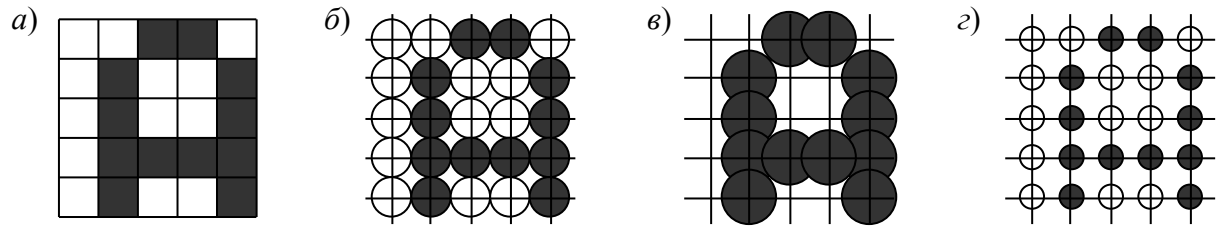


Рисунок 3.3 – Различные формы и размеры пикселов

Глаз человека с нормальным зрением способен различать объекты с угловым размером около одной минуты. Если расстояние между отдельными точками (пикселами) мало, то эти точки уже не воспринимаются как отдельные точки. 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. 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 92…120 dpi — это плохо; например, дисплей размером 17" по диагонали должен обеспечивать не 1024x768 пикселов, а вдвое больше. Но на современном уровне развития техники это пока что невозможно.

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

— *двухцветные* (*бинарные*) — пиксел может принимать один из двух цветов (обычно черный или белый). При этом на каждый пиксел приходится 1 бит памяти; двухцветные изображения обычно используются при печати на лазерном или другом нецветном принтере.

— *полутоновые* — градации яркости серого или другого цвета. Для передачи реалистических черно-белых изображений (например, фотографий) обычно используют 256 градаций яркости — при этом для кодирования информации о пикселе необходим 1 байт (8 бит) памяти.

— *цветные изображения* — пиксел может принимать различные цвета и оттенки; в зависимости от количества бит памяти, приходящихся на 1 пиксель, глубина цвета может составлять 8 (3 бита), 16 (4 бита), 256 (8 бит), 65 536 (16 бит) 16 777 216 (24 бита) и 4 294 967 296 (32 бита). Первые три из указанных используются для создания упрощенных, а последние два — фотореалистических изображений. Глубина в 65 536 цветов позволяет создать и обрабатывать изображение, близкое к фотореалистическому, при меньших затратах памяти и машинного времени.

3.2 Способы растровой развертки

Для вывода на видеомонитор разложенный в ***растр*** образ необходимо представить в виде того шаблона, который требует дисплей. Это преобразование называется ***растровой разверткой***.

В отличие от *дисплейного списка* для векторного дисплея, содержащего информацию только об отрезках или литерах, в данном случае дисплейный список должен содержать информацию о каждом *пикселе* на экране. Необходимо, кроме того, чтобы эта информация организовывалась и выводилась со скоростью *видеогенерации* в порядке сканирования строк, т. е. сверху вниз и слева направо.

Существует четыре способа достижения такого результата — *растровая развертка в реальном времени*, *групповое кодирование*, *клеточное кодирование* и *буфер кадра*.

1) Растровая развертка в реальном времени

При развертке в реальном времени или «на лету» сцена хранится в памяти в виде *дисплейного списка*, аналогичного векторным дисплеям. Во время воспроизведения каждого кадра процессор сканирует эту информацию и вычисляет интенсивность каждого пиксела на экране, рис. 3.4.

При такой развертке *не нужны большие количества памяти*. Требования на память обычно ограничиваются необходимостью хранить дисплейный список плюс одну сканирующую строку. Более того, поскольку информация о сцене хранится в произвольно организованном дисплейном списке, *добавление* или *удаление* информации из списка осуществляется легко, а это удобно для динамических приложений. Однако *сложность* выводимого изображения ограничивается скоростью дисплейного процессора. Обычно это означает, что ограничено число отрезков или многоугольников в картине, количество пересечений со сканирующей строкой или число цветов или полутонов серого.

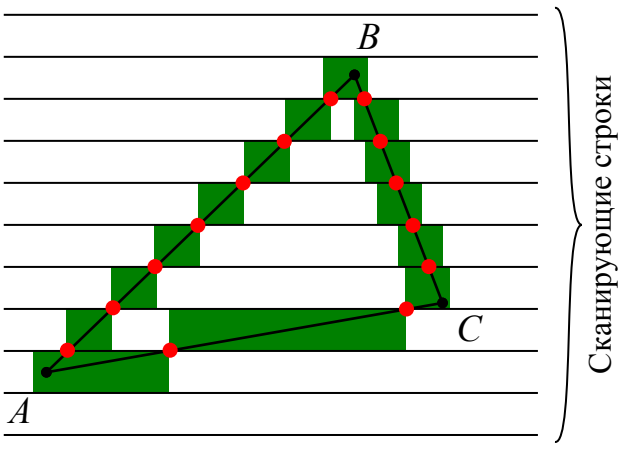


Рисунок 3.4 – Растровая развертка в реальном времени

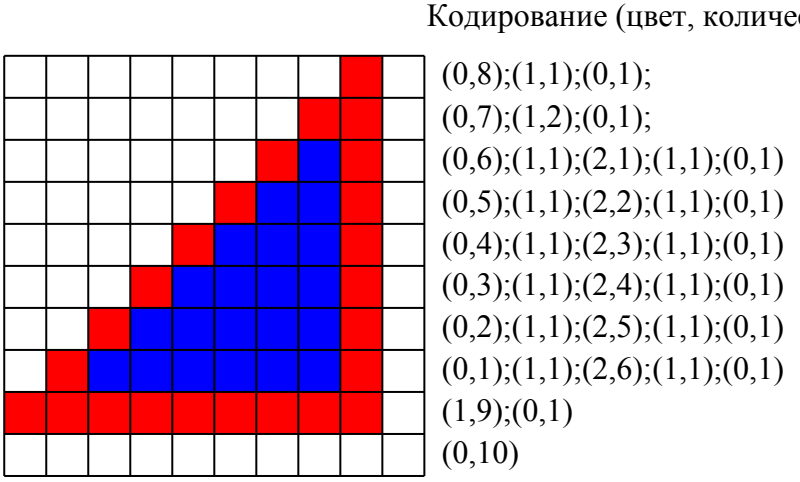


Рисунок 3.5 – Групповое кодирование

Для получения пересечений (если они есть) каждого отрезка дисплейного списка со сканирующей строкой в простейшей реализации метода всякий раз при изображении строки обрабатывается весь дисплейный список. При регенерации видеоизображения на каждую сканирующую строку, а значит, и на обработку всего списка приходится только 63,5 микросекунды. Столь малое время позволяет использовать данный метод только для рисования несложных чертежей.

2) Групповое кодирование

Метод группового кодирования применяется в случаях, когда большие области изображения имеют одинаковую интенсивность или цвет. При групповом кодировании определяется только интенсивность и количество последовательных пикселов с этой интенсивностью на данной сканирующей строке, рис.3.5.

Кодирующие данные следует рассматривать *группами* по два числа. Первое число — цвет или *интенсивность* пиксела, второе — *количество последовательных пикселов* на сканирующей строке.

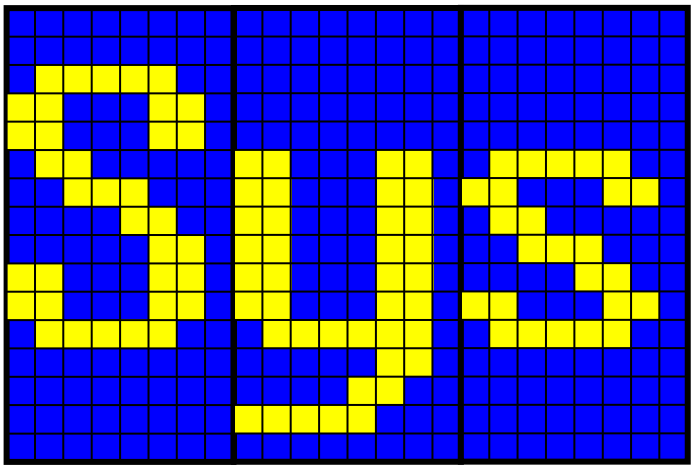
Таким образом, в строке 1 на рис. 3.2, имеется 8 пикселов цвета 0, т. е. черных или фоновых, 1 пиксел цвета 1 (красный) и затем еще 1 цвета 0.

Преимущества группового кодирования: небольшой объем занимаемой памяти (в сравнении с буфером кадра) и возможность отображения областей, заполненных цветом.

Недостатки группового кодирования:

—добавление или удаление отрезков или текста из изображения является трудоемкой операцией и занимает много времени из-за последовательного хранения длин участков;

— для изображений, содержащих большое количество мелких деталей может потребоваться в больше памяти, чем при попиксельном хранении.



Код символа: 83 121 115

Рисунок 3.6 – Клеточное кодирование

3) Клеточное кодирование

В методе клеточного кодирования с помощью минимума информации (один байт) кодируются целые *области изображения*, т. е. *клетки*, рис. 3.6.

В таком терминале область экрана разбивается на клетки или области, достаточно большие, чтобы содержать *одну литеру*. Например, экран можно разбить на области размером 8×8 или 8×16 пикселов. Для дисплея с разрешением 640 × 400 получится 50 или 25 строк, содержащих по 80 клеток.

Для вывода изображения на экран дисплейный контроллер использует две области памяти: *дисплейный буфер* и *область знакогенератора*. В ***дисплейном буфере*** хранятся коды символов или знаков, выводимых на экран. Каждый символ кодируется одним или двумя байтами (в последнем случае второй байт кодирует цвет символа и фона, другие характеристики символа, например мигание).

В области ***знакогенератора*** хранится ***маска символа*** — информация о его начертании. Для символов размером 8×8 или 8×16 пикселов на каждый символ приходится по 8 или 16 байт соответственно.

Помимо литер и знаков препинания знакогенератор содержит набор *графических символов*: горизонтальных и вертикальных линий, сопряжений, стрелок и т.д., позволяющих выводить на экран простейшие изображения — таблицы, схемы. Вывод более сложных графических изображений при таком способе кодирования невозможен.

4) Буфер кадра

Буфер кадра представляет собой большой непрерывный участок памяти, в котором для каждой точки или *пиксела*, отводится как минимум один бит памяти. Эта память называется ***битовой плоскостью***. Для квадратного растра размером 800×600 пикселов требуется 800×600 = 480000 бит или 60 килобайт памяти в одной битовой плоскости.

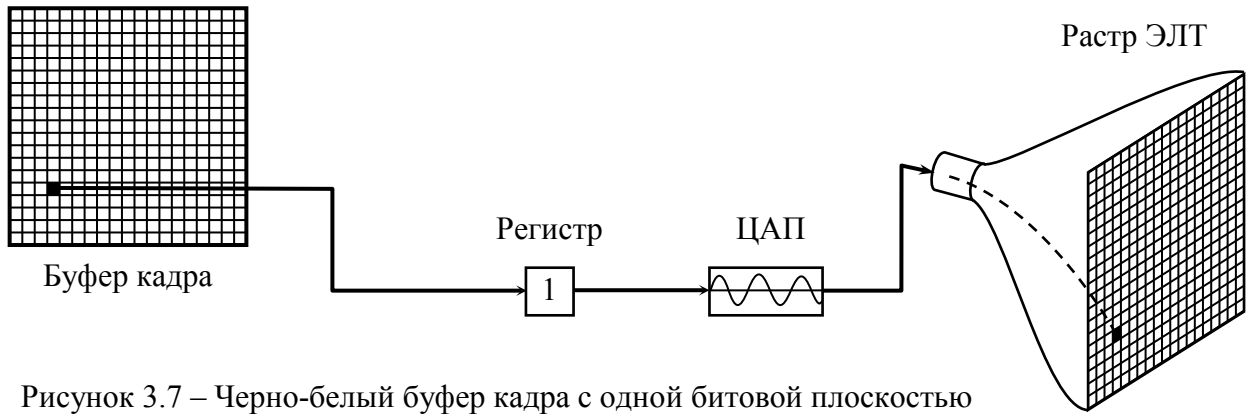


Рисунок 3.7 – Черно-белый буфер кадра с одной битовой плоскостью

Изображение в буфере кадра строится побитно. Из-за того что бит памяти имеет только *два состояния* (0 или 1), имея одну битовую плоскость, можно получить лишь черно-белое изображение.

Битовая плоскость является цифровым устройством, тогда как растровая ЭЛТ — аналоговое устройство, для работы которого требуется электрическое напряжение. Поэтому при считывании информации из буфера кадра и ее выводе на графическое устройство с растровой ЭЛТ должно происходить преобразование из цифрового представления в аналоговый сигнал. Такое преобразование выполняет *цифро-аналоговый преобразователь* (ЦАП).

Каждый пиксел буфера кадра должен быть считан и преобразован, прежде чем он будет отображен на растровой ЭЛТ. На рис. 3.7 приведена схема графического устройства с черно-белой растровой ЭЛТ, построенного на основе буфера кадра с одной битовой плоскостью. Информация из буфера кадра поступает в ***регистр***, а затем ЦАП преобразует содержимое регистра в аналоговый электрический сигнал, который подается на ЭЛТ.

3.3 Кодирование цветов и полутонов

Цвета или *полутона серого* цвета могут быть введены в буфер кадра путем использования *дополнительных битовых плоскостей*.

На рис. 3.8 показана схема буфера кадра с N битовыми плоскостями для ***градаций серого цвета***. *Интенсивность* каждого пиксела на ЭЛТ управляется содержимым ячеек памяти в каждой из N битовых плоскостей.

В соответствующую позицию *регистра* загружается бинарная величина (0 или 1) из каждой плоскости. Двоичное число, получившееся в результате, интерпретируется как *уровень интенсивности* между 0 (темный экран) и 2^N–1 (максимальная интенсивность свечения), т.е. всего можно получить 2^N уровней интенсивности. С помощью ЦАП это число преобразуется в напряжение.

Рис. 3.8 иллюстрирует систему с тремя битовыми плоскостями для $2^3 = 8$ *уровней интенсивности*. Для передачи реалистичного черно-белого изображения используется большее количество битовых плоскостей — до 8. При этом число уровней интенсивности составляет $2^8 = 256$.

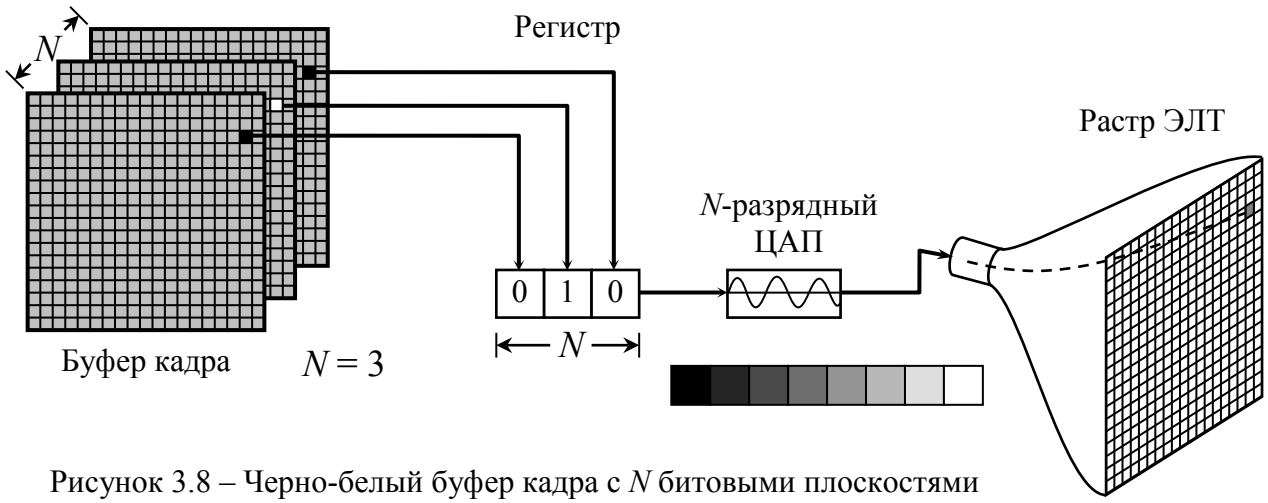
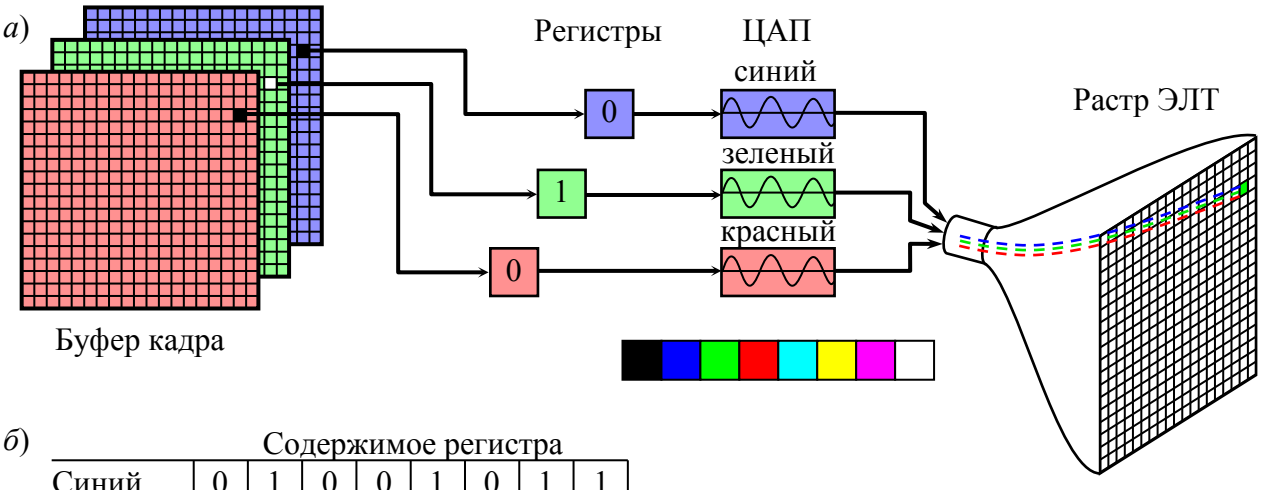


Рисунок 3.8 – Черно-белый буфер кадра с N битовыми плоскостями

Для каждой битовой плоскости требуется полный объем памяти при данном разрешении растра: например, буфер кадра с тремя битовыми плоскостями для растра 800×600 занимает 800×600×3 = 1440000 бит или 180 килобайт памяти, а при 8 битовых плоскостях — 480 килобайт.

Поскольку существует три основных цвета, наиболее простой способ реализации ***цветного изображения*** — цветной буфер кадра с тремя битовыми плоскостями, по одной для каждого из основных цветов, рис. 3.9 *а*. Каждая битовая плоскость управляет индивидуальной электронной пушкой для каждого из трех основных цветов, используемых в видеотехнике. Три основных цвета, комбинируясь на ЭЛТ, дают восемь цветов. Эти цвета и соответствующие им двоичные коды приведены на рис. 3.9 *б*. Изображение, получаемое при таком способе кодирования, имеет весьма низкое цветовое разрешение.

а)



б)

	Содержимое регистра							
Синий	0	1	0	0	1	0	1	1
Зеленый	0	0	1	0	1	1	0	1
Красный	0	0	0	1	0	1	1	1
Цвет пикселя	Черный	Синий	Зеленый	Красный	Голубой	Желтый	Пурпурный	Белый
		Основные			Комбини-рованные			

Рисунок 3.9 – Простой цветной буфер кадра с тремя битовыми плоскостями (*а*) и трехбитовое кодирование цвета (*б*)

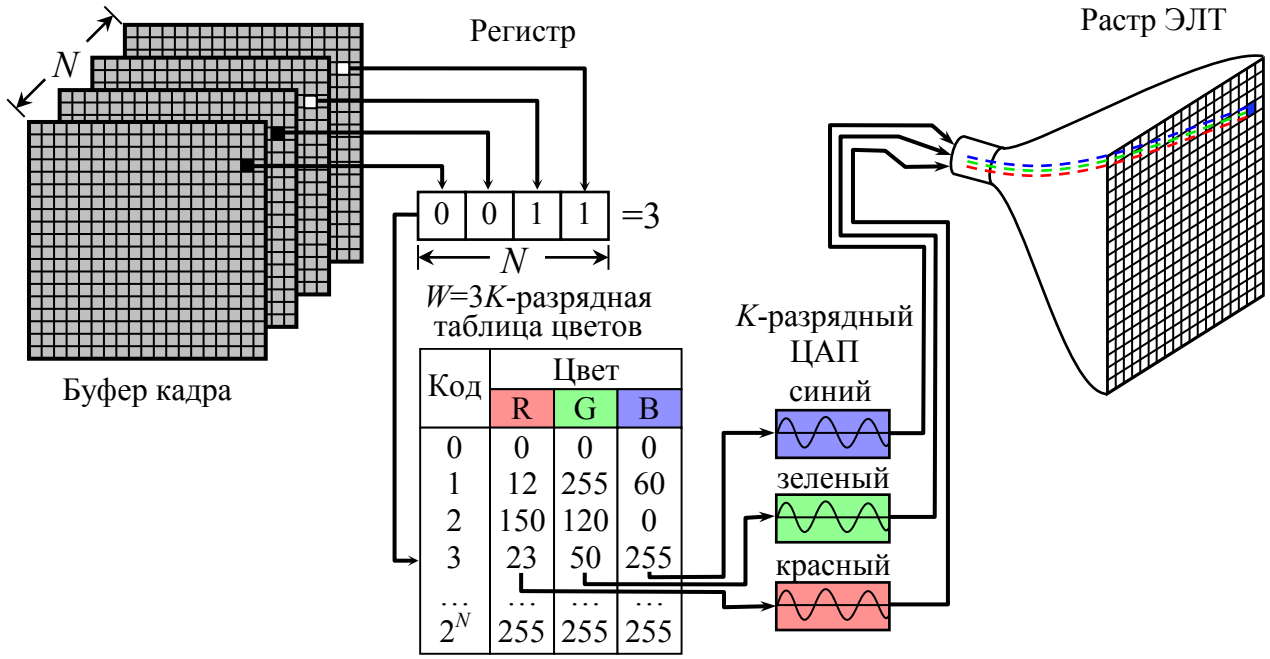


Рисунок 3.10 – Буфер кадра с N битовыми плоскостями и $W = 3K$-разрядной таблицей

Число доступных цветов можно увеличить, воспользовавшись ***таблицей цветов***, рис. 3.10. После считывания из буфера кадра битовых плоскостей получившееся число используется как индекс в таблице цветов. В этой таблице содержится 2^N элементов, совокупность которых называется ***палитрой***. Каждый ее элемент может содержать W бит кодировки цвета, причем $W > N$. В таком случае можно получить 2^W значений цветов, но *одновременно* могут быть доступны лишь 2^N из них. Для получения других значений таблицу цветов не-

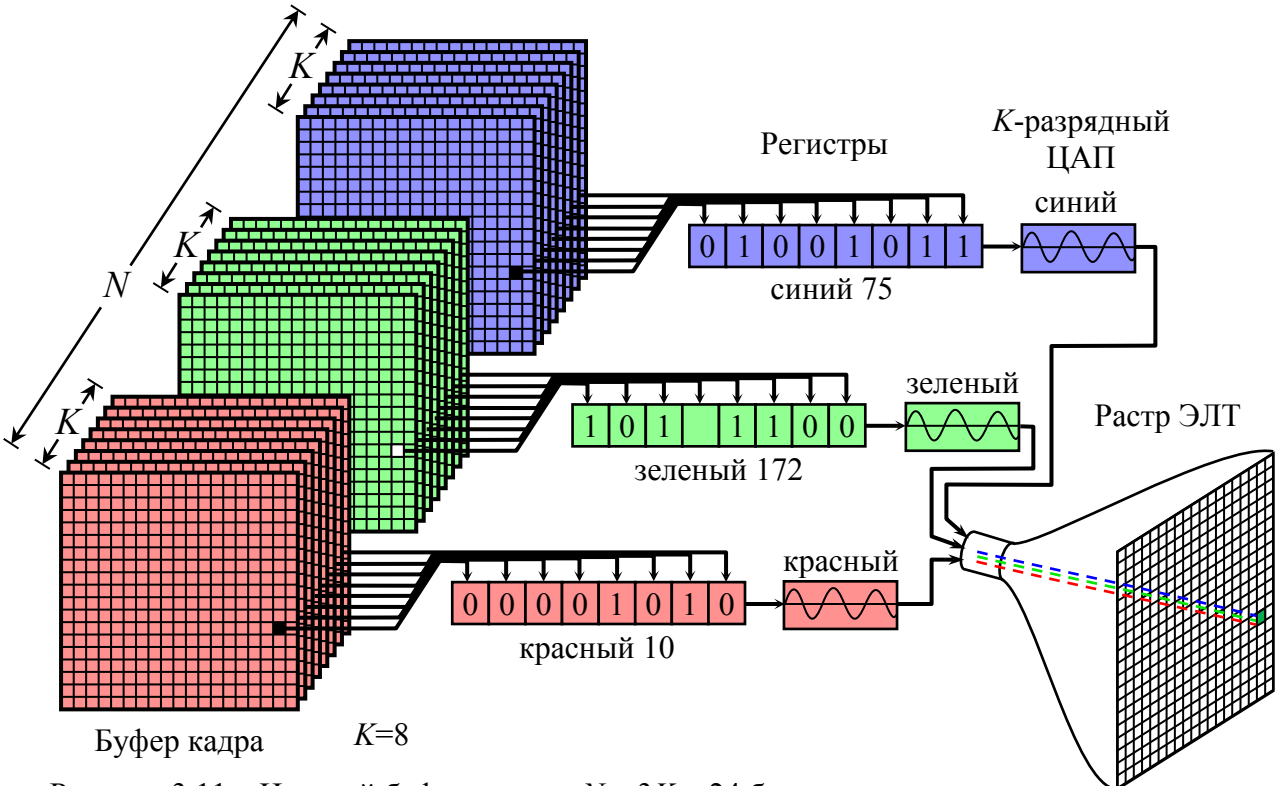


Рисунок 3.11 – Цветной буфер кадра с $N = 3K = 24$ битовыми плоскостями

обходимо изменить — загрузить в нее другую палитру. Чаще всего цвета в палитре задаются по модели RGB, тогда разрядность таблицы цветов $W = 3K$, где K – разрядность кодирования интенсивности каждого канала.

Для отображения ***реалистичных цветных изображений*** необходимо иметь возможность регулировать интенсивность каждого из трех основных цветов: красного, синего и зеленого. Для этого нужно, чтобы каждой из трех цветовых пушек соответствовало несколько битовых плоскостей.

На рис. 3.11 показан цветной буфер кадра с $K = 8$ битовыми плоскостями на каждый цвет, то есть с $N = 8 \times 3 = 24$ битовыми плоскостями. Каждая группа битовых плоскостей управляет 8-разрядным ЦАП и может генерировать $2^8 = 256$ уровней интенсивностей красного, зеленого или синего цвета. Их можно скомбинировать в $2^{24} = 256^3 = 16\ 777\ 216$ цветов. Это «*полноцветный*» буфер кадра.

3.4 Основные форматы растровых графических файлов

Для хранения и передачи растровых изображений существует множество форматов, различающихся структурой записи информации об изображении, максимальным размером растра и цветовым разрешением, возможностью записи нескольких изображений или дополнительной информации и т.д. Одно из основных отличий разных форматов, определяющее их назначение и область применения — возможность и способ ***сжатия информации***. По этому признаку растровые форматы можно разделить на три группы:

— *без сжатия* информации об изображении — растр записывается в файл в таком же виде, в каком он содержится в буфере кадра;

— со *сжатием без потерь* — используются алгоритмы сжатия, не приводящие к потере информации и позволяющие восстановить изображение точно таким же, как до сжатия;

— со *сжатием с потерями* — используются алгоритмы сжатия, приводящие к потере части информации, при этом изображение после восстановления несколько искажается.

К первой группе относится формат ***BMP***, являющийся базовым графическим форматом *OS Windows*. Файлы формата *BMP* могут хранить информацию об изображениях с глубиной цвета от 1 до 24 бит на пиксел — двухцветных, полутоновых, цветных с таблицей цветов и полноцветных. Помимо самого растра, файл имеет заголовок, в котором содержится информация о размере растра, количестве цветов (битов на пиксел) и разрешающей способности. Последняя используется для вывода изображения на печать. Также для неполноцветных изображений файл *BMP* содержит таблицу цветов — палитру.

Поскольку в формате *BMP* отсутствует сжатие, файлы обрабатываются с высокой скоростью и распознаются без ошибок различными программными пакетами, однако занимают значительный объем памяти — для больших

полноцветных изображений это десятки мегабайт. Это затрудняет хранение и делает практически невозможным передачу по каналам связи, например Internet.

Ко второй группе относятся форматы, использующие относительно простые методы сжатия. Примером может служить рассмотренный выше метод *группового кодирования*, который используется в форматах ***PCX***, ***TGA*** и ***TIFF*** и позволяет в несколько раз уменьшить объем файла. Такой метод сжатия наиболее эффективен для изображений типа чертежей, схем — которые имеют большие области, закрашенные одним цветом. Сжатие таким методом сложных полноцветных изображений, в которых каждый пиксел отличен от соседних (например, фотографий) неэффективно и может привести не к уменьшению, а к увеличению объема файла.

Существуют иные способы сжатия без потерь, например *LZ*-алгоритмы, использующие *словарный метод сжатия*. Создается словарь, содержащий повторяемые последовательности значений — *фразы*, находящиеся в исходном массиве данных. Каждая фраза получает код (индекс) в словаре. Кодирование массива осуществляется заменой фраз соответствующими индексами. Алгоритм обеспечивает большую степень сжатия, чем групповое кодирование, но работает медленнее.

LZ-алгоритм сжатия используется в формате ***GIF***, разработанном для обмена растровыми изображениями в сети *Internet*. Формат поддерживает изображения с глубиной цвета до 256 (8 бит на пиксел) и может хранить несколько изображений для создания анимации — последовательной смены изображений через некоторый интервал времени.

Для сжатия изображений типа фотографий разработаны *алгоритмы сжатия информации с потерями*, позволяющие найти и удалить ту часть информации, которая существенным образом не влияет на восприятие реального изображения. При этом после восстановления изображение несколько отличается от исходного, но достаточно «похоже» на него, чтобы человеческий глаз не находил разницы.

В настоящий момент наиболее распространенным является метод ***JPEG*** (*Joint Photographic Experts Group*), разработанный в 1991 году. Сущность метода заключается в том, что изображение:

1) преобразуется из цветовой модели *RGB* в цветовую модель *YCbCr*, которая используется в цветном телевидении; канал яркости *Y* несет значительно больше информации, чем цветовые каналы *Cb* и *Cr*, поэтому последние можно подвергать наибольшему сжатию;

2) разбивается на блоки размером 8×8 пикселов и каждый блок подвергается *дискретному косинусному преобразованию* (*ДКП*), преобразующему пространственное распределение в частотное; в результате получаем блоки 8×8, каждый элемент которых является частотным коэффициентом спектра;

3) информация в блоках фильтруется путем деления на матрицу квантования и последующего округления частотных коэффициентов до целого;

а) *б*)

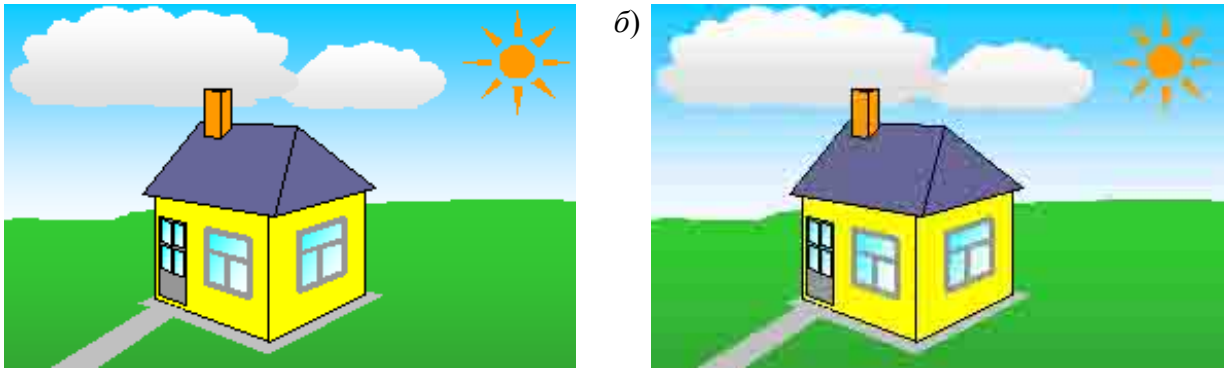


Рисунок 3.27 – Искажения при *JPEG*-сжатии: исходное (*а*) и сжатое (*б*) изображения

при этом информация о высоких частотах в спектре обнуляется, и, таким образом, изображение «упрощается» — остаются только низкие частоты, несущие основную информацию; большинство же элементов блока 8×8 будут равны нулю;

4) информация в блоках кодируется методом группового сжатия.

Метод сжатия *JPEG* позволяет уменьшить объем файла в 10…30 раз, при этом степень сжатия можно задавать — чем больше степень сжатия, тем меньшим будет полученный файл, но большими — искажения. Недостаток метода — искажения картинки при больших степенях сжатия, которые проявляются в виде заметных глазу блоков — «квадратиков», а также размытых границ и ореолов вокруг контуров объектов, рис. 3.27. Искажения наиболее заметны при сжатии изображений с четкими границами — чертежей, схем, поэтому для таких изображений использовать метод *JPEG* не рекомендуется.

Модифицированный метод сжатия *JPEG* используется также для сжатия кадров видеоизображения.

В настоящее время все большее распространение получает метод рекурсивного сжатия изображений, называемый также волновым алгоритмом (*wavelet*-преобразованием). Метод требует больших, чем *JPEG*, объемов памяти и затрат процессорного времени на обработку, но позволяет получить более качественное изображение при той же степени сжатия, что и *JPEG*, поскольку не разбивает картинку на блоки. Изображение становится немного «размытым», но искажения не так заметны. Волновой алгоритм реализован в новом стандарте ***JPEG* 2000**, а также в некоторых других.

Весьма популярным становится формат ***DjVu***, разработанный для размещения в *Internet* документов, сканированных в цвете с высокой разрешающей способностью. Основной принцип *DjVu* — разделение изображения на фон и передний план. Формат использует волновой алгоритм для сжатия подложки (фона) и изображений типа фотографий, а для текста и графики используется эффективный алгоритм сжатия без потерь для черно-белых и палитровых изображений. Формат *DjVu* обеспечивает очень высокую степень сжатия как черно-белых, так и цветных изображений, при этом искажения практически отсутствуют. В файл *DjVu* может быть записано несколько изображений, что позволяет хранить в электронном виде журналы, книги, и т.п.

4. МЕТОДЫ И АЛГОРИТМЫ ДВУМЕРНОЙ ГРАФИКИ

4.1 Координатный метод. Преобразование координат

Фундаментом компьютерной графики является ***аналитическая геометрия***, в основе которой лежит ***координатный метод*** — задание положения точек и объектов на плоскости или в пространстве с помощью ***системы координат***. Наиболее часто используются *прямоугольная* или *Декартова* система координат, что связано как с ее широким применением в аналитической геометрии, так и с тем, что координаты пикселов в дисплеях и печатающих устройствах также задаются в прямоугольной системе координат.

Рассмотрим особенности координатного метода на плоскости. Положение точки в этом случае задается двумя числами — координатами x и y, рис. 4.1 *а*. Положение некоторого объекта может быть задано координатами некоторой базовой точки, являющейся центром его собственной (*локальной*) системы координат, и углом поворота этой системы координат относительно *глобальной системы координат*, связанной с экраном дисплея или листом бумаги в принтере. Пересчет координат точек из одной в другую систему координат называется ***преобразованием координат***.

Пусть существует некий объект, имеющий свою систему координат (x, y), рис. 4.1 *б*. Форма объекта определяется набором точек, положение которых задано координатами в системе (x, y). Для того, чтобы отобразить его в некотором положении на экране, листе бумаги или в некотором более сложном изображении, необходимо пересчитать координаты этих точек в глобальную систему координат (X, Y). Пусть положение объекта задано координатами центра его локальной системы координат (X_0, Y_0) и углом поворота α. Рассмотрим три частных случая преобразования координат.

1. ***Сдвиг*** — центр локальной системы координат смещен в точку (X_0, Y_0), угол поворота равен нулю, рис. 4.1 *в*. Координаты некоторой точки A объекта в новой системе координат составят:

$$\begin{cases} X = x + X_0, \\ Y = y + Y_0, \end{cases} \tag{4.1}$$

где x, y – координаты точки A в системе координат объекта.

2. ***Поворот*** вокруг центра координат локальной системы на угол α, рис. 4.1 *г*. :

$$\begin{cases} X = x\cos\alpha - y\sin\alpha, \\ Y = x\sin\alpha + y\cos\alpha. \end{cases} \tag{4.2}$$

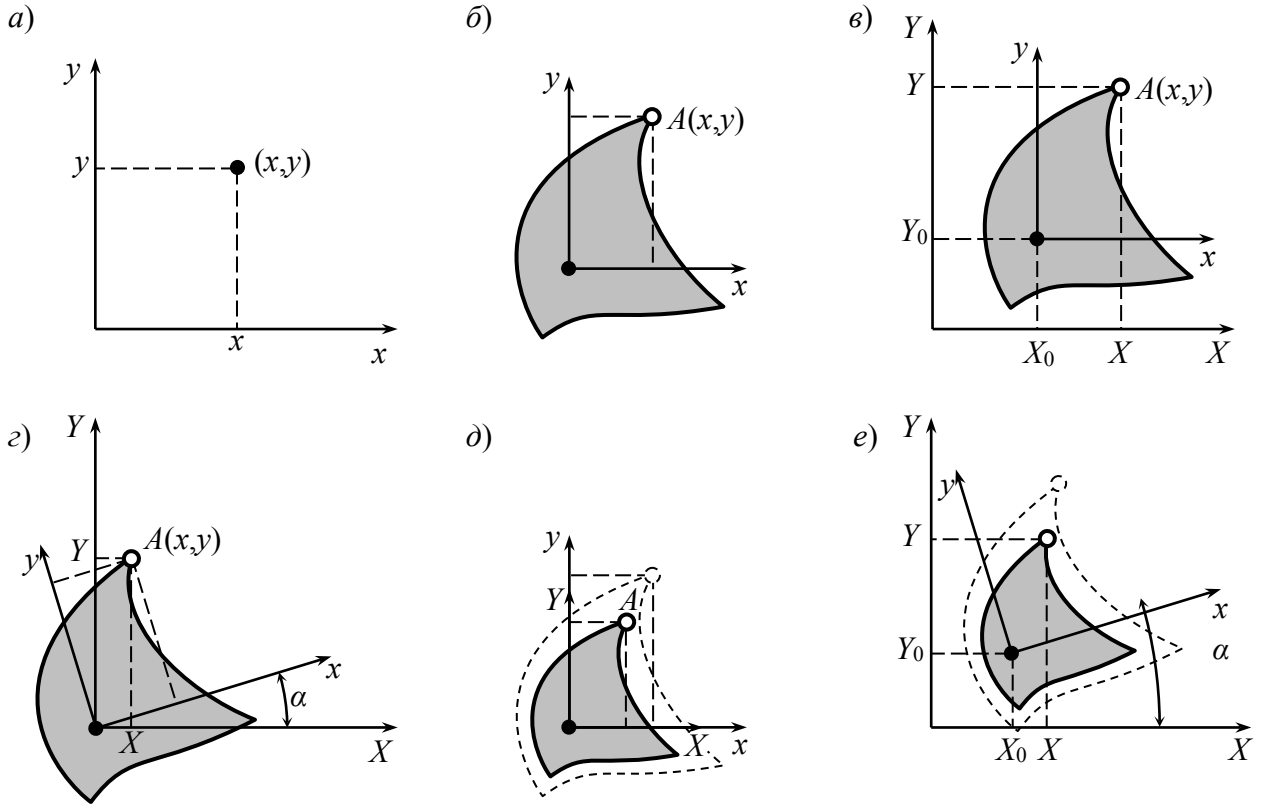


Рисунок 4.1 – Координатный метод и преобразование координат

3. ***Масштабирование*** (*сжатие/растяжение*) с центром в начале координат — увеличение или уменьшение объекта, определяемое коэффициентами масштабирования k_x и k_y, рис. 4.1 *д*.:

$$\begin{cases} X = x\,k_x, \\ Y = y\,k_y. \end{cases} \tag{4.3}$$

Если $k_x = k_y$ — объект масштабируется пропорционально; в противном случае происходит деформация объекта — неодинаковое растяжение или сжатие по разным направлениям.

Как видно из рис. 4.1 *е*., последовательно применяя масштабирование, поворот и смещение координат, можно расположить некоторый объект (геометрический примитив или фрагмент изображения) в произвольном положении. При этом формулы для пересчета координат будут иметь вид:

$$\begin{cases} X = x\,k_x \cos\alpha - y\,k_y \sin\alpha + X_0, \\ Y = x\,k_x \sin\alpha + y\,k_y \cos\alpha + Y_0. \end{cases} \tag{4.4}$$

Подобные преобразования координат в аналитической геометрии называют *аффинными*. В общем виде они могут быть записаны в виде системы уравнений

$$\begin{cases} X = A\,x + B\,y + C, \\ Y = D\,x + E\,y + F. \end{cases} \tag{4.5}$$

Преобразования координат удобно выполнять в матричном виде. Константы A, B, …, F образуют матрицу преобразования, которая, будучи умноженной на матрицу-столбец координат (x, y), дает матрицу-столбец (X, Y). Чтобы учесть константы C и F, необходимо перейти к так называемым *однородным координатам* — добавить еще одну строку со значением, равным 1. Тогда (4.5) можем записать в виде

$$\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = \begin{bmatrix} A & B & C \\ D & E & F \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \tag{4.6}$$

или

$$\boldsymbol{W} = \boldsymbol{A}\,\boldsymbol{w}, \tag{4.7}$$

где $\boldsymbol{w}$ и $\boldsymbol{W}$ – матрицы координат в исходной и новой системах; $\boldsymbol{A}$ – матрица преобразования. Например для операций смещения, поворота и масштабирования получим матрицы соответственно

$$\boldsymbol{A} = \begin{bmatrix} 1 & 0 & X_0 \\ 0 & 1 & Y_0 \\ 0 & 0 & 1 \end{bmatrix}; \tag{4.8}$$

$$\boldsymbol{B} = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}; \tag{4.9}$$

$$\boldsymbol{C} = \begin{bmatrix} k_x & 0 & 0 \\ 0 & k_y & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{4.10}$$

Тогда преобразование (4.4) запишем как

$$\boldsymbol{W} = \boldsymbol{C}\,\boldsymbol{B}\,\boldsymbol{A}\,\boldsymbol{w}. \tag{4.11}$$

4.2 Алгоритмы вывода линий

Многие современные программы (векторные графические редакторы, САПР, математические и другие специальные программы) работают с изображениями, представленными некоторыми *математическими моделями*. Наиболее часто такие изображения составлены из стандартного набора простых элементов, называемых ***графическими примитивами***. Это могут быть прямые и кривые линии, фигуры (прямоугольники, окружности, эллипсы) и т.п. Для того, чтобы описать такое изображение, состоящее из примитивов, достаточно перечислить эти примитивы, указать их параметры (цвет, толщину и стиль линий, способ заливки фигур и др.) и взаимное расположение.

Поскольку практически все современные средства отображения графики (дисплеи, принтеры) растровые, для отображения картинки, заданной в векторном виде, необходимо преобразовать ее в *растр*. Такое преобразование обычно выполняется той программой, в которой изображение было создано, с использованием алгоритмов ***растеризации***.

Для того, чтобы преобразовать в растр линию, необходимо изменить цвет пикселов, по которым эта линия проходит, рис. 4.2 *а*. Чтобы это сделать, нужно получить уравнение линии. Очевидно, наиболее простым графическим примитивом является ***отрезок прямой***, который может быть задан координатами двух точек: начала (x_1, y_1) и конца — (x_2, y_2). Для произвольной точки (x, y), лежащей на отрезке прямой, должно выполняться условие

$$\frac{x - x_1}{y - y_1} = \frac{x_2 - x_1}{y_2 - y_1}. \qquad (4.12)$$

Отсюда можем получить уравнение прямой как функцию y от x или наоборот:

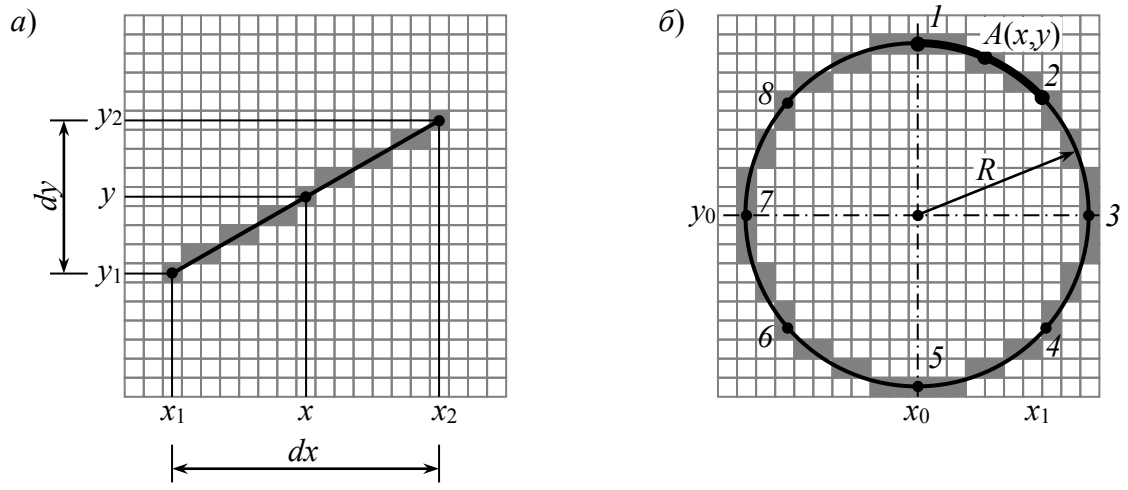


Рисунок 4.2 – Растеризация отрезка прямой (*а*) окружности (*б*)

$$y = y_1 + (x - x_1)\frac{y_2 - y_1}{x_2 - x_1}. \tag{4.13}$$

$$x = x_1 + (y - y_1)\frac{x_2 - x_1}{y_2 - y_1}. \tag{4.14}$$

Очевидно, что дробь в уравнении представляет собой угловой коэффициент и при построении может быть вычислена один раз. Для растеризации отрезка следует выяснить, какая из величин $dx = x_2 - x_1$ и $dy = y_2 - y_1$ больше, и построить «график» функции $y = f_1(x)$ в первом или $x = f_2(y)$ во втором случае. Выполняется простой цикл по соответствующей координате от начальной до конечной точки с шагом в один пиксел. Другая координата определяется расчетом по (4.13) или (4.14) и округляется до целого значения; пиксел с полученными координатами (x, y) закрашивается заданным цветом.

Для точек ***окружности*** выполняется уравнение

$$(x - x_0)^2 + (y - y_0)^2 = R^2, \tag{4.15}$$

где R – радиус окружности, а (x_0, y_0) – координаты ее центра, рис. 4.2 *б*. Преобразовав это выражение, получим зависимость

$$y = y_0 \pm \sqrt{R^2 - (x - x_0)^2}, \tag{4.16}$$

Поскольку окружность симметрична относительно вертикальной и горизонтальной осей, проходящих через ее центр, необязательно рассчитывать все точки окружности. Достаточно выполнить расчет для дуги *1-2* — октанта (1/8) окружности, — для чего выполняется цикл по координате x от x_0 до x_1 с шагом в один пиксел и вычисляются и округляются до целого соответствующие значения y. Затем закрашивается пиксел $A(x, y)$ и еще семь пикселов, расположенных на дугах *2-3*, *3-4*, …, *8-1*.

Для построения ***эллипса*** существует два способа. Первый — аналогичный вышеописанному, при этом используется уравнение эллипса

$$\frac{(x - x_0)^2}{a^2} + \frac{(y - y_0)^2}{b^2} = 1, \tag{4.17}$$

где a и b – половины длин осей эллипса. Расчет выполняется не для октанта, а для всего квадранта, остальные 3 квадранта строятся симметрично.

Второй способ — использование алгоритма построения окружности с последующим масштабированием — «растягиванием» окружности пропорционально длинам осей эллипса.

Описанные выше способы построения линий просты для понимания, но их реализация связана с использованием операций с плавающей точкой, умножения и деления, извлечения квадратного корня. Такие операции выполняются процессором относительно медленно, что приводит к низкой скорости отображения, особенно на маломощных ЭВМ.

Существуют ***инкрементные алгоритмы растеризации***, называемые еще алгоритмами ***Брезенхэма***, позволяющие строить отрезки прямых, окружностей, эллипсов и других кривых, используя только операции целочисленного сложения и вычитания. Сущность алгоритма Брезенхэма проиллюстрируем на примере построения отрезка, рис. 4.3:

1. Определяем величины приращений: $dx = x_2 - x_1$; $dy = y_2 - y_1$.

2. Выбираем большее из приращений: $dx > dy$, значит $d = dx$.

3. Вводим текущие координаты x и y, а также вспомогательные переменные ex и ey. До начала цикла принимаем $x = x_1$; $y = y_1$; $ex = 0$; $ey = 0$. Закрашиваем пиксел с координатами (x, y)

4. Выполняем d раз цикл, в котором переменные ex и ey увеличиваем соответственно на dx и dy; проверяем: если $ex > d$ то выполняем $ex = ex - d$ и $x = x + 1$; если $ey > d$ то $ey = ey - d$ и $y = y + 1$; получаем координаты (x, y) следующего пиксела; закрашиваем пиксел (x, y).

Аналогичные, но несколько более сложные алгоритмы используются для построения окружностей, эллипсов, дуг. Как видно, в приведенном алгоритме используются только операции сложения, вычитания и сравнения. Поскольку последние выполняются во много раз быстрее операций с плавающей точкой, такие алгоритмы обеспечивают высокую скорость отображения даже на маломощных ЭВМ.

К недостаткам алгоритма можно отнести некоторую «неточность» засветки пикселов, что видно из рис. 4.3, а также то, что опорные точки отрезков и других фигур могут иметь лишь целочисленные координаты (середины соответствующих пикселов).

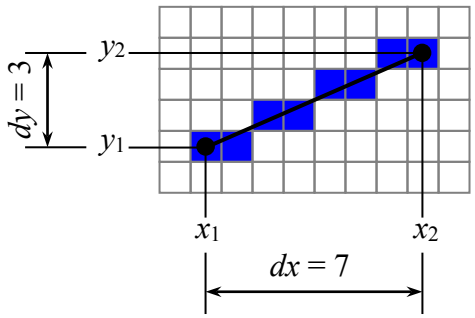


Рисунок 4.3 – Растеризация отрезка прямой алгоритмом Брезенхэма

4.3 Кривые Безье и NURBS

Разработаны 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*: $\{x = f_x(t), y = f_y(t)\}$. 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 $f_x(t)$ и $f_y(t)$. Кроме того $f(x)$ может быть неоднозначной. Например уравнение окружности в параметрическом виде: $\{x(\varphi) = x_0 + R\cos\varphi;\ \ y(\varphi) = y_0 + R\sin\varphi\}$, где φ – некоторый угол, — сравните с выражением (4.16).

Многочлены Безье в общем имеют такой вид:

$$\begin{cases} x(t) = \sum_{i=0}^{m} C_i t^i (1-t)^{m-i} x_i; \\ y(t) = \sum_{i=0}^{m} C_i t^i (1-t)^{m-i} y_i, \end{cases} \tag{4.17}$$

где x_i и y_i – координаты точек-ориентиров P_i, а величины C_i – коэффициенты бинома Ньютона. Значение m можно рассматривать и как степень полинома, и как значение, которое на единицу меньше количества точек-ориентиров. Параметр t задает точку на кривой — в начальной точке кривой $t = 0$, в конечной $t = 1$, каждому значению t в диапазоне от 0 до 1 соответствует некоторая точка на кривой.

Как видно из выражений, вид кривой Безье при заданной степени m однозначно определяется расположением точек-ориентиров P_i. Именно эта особенность делает кривые Безье универсальным и чрезвычайно удобным графическим примитивом.

Рассмотрим кривые Безье, классифицируя их по значениям m и количеству точек-ориентиров.

m = 1 (по двум точкам):

$$\begin{cases} x(t) = (1-t)\, x_0 + t\, x_1; \\ y(t) = (1-t)\, y_0 + t\, y_1. \end{cases} \tag{4.18}$$

Кривая вырождается в отрезок прямой линии, которая определяется конечными точками P_0 и P_1, как показано на рис. 4.4 *а*.

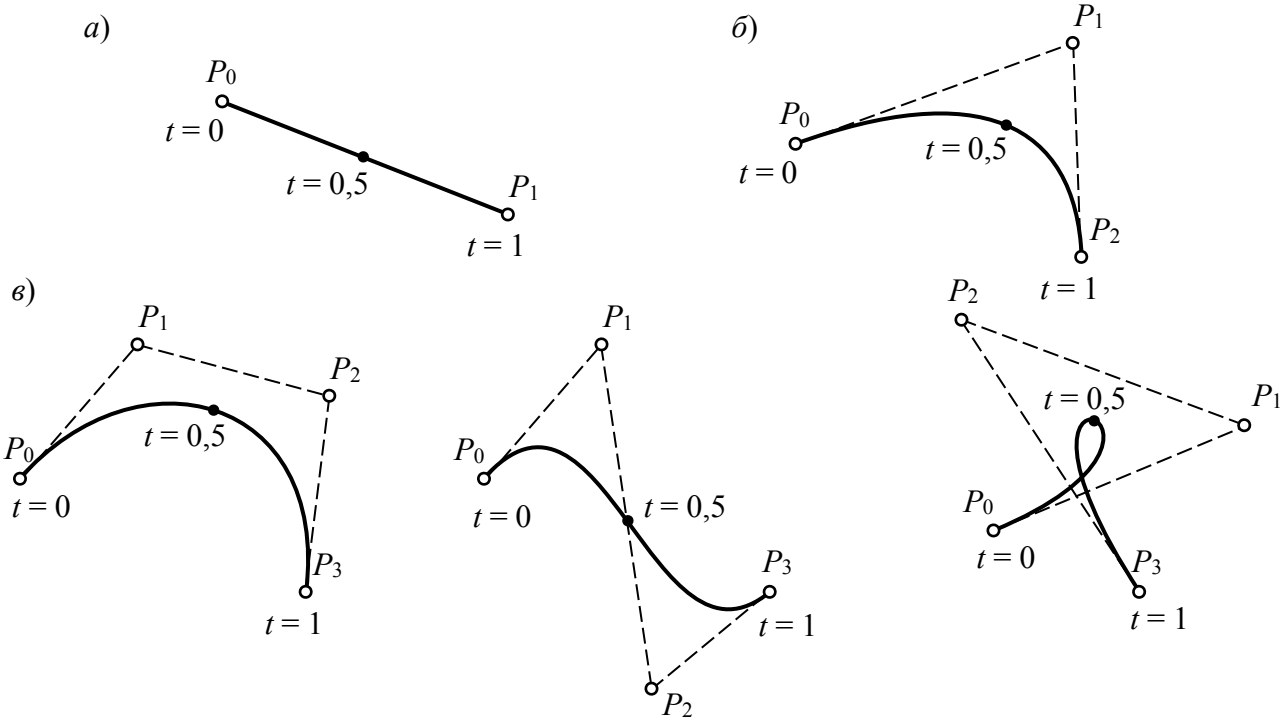


Рисунок 4.4 – Кривые Безье первого (*а*), второго (*б*) и третьего (*в*) порядков

m = 2 (по трем точкам):

$$\begin{cases} x(t) = (1-t)^2 x_0 + 2t(1-t) x_1 + t^2 x_2; \\ y(t) = (1-t)^2 y_0 + 2t(1-t) y_1 + t^2 y_2. \end{cases} \tag{4.19}$$

Получаем квадратичную кривую, показанную на рис. 4.4 *б*.

m = 3 (по четырем точкам):

$$\begin{cases} x(t) = (1-t)^3 x_0 + 3t(1-t)^2 x_1 + 3t^2(1-t) x_2 + t^3 x_3; \\ y(t) = (1-t)^3 y_0 + 3t(1-t)^2 y_1 + 3t^2(1-t) y_2 + t^3 y_3. \end{cases} \tag{4.20}$$

Кривая Безье 3-го порядка может иметь весьма разнообразную форму, рис. 4.4 *в*, что позволяет использовать ее для построения различных линий, фигур, контуров произвольной формы. Кривые Безье являются основными графическими примитивами большинства векторных графических редакторов «художественного» назначения (например, Corel Draw, Adobe Illusttrator), а также используются в инженерных и архитектурных пакетах (AutoCAD, ArchiCAD, КОМПАС-3D) для моделирования объектов сложной формы.

Для построения кривой Безье можно использовать зависимости (4.19) и (4.20), рассчитав координаты для набора значений t от 0 до 1 с некоторым шагом. Однако существует алгоритм построения кривых Безье, не требующих выполнения сложных вычислений, возведения в степень. Это *геометрический алгоритм*, позволяющий найти координаты (x, y) точки кривой Безье по значению параметра t:

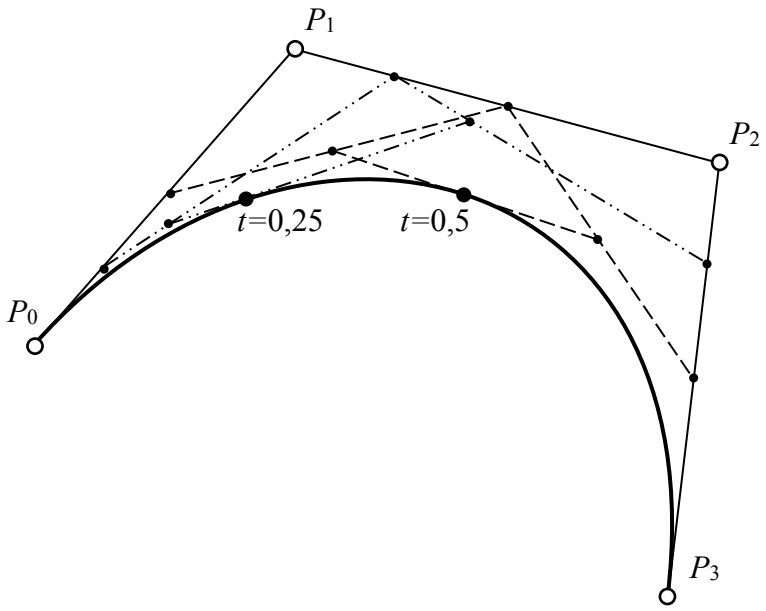


Рисунок 4.5 – Геометрический алгоритм построения кривых Безье

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. Это продолжается до тех пор, пока не будет получена единственная точка деления. Эта точка и будет точкой кривой Безье.

На рис. 4.5 показаны построения для $t = 0{,}5$ (пунктирная линия) и для $t = 0{,}25$ (штрих-штрих-пунктирная линия). Процедура нахождения середины отрезка весьма проста, выполняется посредством простых целочисленных операций, поэтому алгоритм может быть легко реализован на ЭВМ и обеспечивает высокую скорость построения.

Для создания сложных кривых линий используются ***полилинии Безье*** — несколько кривых Безье, состыкованных вместе и образующих единый графический примитив, рис. 4.6. Полилинии Безье могут быть как *разомкнутыми*, так и *замкнутыми.* В последнем случае получаем ***контур*** с криволинейными границами, который может быть, например, заполнен некоторым цветом с помощью алгоритма заполнения.

Точки, в которых выполняется стыковка отдельных кривых Безье (точки P_0, P_3, P_6 и P_9 на рис. 4.6.) называются ***узлами*** полилинии. В зависимости от способа стыковки соседних кривых узлы полилинии Безье могут быть:

— ***гладкими*** (стыковка кривых с выравниванием по первой и второй производным) — в таких узлах выполняются условия совпадения направлений

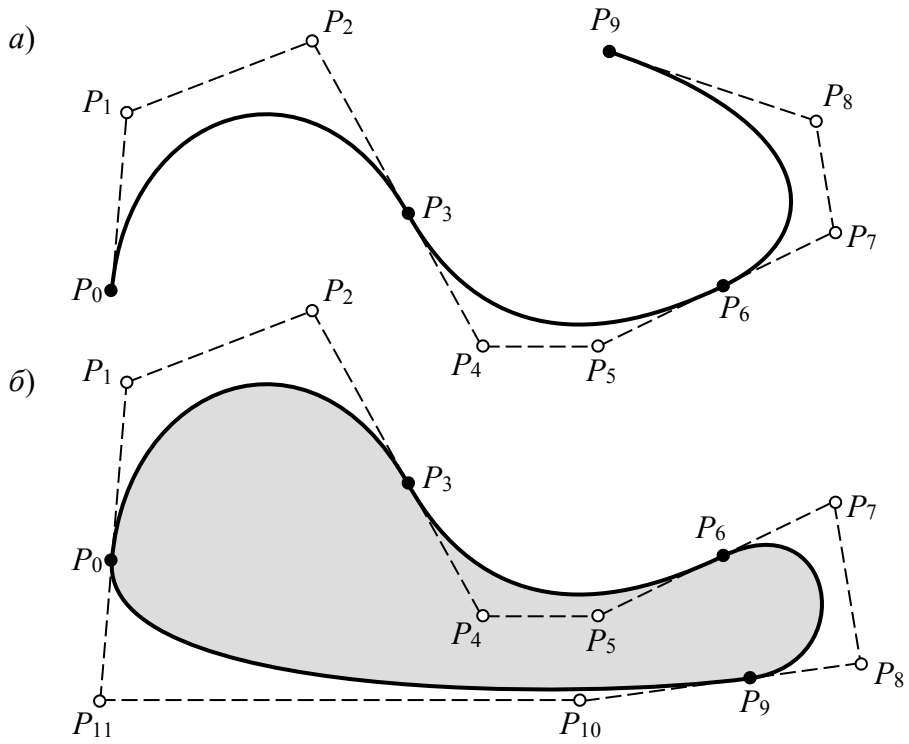


Рисунок 4.6 – Разомкнутая (*а*) и замкнутая (*б*) полилинии Безье

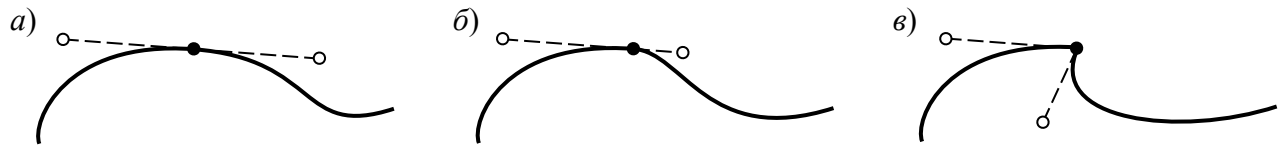


Рисунок 4.7 – Гладкий (*а*), прямой (*б*) и угловой (*в*) узлы полилинии Безье

касательных к обеим кривым и одинаковых радиусов кривизны обеих кривых в точке стыковки;

— ***прямыми*** (стыковка с выравниванием по первой производной) — выполняется условие совпадения направлений касательных к обеим кривым в точке стыковки, радиусы кривизны могут быть различными;

— ***угловыми*** (произвольная стыковка) — и направления касательных к обеим кривым и радиусы кривизны могут быть различными.

Обобщение методов Безье и B-сплайнов в начале 70-х годов позволило получить одно из мощнейших и универсальных средств геометрического моделирования криволинейных обводов — ***NURBS-технологию***.

Аббревиатура ***NURBS*** обозначает ***Non-Uniform Rational B-Splines***, то есть ***неоднородные рациональные B-сплайны***. Это математические объекты для задания двумерных кривых и гладких поверхностей в трехмерном пространстве.

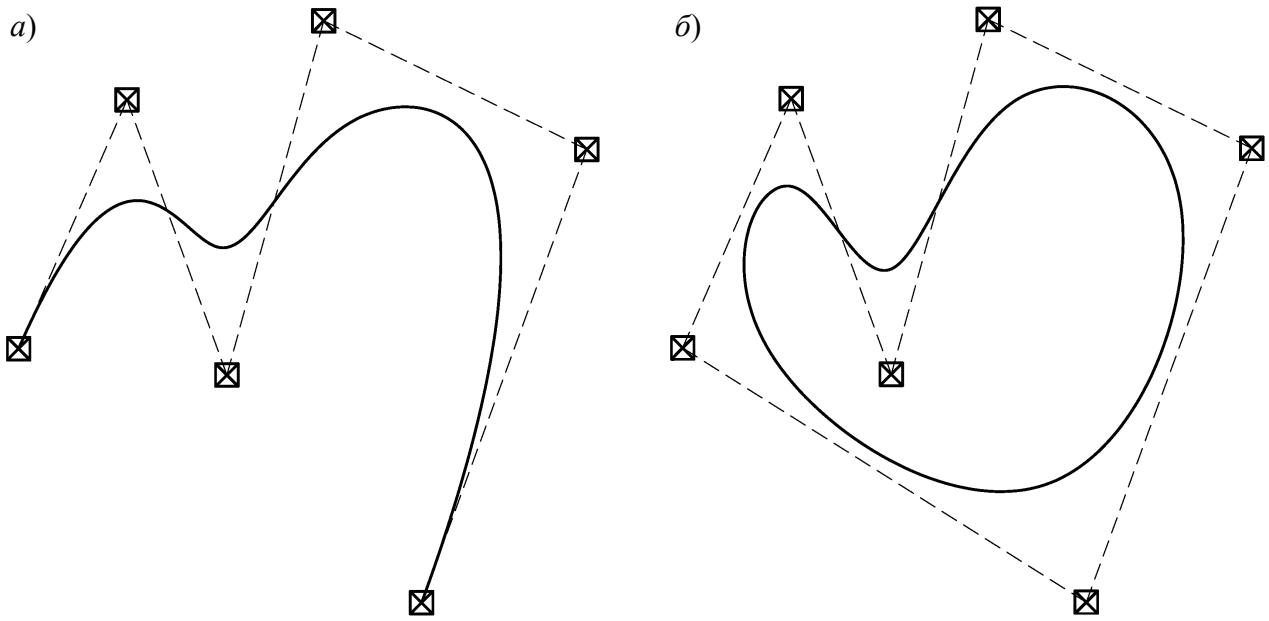


Рисунок 4.8 – Разомкнутая (*а*) и замкнутая (*б*) *NURBS*-кривые

В отличие от кривых Безье, *NURBS-кривые* могут быть заданы ***любым числом точек-ориентиров***, могут быть как разомкнутыми, так и *замкнутыми*, рис. 4.7. Точки-ориентиры кривых или поверхностей NURBS могут обладать различными ***весами***, т. е. в различной степени «притягивать» к себе кривую (на что указывает слово «*неоднородные*» в названии кривых).

Из-за своей гибкости и точности ***NURBS-модели*** могут использоваться в любом процессе иллюстрации, анимации и промышленного дизайна. Большинство современных САПР, систем компьютерной графики и анимации поддерживают моделирование с использованием *NURBS*-кривых и поверхностей, поскольку:

— с помощью NURBS проще имитировать поверхности природных объектов или объектов, поверхности которых имеют сложным образом искривленные профили (например, обводы корпуса автомобиля);

— NURBS-модели обеспечивают лучшее качество визуализации объектов благодаря разбиению на грани, выполняемому с использованием аналитических выражений (слово «*рациональные*» означает, что объект *NURBS* может быть описан с помощью математических формул).

4.4 Алгоритмы закрашивания фигур

Фигурой 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.

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.

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;

— алгоритмы, которые используют *математическое описание контура*.

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.

Алгоритм закрашивания линиями получил широкое распространение в компьютерной графике. Сущность его заключается в том, что на каждом шаге закрашивания рисуется горизонтальная линия, которая размещается между пикселами, имеющими цвет контура. Затем проводится анализ цвета пикселов на соседних строчках, и если цвет пикселов не равен цвету пикселов контура, процедура заполнения линии вызывается снова для следующей строки. Закрашивание начинается в некоторой точке A, координаты которой нужно задать при вызове процедуры, и заканчивается, когда внутри контура не остается ни одного незакрашенного пиксела, рис. 4.9.

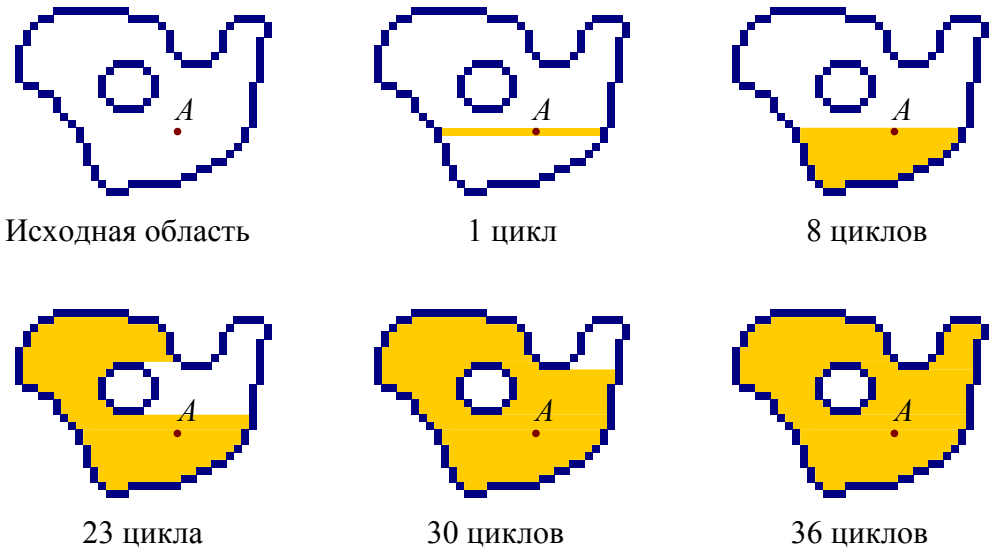


Рисунок 4.9 – Закрашивание области с помощью алгоритма закрашивания линиями

Основная область применения алгоритмов закрашивания произвольного контура — растровые графические редакторы (*Paint*, *Adobe Photoshop* и др.), где процесс обработки изображения (в частности, задание начальной точки), контролируется человеком. Для использования в программах, выполняющих растеризацию векторных изображений, такие алгоритмы слишком медленные и не обеспечивают стопроцентной гарантии правильного закрашивания всего контура. Алгоритмы заполнения, применяемые для таких целей, 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.

Среди всех фигур наиболее просто заполнять ***прямоугольник***. Если прямоугольник задан координатами противоположных углов, например, левого верхнего (x_1, y_1) и правого нижнего (x_2, y_2), тогда алгоритм может состоять в последовательном рисовании горизонтальных линий заданного цвета рис. 4.10 *а*.

Для заполнения ***круга*** можно использовать алгоритм вывода контура, рассмотренный выше. 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. 4.10 *б*. Так же можно создать и алгоритм заполнения эллипса.

Контур ***полигона*** определяется вершинами, которые соединены отрезками прямых — ребрами, рис. 4.10 *в*. Рассмотрим один из наиболее популярных алгоритмов заполнения полигона. Его основная идея — закрашивание фигуры отрезками горизонтальных линий. Алгоритм представляет собою цикл вдоль оси y, в ходе этого цикла выполняется поиск точек пересечения линии контура с соответствующими горизонталями. В алгоритме использовано топологическое свойство контура фигуры, состоящее в том, что любая прямая линия пересекает любой замкнутый контур четное количество раз.

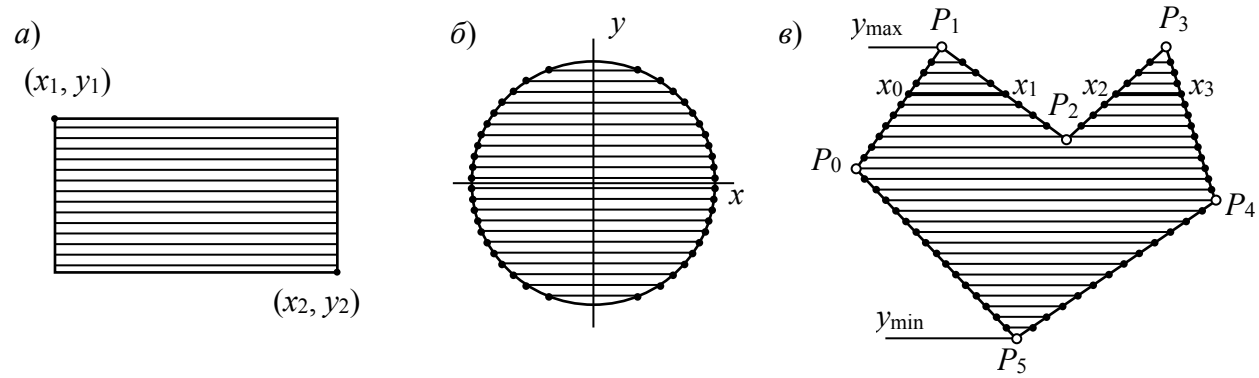


Рисунок 4.10 – Заполнение прямоугольника (*а*), круга (*б*) и полигона (*в*)

Для определения координат x точек пересечения для каждой горизонтали необходимо перебирать все n ребер контура. Координата x пересечения ребра $P_i - P_{i+1}$ с горизонталью y равняется

$$x = x_i + \left(y_{i+1} - y\right)\frac{x_{i+1} - x_i}{y_{i+1} - y_i}. \qquad (4.21)$$

Полученный в результате перебора ребер массив точек пересечения упорядочивается по возрастанию, (x_0, x_1, x_2, x_3, см. рис. 4.10 *в*.) и разбивается на пары, которые соединяются отрезками горизонтальных линий.

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.

4.5 Стиль линии. Перо. Алгоритмы вывода толстой линии

Возможности алгоритмов, описанных в п. 4.2, ограничены выводом линии толщиной в 1 пиксел. Очевидно, что для построения качественных изображений этого недостаточно. Необходимо иметь возможность строить линии произвольной ширины.

Для описания разных по виду изображений на основе линий используют термины ***стиль линий*** и ***перо***. Термин *перо* иногда делает более понятной суть алгоритма вывода линий для некоторых стилей — в особенности для толстых линий. Например, если для тонкой непрерывной линии перо соответствует одному пикселу, то для толстых линий перо можно представить себе как фигуру или отрезок линии, который скользит вдоль оси, оставляя за собой след, рис. 4.11.

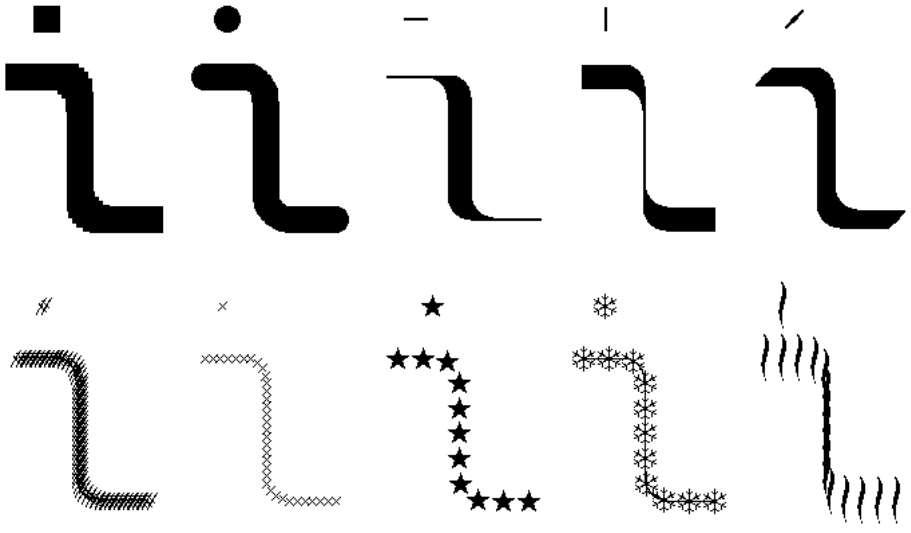


Рисунок 4.11 – Линия, нарисованная с различными формами пера

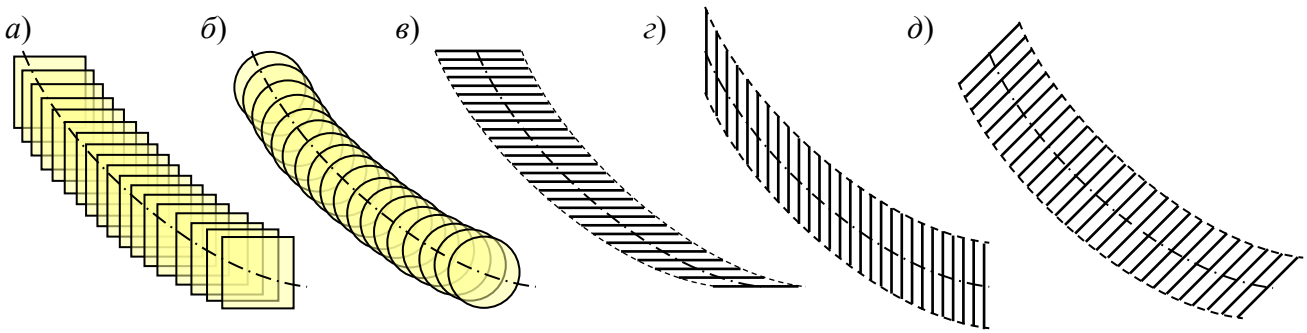


Рисунок 4.12 – Перья прямоугольной (*а*) и круглой (*б*) формы, перья в виде горизонтального (*в*), вертикального (*г*) и наклонного (*д*) отрезков

Взяв за основу любой алгоритм вывода обычных тонких линий (например, алгоритм Брезенхэма), его можно модифицировать для вывода толстой линии, если вместо рисования отдельного пиксела с координатами (x, y) выполнять вывод в этом месте фигуры или линии, которые соответствуют форме пера — прямоугольника, круга, отрезка прямой, другой фигуры.

Такой подход к разработке алгоритмов толстых линий имеет преимущества и недостатки. Преимущество в простоте — можно почти без изменений использовать уже известные эффективные алгоритмы для вычисления координат точек линии оси, например, алгоритмы Брезенхэма. Недостаток — неэффективность для некоторых форм пера. Для перьев, которые соответствуют фигурам с заполнением, количество тактов работы алгоритма пропорционально квадрату толщины линии. При этом делается много лишней работы — большинство пикселов многократно закрашивается в одних и тех же точках рис. 4.12 *а*, *б*.

Такие алгоритмы более эффективны для перьев в виде отрезков линий. В этом случае любой пиксел рисуется только один раз. Но здесь важен наклон изображаемой линии. Ширина пера зависит от наклона рис. 4.12 *в*, *г*, *д*. Заметим, что горизонтальное перо не может рисовать толстую горизонтальную линию.

Для вывода толстых линий с помощью пера в качестве отрезка линии чаще всего используются отрезки горизонтальной линии или вертикальной, реже — диагональные отрезки под углом 45 градусов. Целесообразность использования такого способа определяется большой скоростью вывода горизонтальных и вертикальных отрезков прямой. При этом чтобы минимизировать количество тактов вывода, толстые линии, которые ближе по наклону к вертикальным, рисуют горизонтальным пером, а линии, которые ближе по наклону к горизонтали, рисуют вертикальным пером.

В современных программах, осуществляющих растеризацию векторных изображений, для рисования толстых линий используют *алгоритмы заполнения фигур*. Например толстую ломаную линию можно представить в виде полигона, рис. 4.13 *а*. Для этого вычисляются координаты точек пересечения линий, отстоящих от осевой линии звеньев ломаной на величину $h/2$ (половину

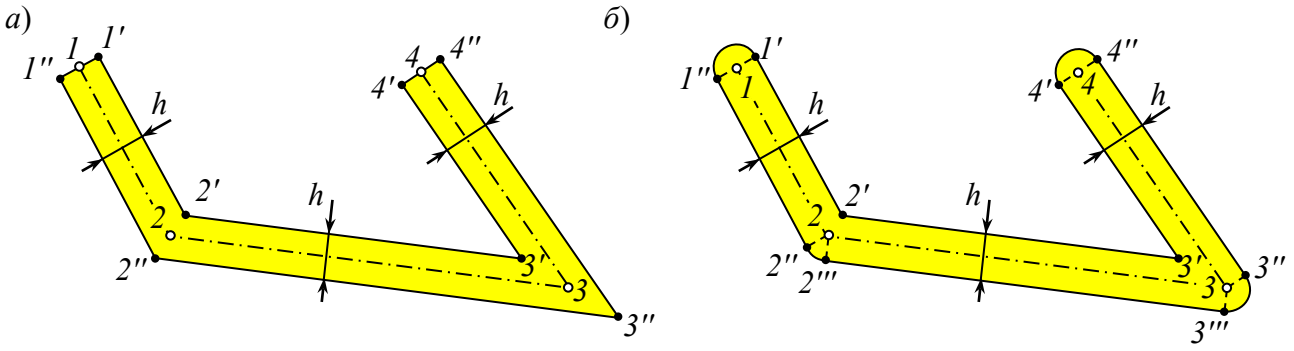


Рисунок 4.13 – Рисование толстой линии методом заполнения полигона (*а*) и фигуры с криволинейными элементами (*б*)

толщины линии) — точки *2'*, *3'*, *2''*, *3''*, а также линий, перпендикулярных оси и проходящих через концы ломаной — точки *1'*, *1''*, *4'* и *4''*. Все эти точки образуют контур полигона, который закрашивается стандартным алгоритмом.

Основной недостаток такого подхода — острые углы, особенно заметные, если звенья ломаной расположены под малым углом друг к другу. Во многих программах векторной графики используют усовершенствованные алгоритмы, обеспечивающие сглаживание острых углов дугами окружностей. Для придания линии более естественного вида, ее концы также могут скругляться, рис. 4.13 *б*.

Несмотря на то, что описанный способ построения линий требует значительно большего объема вычислений (необходимо вычислить координаты точек полигона и выполнить алгоритм его заполнения), и, следовательно, имеет меньшую скорость вывода изображения, он получил весьма большое распространение в современных графических пакетах, так как значительно более гибок, позволяет легко масштабировать линии, менять их толщину, изменять разрешение растра (например, при выводе изображения на печать). При этом качество отображения линии, ее «гладкость», качество стыковки звеньев линии не уменьшается.

5. ОБРАБОТКА РАСТРОВЫХ ИЗОБРАЖЕНИЙ

Элементами растровых изображений являются не графические примитивы, а отдельные пикселы. Это обуславливает специфические методы работы с такими изображениями. Существует несколько основных операций, выполняемых над растрами.

5.1 Масштабирование

Масштабирование растровых изображений, в отличие от векторных, связано с *изменением разрешающей способности растра*. Такое масштабирование выполняется в двух ситуациях.

Во-первых, такое масштабирование необходимо выполнять, если изображение должно быть отображено с помощью некоторого растрового графического устройства, разрешающая способность которого отличается от разрешающей способности самого изображения. Например, мы помещаем растровое изображение на страницу текстового редактора MS Word. При отображении этой страницы на экране разрешение рисунка должно быть приведено к разрешающей способности экрана, а при печати страницы — к разрешению принтера. В общем случае коэффициенты масштабирования растра по осям x и y составят

$$k_{x.\text{р}} = k_{x.\text{и}}\, m\, \frac{R_\text{э}}{R_\text{и}} \qquad \text{и} \qquad k_{y.\text{р}} = k_{y.\text{и}}\, m\, \frac{R_\text{э}}{R_\text{и}}, \tag{5.1}$$

где $k_{x.\text{и}}$ и $k_{y.\text{и}}$ – коэффициенты масштабирования при размещении изображения на странице; m – масштаб отображения страницы на экране (или принтере); $R_\text{и}$ и $R_\text{э}$ – разрешающая способность растров изображения и экрана (принтера). Заметим, что само исходное изображение не изменяется, оно только отображается на экране или бумаге.

Во-вторых, разрешающая способность растра самого изображения может быть изменена при помощи специальных программ обработки растровых изображений. Например, разрешение растра можно уменьшить, чтобы сократить объем памяти, занимаемой изображением. Другой пример — совмещение по размерам двух растровых изображений для создания фотоколлажа.

Случаи, когда необходимо уменьшить разрешение растра и когда его нужно увеличить, принципиально различны. В случае ***уменьшения разрешения*** растра имеет место *избыточность информации*, поскольку исходное изображение содержит больше пикселов, чем изображение, получаемое в результате.

Существуют два метода уменьшения разрешения. Наиболее простой и быстрый — присвоение пикселу создаваемого изображения цвета *одного* пиксела исходного. Обычно берут пиксел, расположенный ближе всего к центру нового пиксела, рис. 5.1 *а*. Информация из других пикселов теряется, поэтому качество изображения при таком способе существенно ухудшается,

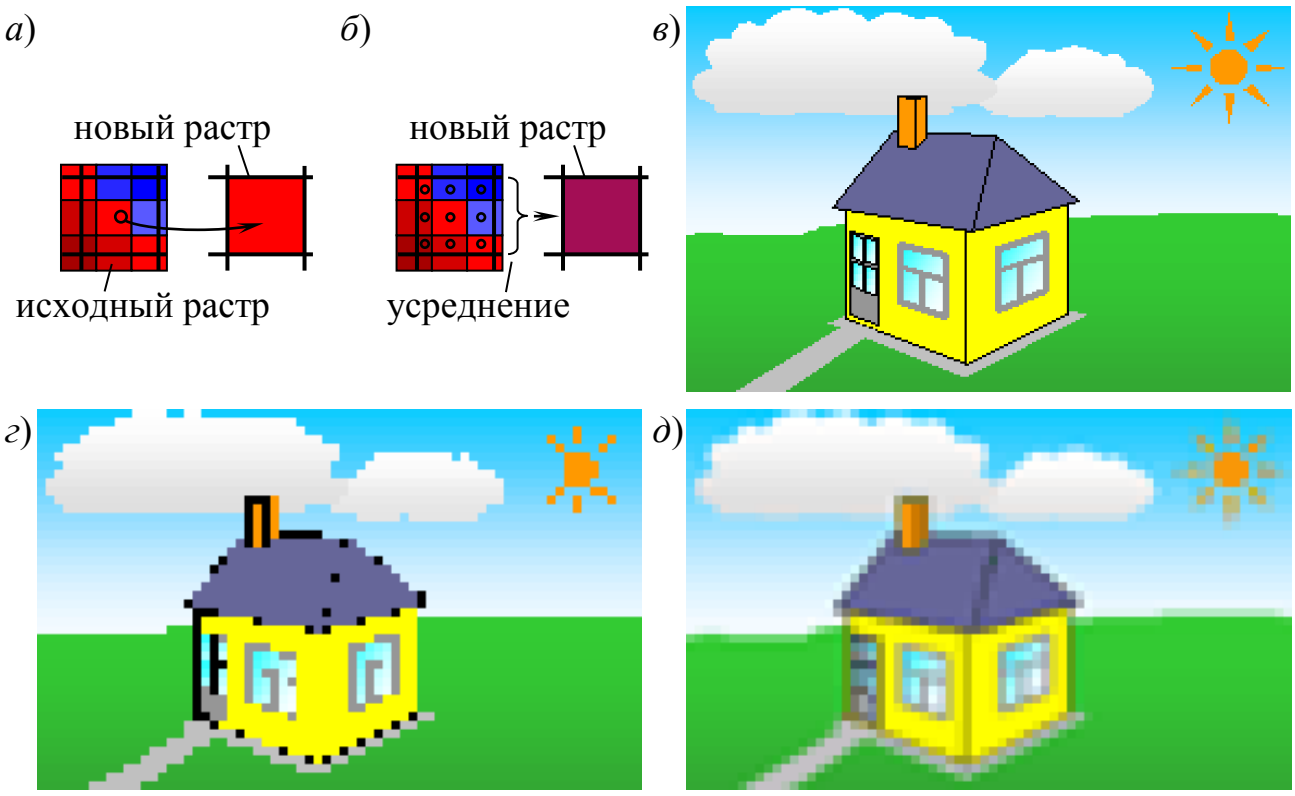


Рисунок 5.1 – Уменьшение разрешения растра без усреднения (*а*, *г*) и с усреднением (*б*, *д*); разрешающая способность исходного изображения (*в*) уменьшена в 4 раза (*г*, *д*)

рис. 5.1 *г*, — метод используется обычно только для быстрого просмотра изображений.

Более качественное изображение дает метод ***усреднения цветов*** всех пикселов исходного изображения, попадающих в область пиксела нового, рис. 5.1 *б*, *д*. При этом при дробных значениях коэффициентов масштабирования в эту область будут попадать как целые пикселы, так и части пикселов, поэтому усреднение необходимо выполнять с учетом площадей пикселов:

$$C_{\text{м}} = \frac{1}{S_{\text{м}}} \sum_i S_i \, C_i \,, \qquad (5.2)$$

где $C_{\text{м}}$ и $S_{\text{м}}$ – цвет и площадь пиксела масштабированного изображения; C_i и S_i – цвета пикселов исходного изображения и площади участков этих пикселов, попадающих в область пиксела масштабированного изображения.

Увеличение разрешения растра можно представить как задачу, обратную рассмотренной. При этом имеем недостаток информации — в получаемом изображении должно быть *больше* пикселов, чем в исходном. Простейший способ увеличения разрешения растра — *заполнение* цветом каждого пиксела той области, которая соответствует этому пикселу в увеличенном изображении. В результате получим изображение, состоящее из прямоугольников, рис. 5.2 *б*.

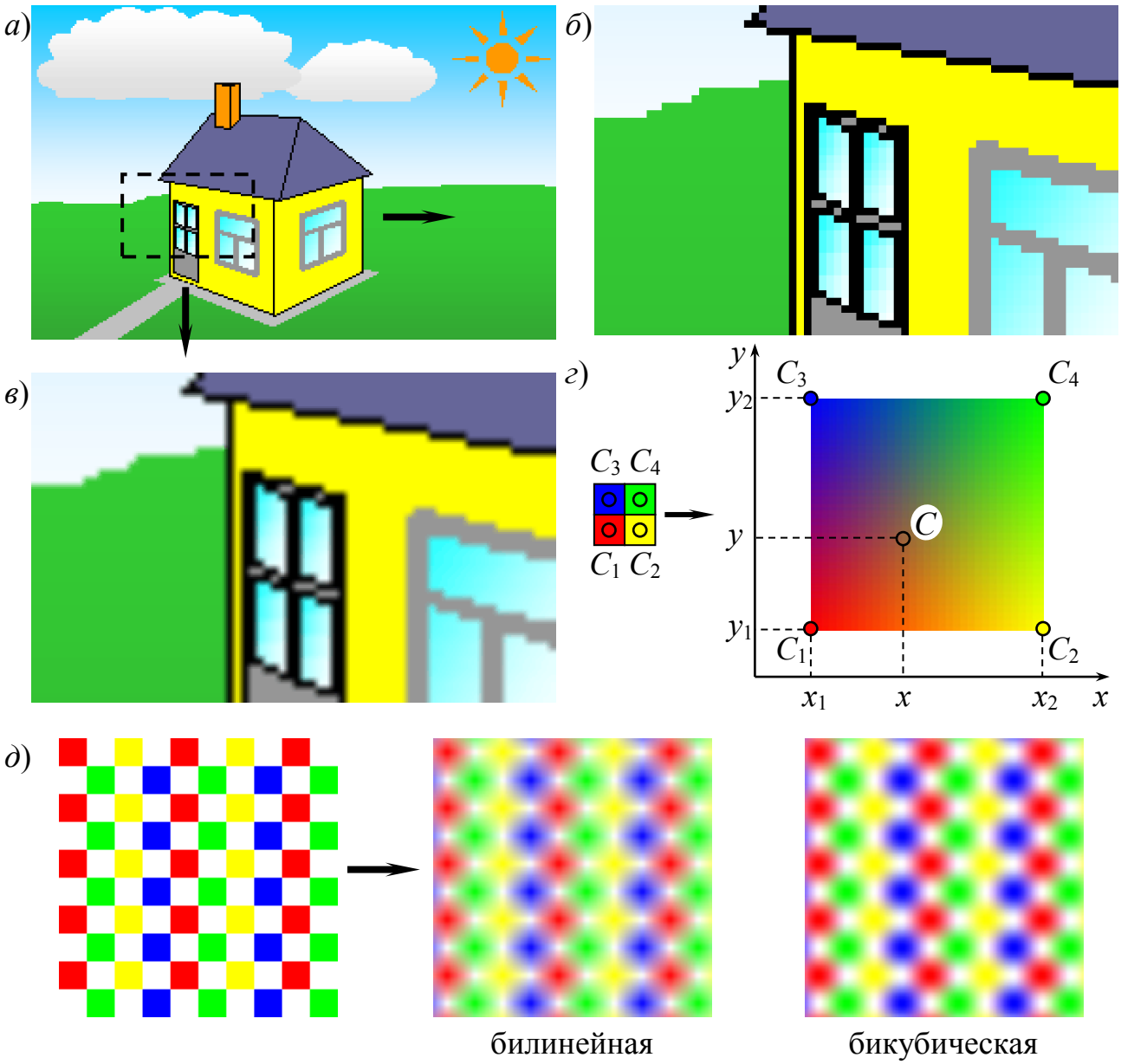


Рисунок 5.2 – Увеличение разрешения растра (*а*) в 4 раза без интерполяции (*б*) и с билинейной интерполяцией цвета пикселов (*в*); схема билинейной интерполяции (*г*); сравнение билинейной и бикубической интерполяций (*д*)

Более качественный результат дают методы ***интерполяции*** цвета пикселов, рис. 5.2 *в*. Простейший ее вариант — ***билинейная интерполяция***. Сущность ее заключается в том, что при масштабировании изображения определяются новые положения *центров пикселов*. При увеличении фрагмента изображения, рис. 5.2 *г*, центры пикселов C_1, C_2, C_3 и C_4 расположились в точках с координатами (x_1, y_1), (x_2, y_1), (x_1, y_2) и (x_2, y_2) соответственно. Прямоугольная область с вершинами в этих точках должна быть заполнена цветовым переходом, в котором цвет пиксела C с координатами центра (x, y) определяется путем усреднения цветов вершин по зависимости

$$C = \frac{1}{(x_2 - x_1)(y_2 - y_1)} \begin{pmatrix} C_1(x_2 - x)(y_2 - y) + C_2(x - x_1)(y_2 - y) + \\ + C_3(x_2 - x)(y - y_1) + C_4(x - x_1)(y - y_1) \end{pmatrix}, \qquad (5.3)$$

Знаменатель дроби — площадь прямоугольника. Вдоль ребер прямоугольника цвет пикселов определяется только двумя вершинами, что обеспечивает незаметную стыковку соседних областей интерполяции

Существуют другие, более сложные зависимости, использующиеся для интерполяции при увеличении масштаба растра, например ***бикубическая***. Изображение в таком случае получается более качественным, рис. 5.2 *д*, но увеличиваются объемы вычислений.

5.2 Цветовая коррекция

Цветовая коррекция растровых изображений — изменение цвета, яркости, цветовой насыщенности пикселов всего изображения или его части по некоторому закону.

Наиболее часто используются три вида цветовой коррекции: смешение каналов, коррекция уровней и HLS-коррекция.

Смешение каналов — наиболее простой вид цветокоррекции. Преобразование цвета для RGB-изображения, выполняется по зависимостям:

$$\begin{cases} R' = k_{r-r}R + k_{g-r}G + k_{b-r}B \\ G' = k_{r-g}R + k_{g-g}G + k_{b-g}B \\ B' = k_{r-b}R + k_{g-b}G + k_{b-b}B \end{cases}, \qquad (5.4)$$

где R, G и B – исходные интенсивности красного, зеленого и синего каналов; R', G' и B' – интенсивности после преобразования; $k_{r\text{-}r}$, $k_{g\text{-}r}$ … $k_{b\text{-}b}$ – коэффициенты, задающие взаимовлияние каналов. Коэффициенты могут быть как положительными, так и отрицательными — в последнем случае интенсивность соответствующего канала вычитается.

На рис. 5.3 приведен пример смешения каналов — к красному каналу добавлено 50% интенсивности синего канала.

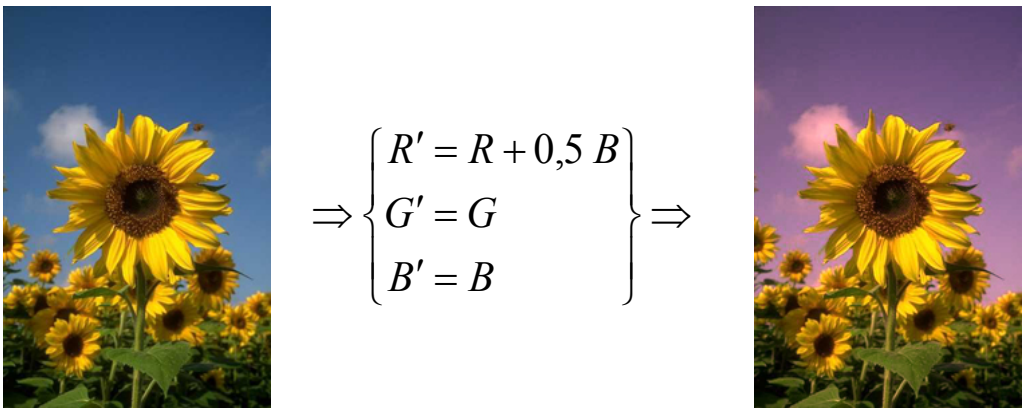


Рисунок 5.3 – Смешение каналов

Коррекция уровней заключается в преобразовании по определенному закону информации о яркости пиксела или интенсивности цветовых каналов. Для RGB-изображения запишем:

$$\begin{cases} R' = f_R(R), \\ G' = f_G(G), \\ B' = f_B(B), \end{cases} \tag{5.5}$$

где f_R, f_G и f_B – функции преобразования. В отличие от преобразования (5.4) коррекция уровней предполагает независимое преобразование каждого из цветовых каналов. В случае, если $f_R = f_G = f_B = f$ (все три канала преобразуются по единому закону) — выполняется ***коррекция яркостей*** пикселов:

$$I' = f(I), \tag{5.6}$$

где I и I' – яркость пиксела до и после преобразования.

Как известно, для RGB-изображения интенсивность каждого из каналов (красного, зеленого и синего) кодируется числом в диапазоне от 0 до $(2^n–1)$, где n – разрядность (количество бит информации на канал). В полноцветном изображении на каждый канал приходится 8 бит, т.е. имеем диапазон кодирования интенсивностей от 0 до 255. В дальнейшем будем задавать интенсивность в процентах — наибольшей интенсивности будет соответствовать 100 %.

Коррекцию уровней удобно представить на графике, рис. 5.4, где по шкале абсцисс отложены интенсивности каналов (яркости пикселов) исходного изображения, а по шкале ординат — преобразованного. Кривая *1* графика функции преобразования для каждого значения интенсивности исходного изображения задает значение интенсивности преобразованного. Диагональная линия 2 соответствует отсутствию преобразования (изображение не меняется). Чем больше отклонение кривой *1* от линии *2*, тем существеннее изменение интенсивностей.

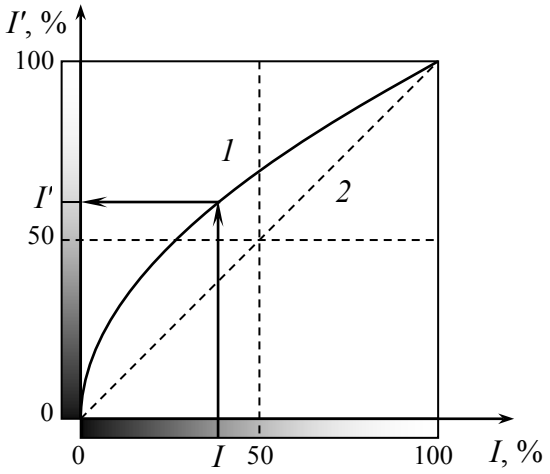


Рисунок 5.4 – Коррекция уровней

На рис. 5.5 приведены примеры коррекции уровней. Для обработки изображений особое значение имеют изменение яркости, контрастности, и выравнивание уровней.

Изменение яркости, рис. 5.5 *д*, *е*, выполняется как увеличение или уменьшение интенсивности всех пикселов на величину $\Delta_я$.

$$I' = I \pm \Delta_я . \qquad (5.7)$$

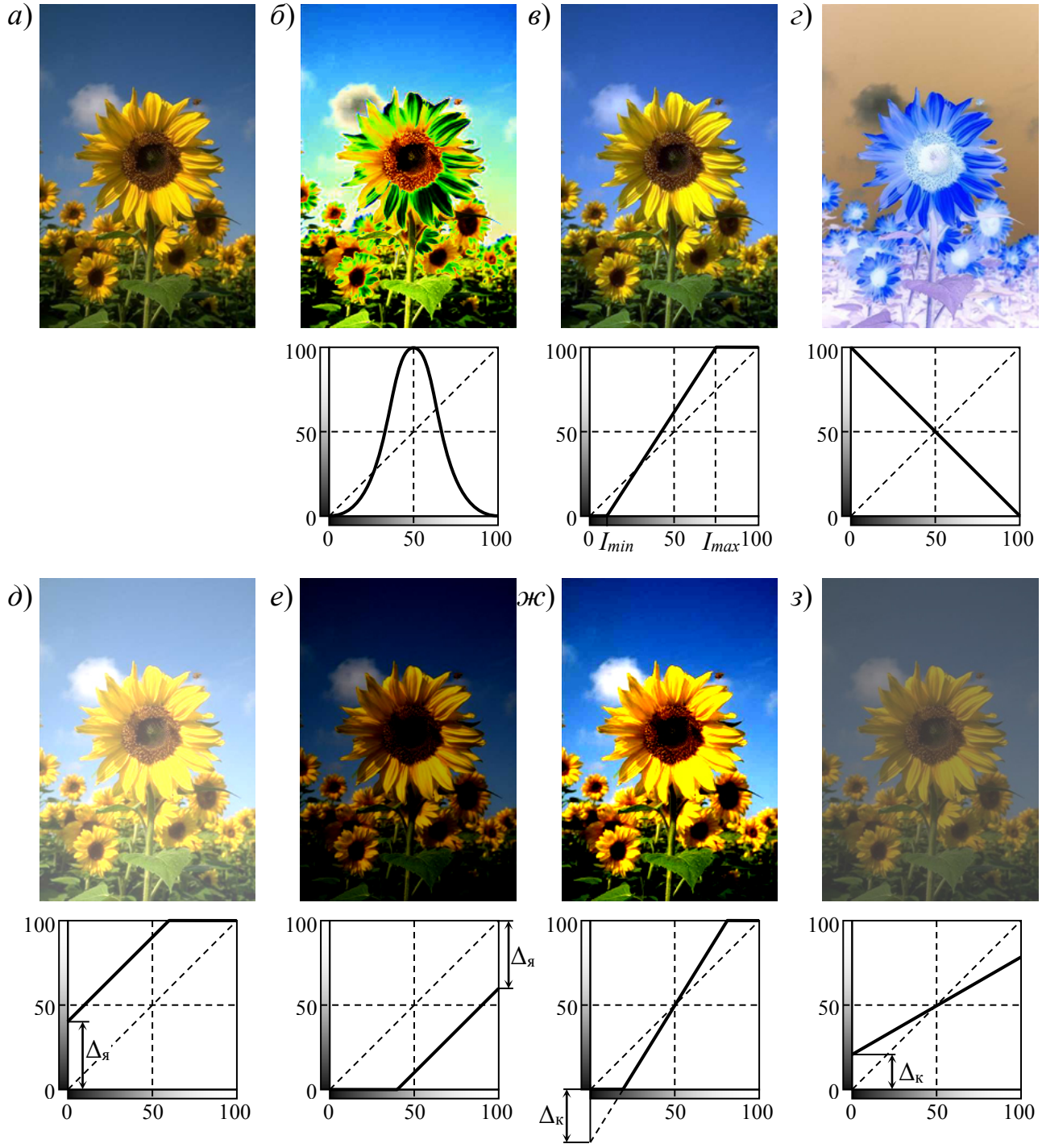


Рисунок 5.5 – Примеры коррекции уровней: исходное изображение (*а*); произвольная коррекция (*б*); выравнивание уровней (*в*); инверсия (*г*); увеличение яркости (*д*); уменьшение яркости (*е*); увеличение контрастности (*ж*); уменьшение контрастности (*з*)

Изменение контрастности изображения, рис. 5.5 *ж*, *з*, на величину $\Delta_к$ выполняется по зависимости

$$I' = I \pm \Delta_к \left(\frac{I}{I_{0,5}} - 1 \right), \tag{5.8}$$

где $I_{0,5}$ – интенсивность, равная 50 % максимальной.

Поскольку интенсивность цветовых каналов ограничена интервалом 0...(2^n–1), изменение яркости и увеличение контрастности приводят к *потере части информации* (горизонтальные участки соответствующих графиков). Выполнение обратной операции (например, уменьшение яркости после ее увеличения) не позволит восстановить изображение в исходном виде.

Операция ***выравнивания уровней***, рис. 5.5 *в*, предполагает анализ всех пикселов изображения для поиска наибольшей I_{max} и наименьшей I_{min} интенсивностей цветовых каналов (или канала яркости). Преобразование выполняется таким образом, чтобы интенсивности цветовых каналов заняли весь диапазон интенсивностей 0...(2^n–1):

$$I' = \frac{I - I_{\min}}{I_{\max} - I_{\min}} I_{1,0}, \tag{5.9}$$

где $I_{1,0}$ — максимально возможное значение интенсивности (2^n–1). Выравнивание уровней выполняется автоматически программами для обработки растровых изображений и позволяет улучшить вид изображения, полученного со сканера или с помощью цифрового фотоаппарата.

К рассмотренной группе операций относится также ***гамма-коррекция***. При выводе на монитор RGB-изображения дисплейный контроллер выполняет преобразование *числовых значений* интенсивностей цветовых каналов в *аналоговый сигнал*, в котором уровни напряжения задают яркость световых пятен на экране электронно-лучевой трубки (ЭЛТ). Однако, зависимость между напряжением U, поданным на ЭЛТ, и яркостью I свечения люминофора нелинейна:

$$I = c\, U^{\gamma}, \tag{5.10}$$

где γ – показатель степени, для цветных мониторов равный 2,3 … 2,8; c – константа. Для того, чтобы скомпенсировать нелинейность передачи яркости и получить на экране изображение с корректной передачей цветов (что необходимо, например, в издательском деле), выполняется *калибровка монитора*, чтобы определить значения γ и c, и затем, при выводе изображения, выполняется коррекция уровней по зависимости, обратной (5.10):

$$I' = \left(\frac{I}{c} \right)^{\frac{1}{\gamma}}, \tag{5.11}$$

Примерный вид графика гамма-коррекции показан на рис. 5.4.

HLS-коррекция заключается в преобразовании цветовой кодировки из модели RGB в модель HLS (цветовой тон, светлота и насыщенность), изменении этих параметров и обратном преобразовании из HLS в RGB.

В модели HLS цветовой тон задается как угол в интервале 0…360° по круговой шкале цветов. Изменение цветового тона можно представить как поворот цветового круга на заданный угол δ_H, рис. 5.6. На рис. 5.7 *б*, *в* и *г*. показаны примеры такого преобразования. Изменение светлоты и насыщенности выполняется добавлением или вычитанием некоторого значения, аналогично изменению яркости (5.7), рис. 5.7 *д*, *е*, *ж* и *з*.

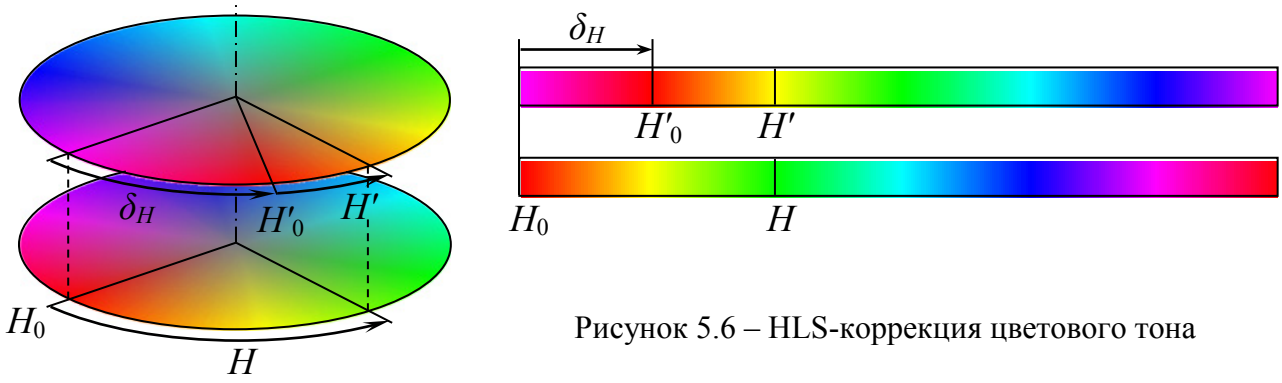


Рисунок 5.6 – HLS-коррекция цветового тона

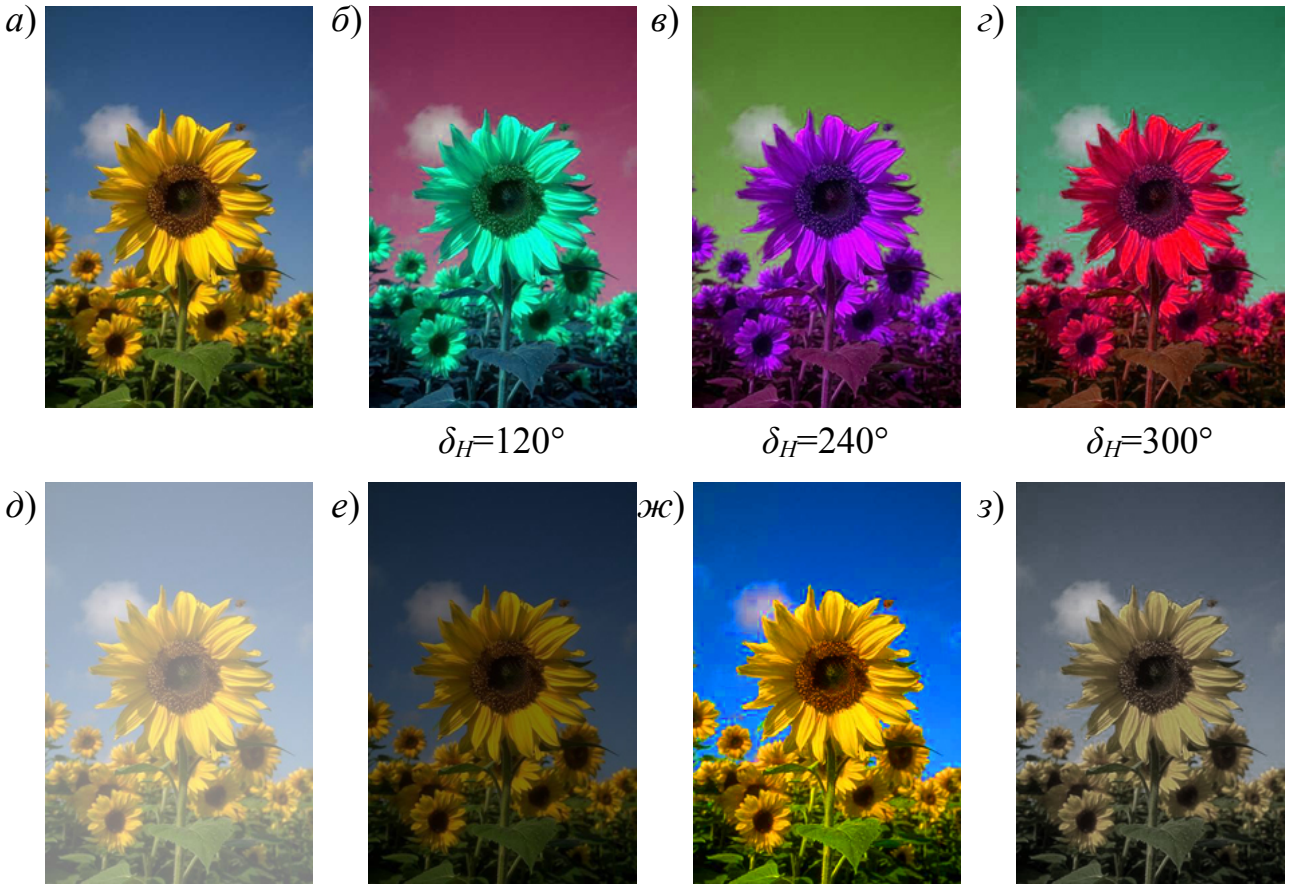


Рисунок 5.7 – Примеры HLS-коррекции: исходное изображение (*а*); коррекция цветового тона (*б*, *в* и *г*); увеличение светлоты (*д*); уменьшение светлоты (*е*); увеличение насыщенности (*ж*); уменьшение насыщенности (*з*)

5.3 Фильтрация

Фильтрацией растрового изображения называется преобразование, при котором цвет каждого пиксела получаемого изображения определяется как функция цветов нескольких пикселов исходного изображения (обычно расположенных в окрестности данного пиксела).

Математическая операция, выполняемая при фильтрации изображения, называется ***сверткой***, а функция, используемая для фильтрации — ***функцией свертки***. Область изображения, задействованная в вычислении цвета одного пикела, назовем ***областью свертки***, а сам этот пиксел — ***ядром свертки***, рис. 5.8.

В общем виде процесс фильтрации можно описать выражением:

$$C'_{i,j} = \sum_{(k,p)\subset\Delta} C_{i+k,j+p} \cdot F(k,p), \qquad (5.12)$$

где C и C' – цвета пикселов исходного и фильтрованного изображений; i, j – координаты пиксела, цвет которого вычисляется; $F(k, p)$ – функция свертки; Δ – область свертки; k, p – переменные функции свертки (координаты в системе координат области свертки).

В процессе фильтрации изображения ядро свертки перемещается, проходя через каждый пиксел изображения. Для каждого пиксела выполняется суммирование яркостей всех пикселов, попадающих в область свертки, умноженных на значение функции свертки для этих пикселов.

Простейший способ задания функции свертки — *матричный*. Значение функции свертки для каждого пиксела области свертки задается соответствующим значением в матрице. Для удобства и ускорения вычислений эти коэффициенты обычно задаются целочисленными, а выходной уровень интенсивности нормируется масштабным коэффициентом S:

$$F(k,p) = \frac{1}{S}\begin{Vmatrix} \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & f_{-1,1} & f_{0,-1} & f_{1,1} & \cdots \\ \cdots & f_{-1,0} & f_{0,0} & f_{1,0} & \cdots \\ \cdots & f_{-1,-1} & f_{0,-1} & f_{1,-1} & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \end{Vmatrix}, \qquad (5.13)$$

где $f_{k,p}$ – коэффициенты.

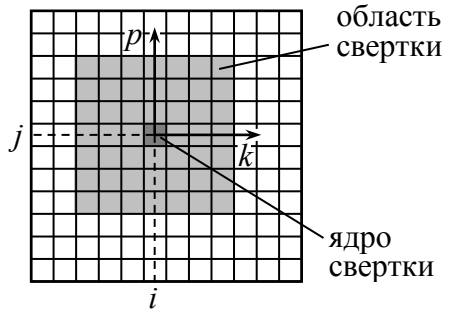


Рисунок 5.8 – Схема фильтрации изображения

Для того, чтобы яркость изображения после фильтрации оставалась неизменной, должно выполнятся *условие нормирования*

$$\sum_{(k,p)\subset\Delta} F(k,p) = 1, \tag{5.14}$$

т.е. сумма значений функции свертки для всей области свертки должна быть равна единице. Для этого масштабный коэффициент должен быть равен

$$S = \frac{1}{\sum\limits_{(k,p)\subset\Delta} F'(k,p)}, \tag{5.15}$$

где $F'(k,p)$ — ненормированная матрица коэффициентов свертки.

Наиболее часто используемыми фильтрами, для которых выполняется условие (5.14), являются фильтры ***размытия*** и ***повышения резкости***, примеры которых приведены на рис. 5.9 *б* и *в*. Фильтры размытия добавляют к цвету пиксела — ядра свертки цвета соседних пикселов. В результате изображение становится менее резким, «размытым». Фильтры повышения резкости, наоборот, вычитают цвета пикселов соседних с ядром свертки. Это приводит к повышению четкости границ областей, закрашенных разными цветами.

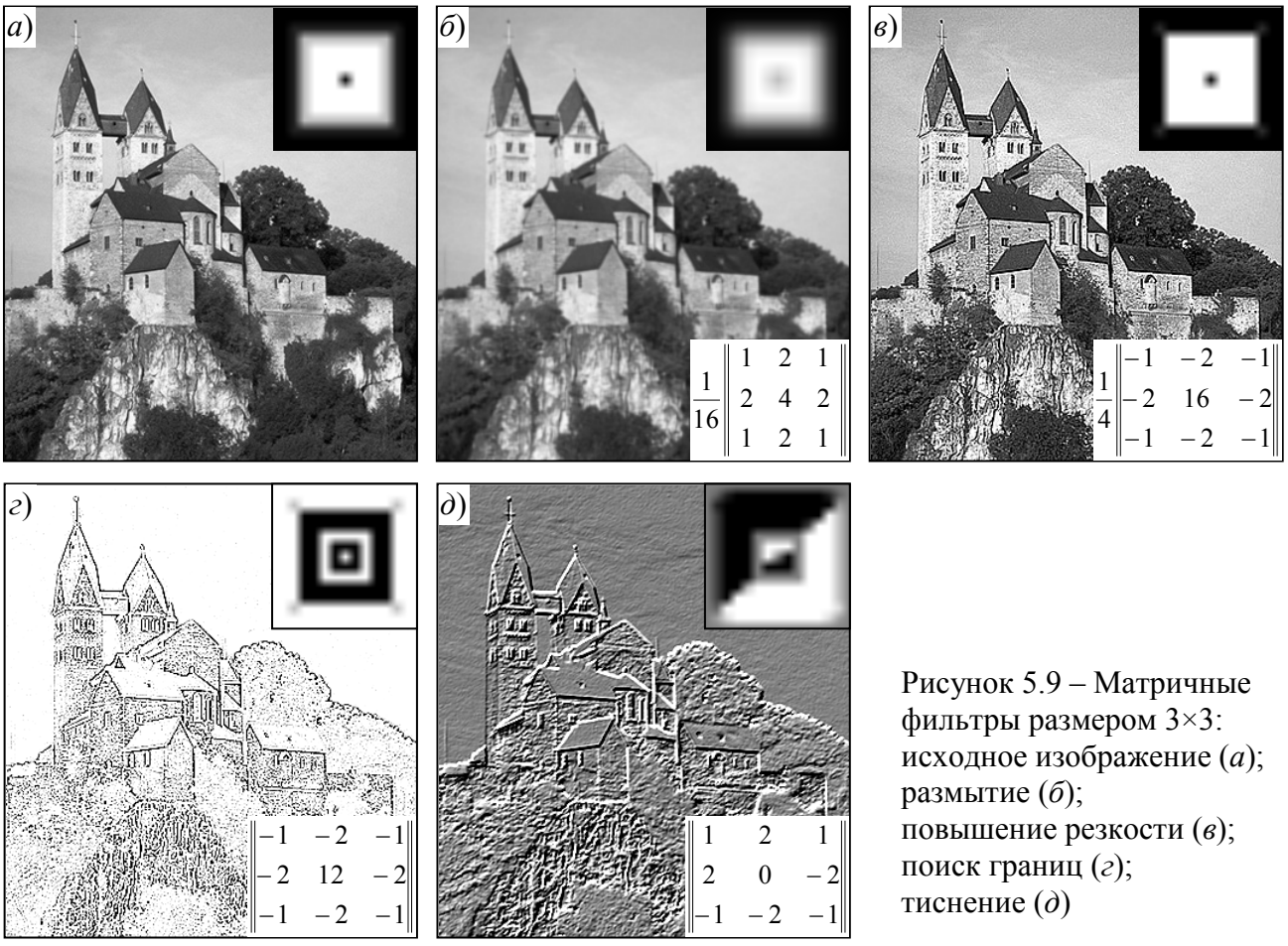


Рисунок 5.9 – Матричные фильтры размером 3×3: исходное изображение (*а*); размытие (*б*); повышение резкости (*в*); поиск границ (*г*); тиснение (*д*)

Если сумма значений функции свертки для всей области свертки равна нулю, такой фильтр превратит однородные области изображения (заполненные одним цветом) в черные. Отличаться от черного фона будут только те пикселы, которые находятся на границе областей, заполненных разными цветами, или в области цветового перехода. Такие фильтры используют для ***поиска границ*** объектов на изображении рис. 5.9 *г*, и для получения «рельефных» изображений — фильтр «***тиснение***», рис. 5.9 *д*.

Фильтр *поиска границ* дает изображение, которое выглядит как белый рисунок на черном фоне. Изображение, представленное на рис. 5.9 *г*, было инвертировано, чтобы получить эффект рисунка на белой бумаге.

При создании рельефного изображения, рис. 5.9 *д*, для превращения черных областей в серые в процессе фильтрации к интенсивности пиксела добавлялась величина, равная половине максимальной интенсивности.

Более универсальны фильтры, в которых функция свертки задана *аналитически*. Примером может служить фильтр ***размытия Гаусса*** с функцией свертки

$$F(k,p)=\frac{1}{2\pi\,\sigma^2}\cdot e^{-\frac{k^2+p^2}{2\sigma^2}}, \qquad (5.16)$$

где σ – параметр функции Гаусса, определяющий размер области свертки и степень размытия изображения, рис. 5.10. Область свертки представляет собой окружность радиусом 3σ и может иметь любой, в том числе весьма большой, размер.

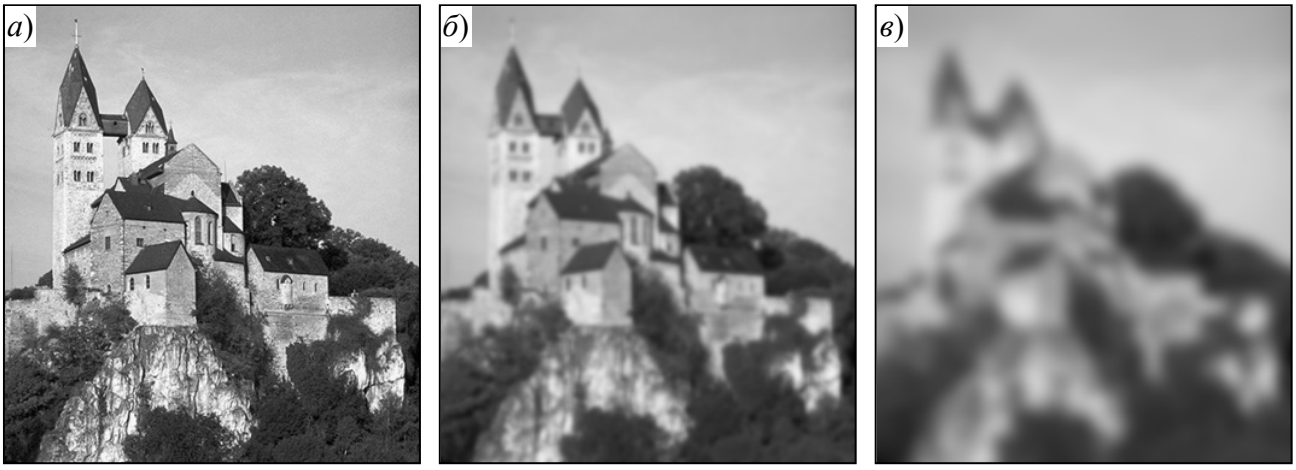


Рисунок 5.10 – Фильтр размытия Гаусса: исходное изображение (*а*), $\sigma = 2$ (*б*) и $\sigma = 6$ (*в*)

5.4 Палитризация и псевдотонирование

Для полноценной передачи цветных или полутоновых фотографических изображений необходимо иметь возможность кодировки интенсивности цветовых каналов или канала яркости с достаточно малым шагом квантования. В современных графических системах обычно используют 8-битовую кодировку, позволяющую задать $2^8 = 256$ уровней интенсивности (см. п. 3.3). Между тем, возможны ситуации, когда использовать такое количество цветов невозможно или нерационально по таким причинам:

1) Видеоадаптер не обладает достаточными аппаратными средствами для отображения полноцветного изображения (недостаточно памяти для хранения 24 битовых плоскостей, малая скорость обработки информации, низкая разрядность ЦАП). Сейчас эта проблема не столь актуальна, но недорогие компьютеры 10 … 15-летней давности в лучшем случае обеспечивали отображение 16 цветов, компьютеры 80-х годов — 4 цвета или вообще 2 — черный и белый. И сейчас подобная проблема возникает для маломощных цифровых устройств, например, контролеров различного оборудования.

2) При печати изображения. В отличие от дисплеев, устройства, печатающие на бумаге (струйные, лазерные и др. принтеры), располагают лишь несколькими цветами — по количеству цветов краски или тонера плюс белый цвет — чистая бумага. Обычно в цветной принтер заправляют краску 4 цветов, реже (в дорогих моделях) — 7 цветов. Нецветные принтеры заправлены одной черной краской и дают двухцветное изображение.

3) При необходимости уменьшить размер файла изображения, например при передаче его по сети. Для некоторых типов изображений (схемы, чертежи, рисунки) большое количество полутонов не нужно и даже вредно — при печати такие изображения могут получиться нечеткими, «грязными».

Обычно используются два типа изображений с низким цветовым разрешением: ***двухцветные*** (один бит на пиксел) и изображения с ***таблицей цветов*** (***палитровые***), см. п. 3.3.

Простейшим способом преобразования полутонового изображения в двухуровневое является ***пороговый метод***, при котором участки исходного изображения, интенсивность которых выше некоторого *порогового значения*, отображаются пикселями белого цвета, а участки, интенсивность которых ниже — черными пикселями, рис. 5.11. Пороговую величину обычно устанавливают приблизительно равной половине максимальной интенсивности.

Метод эффективен для чертежей, схематических рисунков и других изображений, где нет необходимости в полутонах — наоборот, нужно сделать такое изображение максимально четким и контрастным. Более сложные изображения (фотографии) становятся малореалистичными и плохо воспринимаются. Из-за больших ошибок выводимой интенсивности для каждого пикселя происходит потеря большого количества неконтрастных деталей.

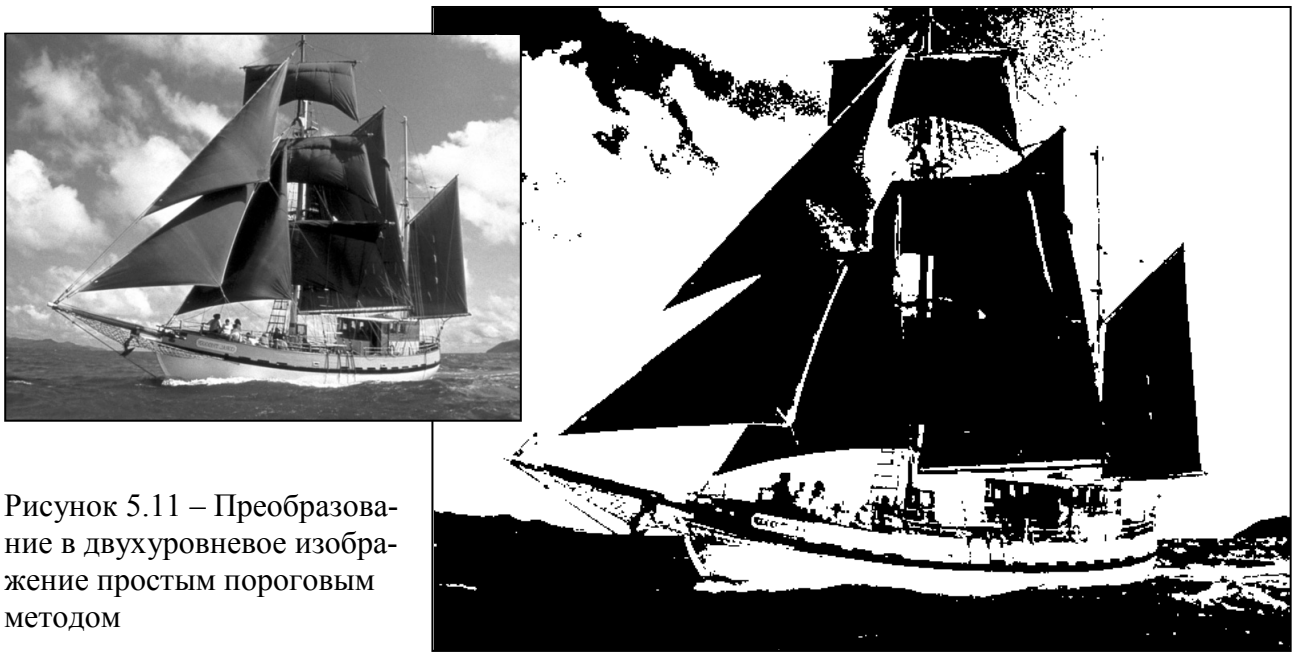


Рисунок 5.11 – Преобразование в двухуровневое изображение простым пороговым методом

Преобразование полноцветного изображения в цветное с малым количеством цветом может выполняться сходным образом — произвольный цвет пиксела заменяется ближайшим цветом из палитры. Такую процедуру называют ***палитризацией*** способом ***округления***. Степень различия исходного и нового цветов определяется суммой разностей интенсивностей для каждого из цветовых каналов, взятых по модулю:

$$\Delta_i = \left|R - R_{\Pi i}\right| + \left|G - G_{\Pi i}\right| + \left|B - B_{\Pi i}\right|, \qquad (5.17)$$

где R, G, B – интенсивности цветовых каналов некоторого пиксела исходного изображения; $R_{\Pi i}$, $G_{\Pi i}$, $B_{\Pi i}$ – интенсивности цветовых каналов i-го цвета из палитры. В ходе преобразования цвет пиксела заменяется индексом цвета палитры, для которого разница Δ будет наименьшей.

Качество получаемого таким способом изображения зависит от двух факторов — *количества цветов* палитры и *содержимого палитры*. Количество цветов палитры ограничено разрядностью кодирования информации и равно 2^N, где N – количество битовых плоскостей (число бит на один пиксел). На рис. 5.12 *а* показан результат округления цветной фотографии при 8-и цветах палитры, соответствующих основным RGB-цветам.

Использование стандартных палитр предоставляет весьма ограниченные возможности цветопередачи. Поэтому основная задача палитризации — подобрать цвета палитры так, чтобы небольшим количеством цветов наиболее качественно передать изображение. Такой подбор может быть выполнен как вручную, так и автоматически. Палитры, полученные *автоматически* путем поиска наиболее подходящего набора цветов, называют ***адаптивными***, рис. 5.12 *б*. Один из наиболее распространенных алгоритмов поиска называется *кластеризацией*:

1. Выбирается *начальное приближение* из 2^N цветов, расположенных некоторым образом внутри цветового куба.

2. Объем цветового куба разбивается на 2^N *кластеров*. Каждый кластер содержит цвета, для которых ошибка Δ_i (5.17) относительно центра кластера (*i*-го выбранного цвета) меньше, чем ошибка относительно всех других выбранных цветов, рис. 5.12 *в*. Проще — это область, ближайшая к выбранному цвету.

4. Выполняется *статистический анализ распределения цветов* в изображении и для каждого кластера находится *центроида* — усреднение всех цветов, попавших в кластер.

5. Центроиды кластеров назначаются новыми цветами палитры и процедура кластеризации повторяется начиная с шага 2. Поиск выполняется до тех пор, пока центроиды всех кластеров не совпадут с уже выбранными цветами палитры.

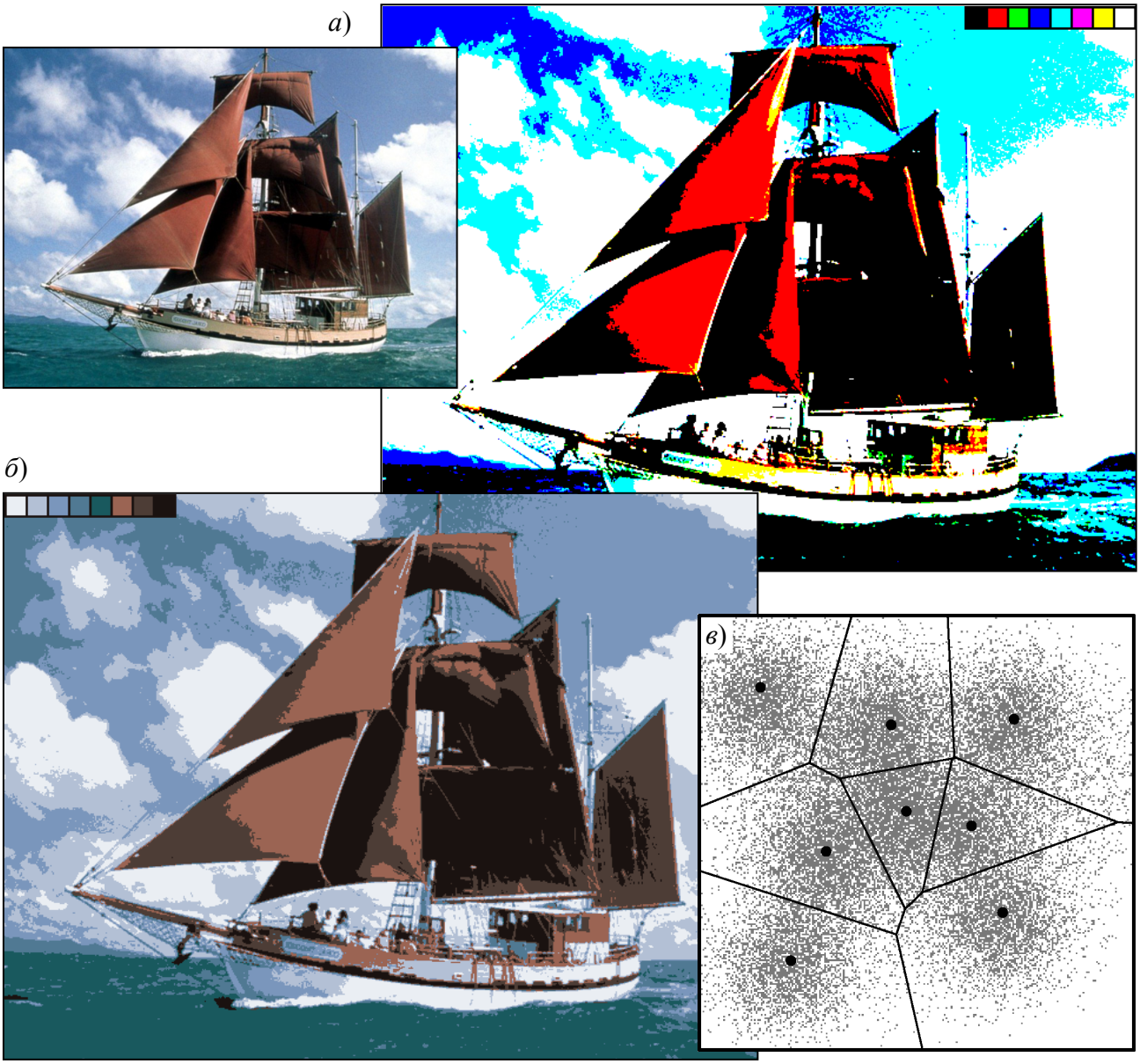


Рисунок 5.12 – Палитризация способом округления при 8-цветной стандартной (*а*) и адаптивной (*б*) палитрах, кластеризация пространства цветов (*в*)

Для улучшения качества двухцветных изображений и изображений с малым числом цветов используется *метод полутонов* или ***псевдотонирование***. Успех метода полутонов зависит от свойства зрительной системы человека объединять или сглаживать дискретную информацию.

Метод полутонов известен довольно давно. Первоначально он использовался при изготовлении гравюр, шелковых картин, текстильных изделий. В *гравюрах*, матрицы которых изготавливаются вручную, полутона передаются штрихами, расположенными более или менее часто, рис. 5.13 *а*.

В 1880 г. Стефаном Хагеном была изобретена современная полутоновая печать. Изображение с помощью специальной решетки разлагается на точки, причем размер каждой точки зависти от интенсивности отображаемого участка изображения. В этом методе можно получить большое количество фотографических полутонов серого, используя двухуровневую среду: черную краску на белой бумаге, рис. 5.13 *б*. При использовании 4-х красок (черной, желтой, голубой и пурпурной) получают цветные изображения, рис. 5.13 *в*. Качество передачи изображения зависит от шага решетки. Для газетных фотографий применяются решетки от 50 до 90 точек на дюйм. Бумага более высокого качества, позволяет использовать решетки с количеством точек от 100 до 300 на дюйм.

Размер точки цифрового изображения (пиксела экрана или принтера) обычно фиксирован. Поэтому в компьютерной графике используются методы, несколько отличные от полиграфических.

Для передачи полутонов используется ***метод конфигураций***. Участки изображения (обычно квадратной формы), содержащие несколько пикселов, объединяются в клетки — ***конфигурации***, — которые заполняются по определенной схеме в зависимости от уровня интенсивности участка изображения. Количество передаваемых уровней или полутонов определяется количеством пикселов в конфигурации — на единицу больше числа пикселов.

На рис. 5.14 *а* показана одна из возможных групп конфигураций для двухуровневого черно-белого дисплея. Для каждой клетки используется *четыре* пиксела (2×2). При такой организации получается *пять* возможных уровней или тонов серого. Число доступных уровней интенсивности можно увеличить с помощью увеличения размера клетки. Конфигурации для клетки 3×3 пикселов приведены на рис. 5.14 *б*. Они дают десять уровней интенсивности.

Рисунок конфигураций может быть различным, но должен удовлетворять таким требованиям:

1) Область, заполненная одинаковыми конфигурациями, должна казаться как можно более однородной.

2) Каждая последующая конфигурация должна получаться из прдыдущей путем добавления одного закрашенного пиксела (остальные пикселы должны оставаться такими же).

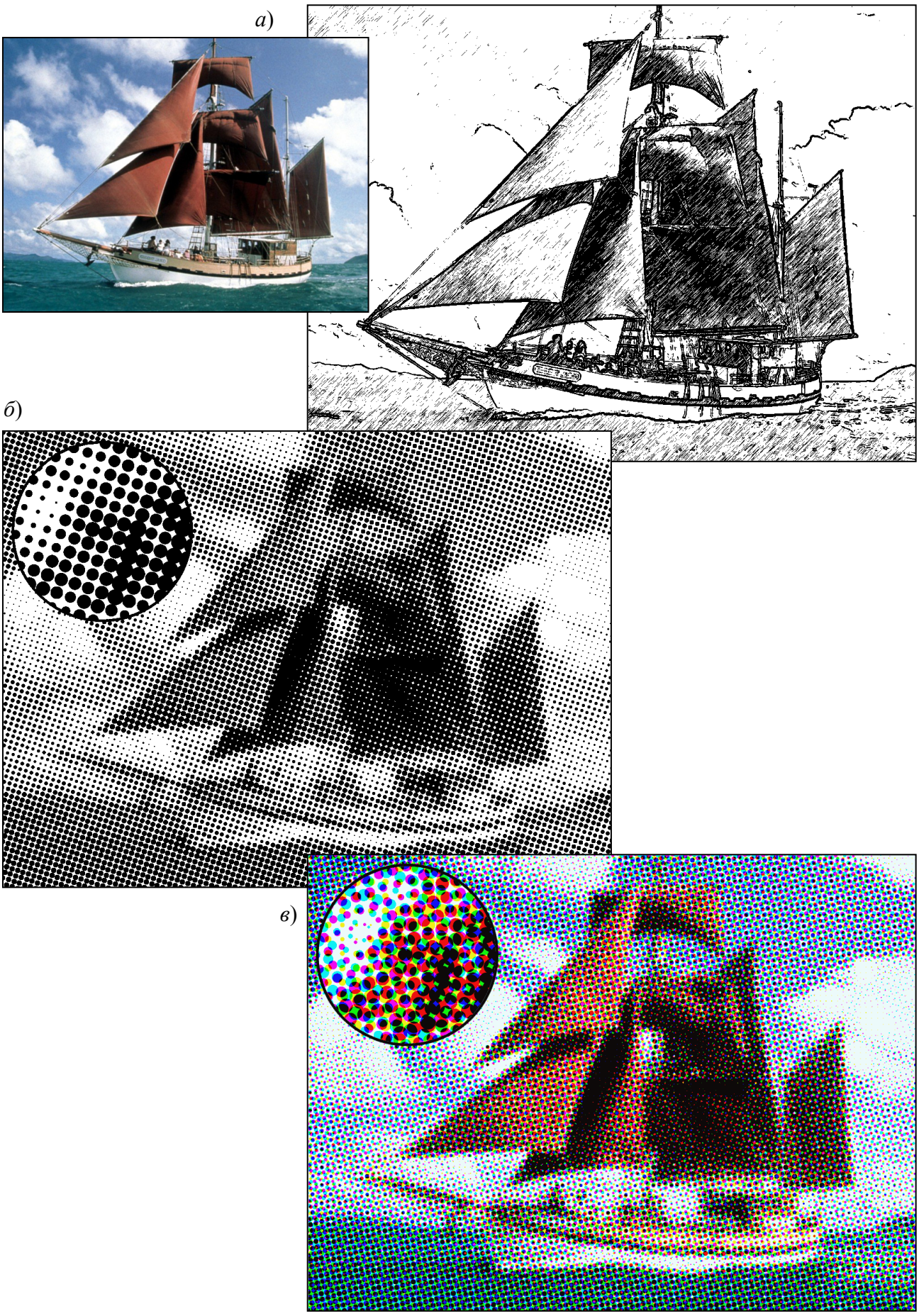


Рисунок 5.13 – Метод полутонов: гравюра (*а*), полутоновая (*б*) и цветная (*в*) печать

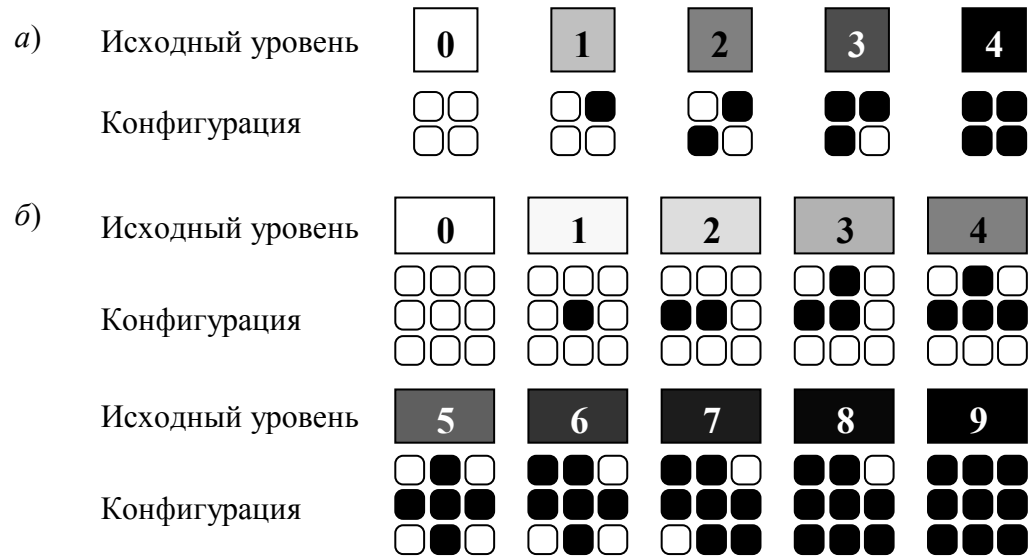


Рисунок 5.14 – Двухуровневые конфигурации 2×2 (*а*) и 3×3 (*б*)

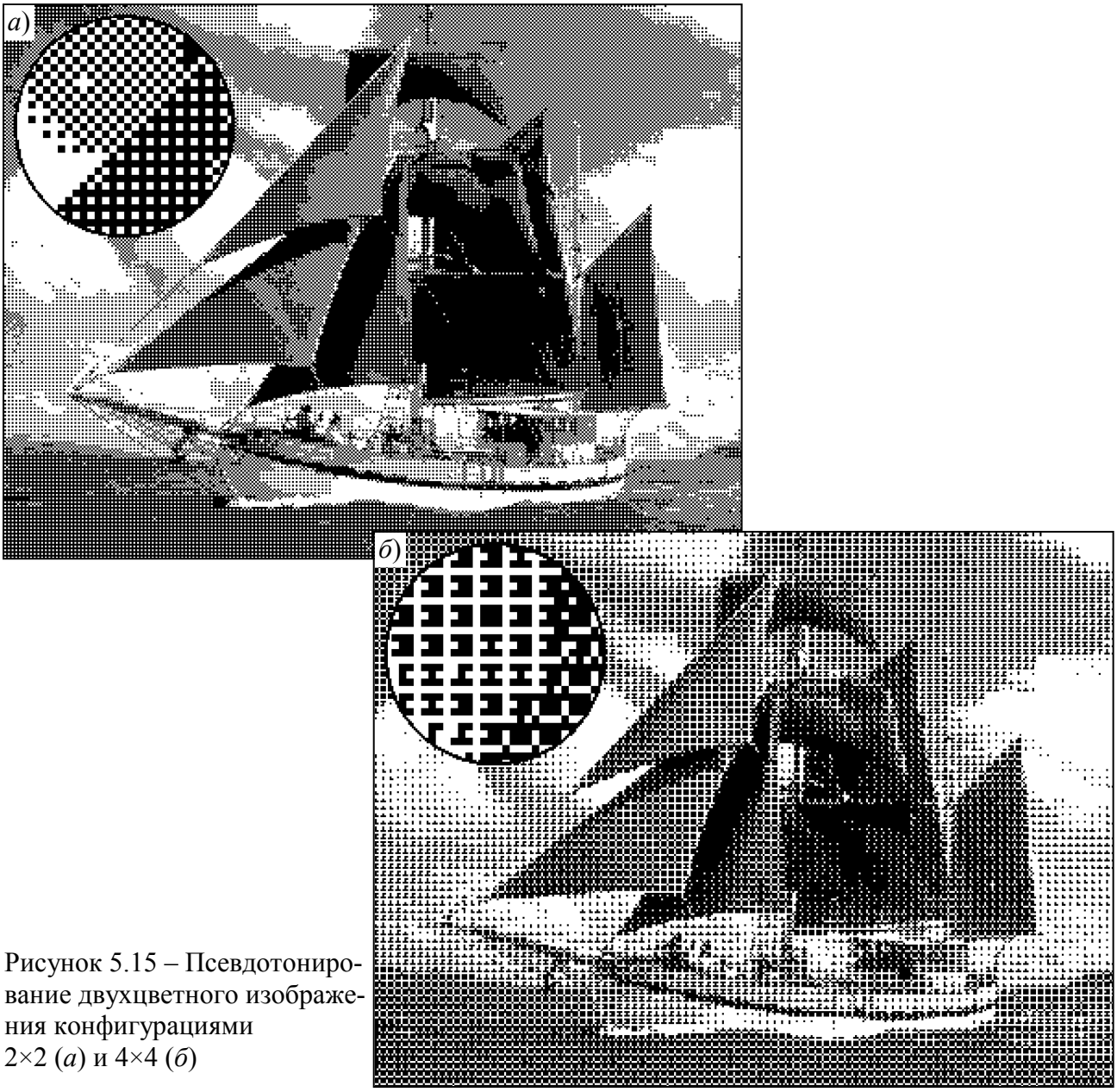


Рисунок 5.15 – Псевдотонирование двухцветного изображения конфигурациями 2×2 (*а*) и 4×4 (*б*)

Примеры изображений, аппроксимированных конфигурациями (2×2) и (4×4) приведены на рис. 5.15. Заметно, что улучшение *цветового разрешения* (количества полутонов) при увеличении размера клетки сопровождается ухудшением пространственного разрешения, что приводит к потере мелких деталей. Кроме того, повторяющиеся конфигурации приводят к появлению эффекта *фактуры* — регулярных узоров, ухудшающих восприятие изображения.

Метод конфигураций дает лучший эффект, если при преобразовании полутонового изображения в двухцветное можно увеличить разрешение растра. Наилучший результат дает вариант, при котором каждому пикселу исходного изображения будет соответствовать одна клетка (конфигурация) — потребуется увеличить разрешение во столько раз, каков размер конфигураций. Разумеется, это возможно только в том случае, если графическое устройство может отобразить картинку с таким разрешением.

Существует метод улучшения визуального разрешения для двухуровневых изображений без уменьшения пространственного разрешения — ***метод диффузии ошибки*** (метод *Флойда-Стейнберга*). Его можно рассматривать как модифицированный пороговый метод.

Изображение сканируется построчно; интенсивность каждого пиксела сравнивается с пороговой величиной (обычно половиной максимальной интенсивности), и выполняется замена цвета на черный или белый. При этом вычисляется *ошибка Δ* (разность исходной и конечной яркостей), которая может быть как положительной, так и отрицательной, рис. 5.16 *а*. Ошибка в заданной пропорции, рис. 5.16 *б*, распределяется по соседним пикселам до их сравнения с пороговой величиной.

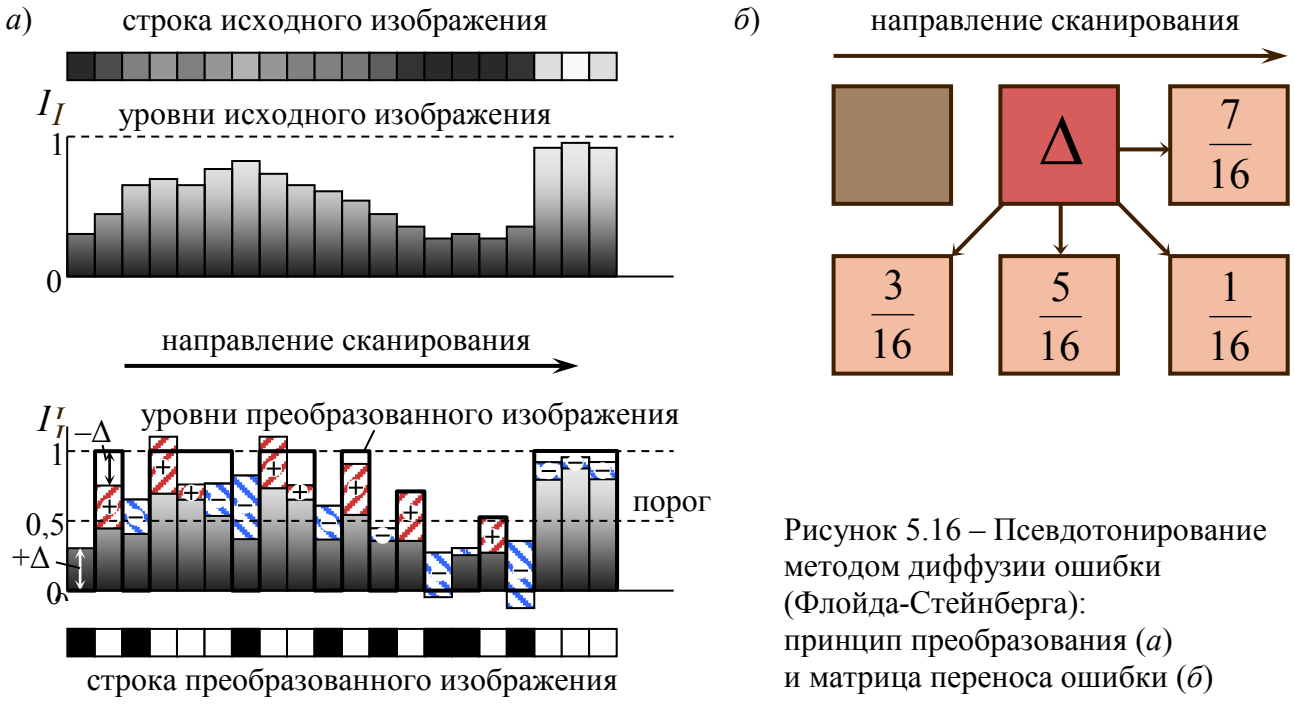


Рисунок 5.16 – Псевдотонирование методом диффузии ошибки (Флойда-Стейнберга): принцип преобразования (*а*) и матрица переноса ошибки (*б*)

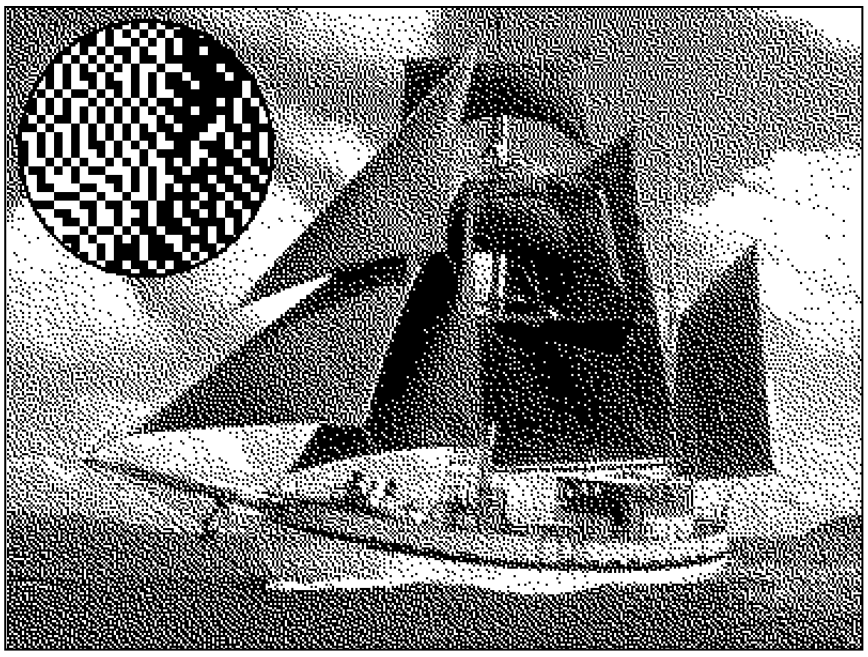


Рисунок 5.17 – Изображение, полученное методом диффузии ошибки

В результатае участок с некоторым уровнем интенсивности заполняется хаотическим набором светлых и темных точек, причем соотношение между количеством тех и других соответствует средней интенсивности участка, рис. 5.17. Распределение ошибки на соседние пикселы улучшает вид деталей изображения, поскольку информация, заключенная в изображении, не теряется.

Метод диффузии ошибки дает более качественное псевдо-полутоновое изображение, чем *метод конфигураций*, поскольку не снижает пространственного разрешения и исключает появление эффекта фактуры.

Методы конфигураций и диффузии ошибки применяются и для улучшения качества *цветных изображений* с малым числом цветов, в том числе палитровых, рис. 5.18.

Методы конфигураций и диффузии ошибки широко использовались на ЭВМ, имевших видеоадаптеры с малым цветовым разрешением. В настоящее время основная область применения этих методов — печать фотографических и других изображений на матричных, струйных и лазерных принтерах, поскольку для этих устройств псевдотонирование — единственный способ передачи полутонов.

Рисунок 5.18 – Псевдотонирование цветного изображения методами конфигураций (*а*) и диффузии ошибки (*б* и *в*)

5.5 Дефекты растровых изображений и их устранение

Дискретная структура растрового изображения приводит к некоторым ***искажениям*** машинно-сгенерированных изображений. Можно выделить два типа таких искажений: ***ступенчатость*** ребер (*лестничный эффект*) и некорректная визуализация тонких деталей или фактуры.

Основная причина появления *лестничного эффекта* заключается в том, что отрезки, ребра многоугольника, цветовые границы и т. д. имеют непрерывную природу, тогда как растровое устройство дискретно. Для представления отрезка, ребра многоугольника и т. д. на растровом устройстве необходимо начертить их в *дискретных координатах*, рис. 5.19 *а*.

В алгоритмах разложения отрезка в растр и заполнения многоугольника, интенсивность или цвет пикселя определяются интенсивностью или цветом единственной точки внутри области пикселя — его центра. Если центр находится внутри, то активируется весь пиксель. Если центр находился вне, то вся область пикселя игнорируется. В результате получается характерное ступенчатое, зазубренное ребро многоугольника или отрезок, рис. 5.19 *б*. Существует два метода устранения искажений изображения такого рода.

Первый метод связан с *увеличением разрешения растра*. Таким образом, учитываются более мелкие детали, а ступени становятся неразличимо малы. Однако существует определенное ограничение на способность растровых графических устройств с ЭЛТ выводить очень мелкие растры. В настоящее время предел составляет около 2000 пикселей в строке.

Второй метод устранения искажений изображения состоит в *размывании краев*. При этом пиксель трактуется не как точка, а как конечная область. Интенсивность или оттенок пикселя определяется способом *усреднения* оттенков всех элементов, попадающих в эту область.

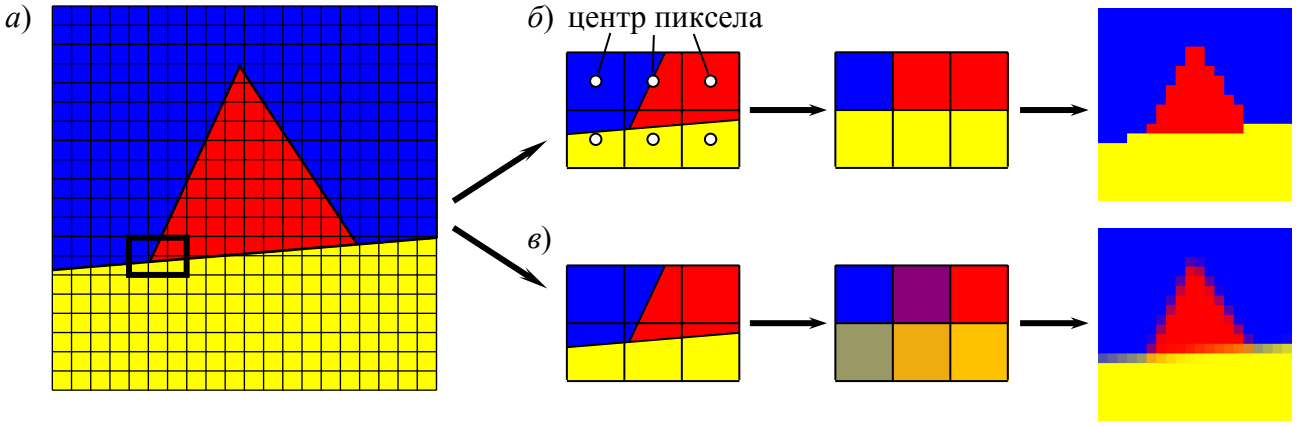


Рисунок 5.19 – Лестничный эффект: наложение непрерывных элементов на растр (*а*), растеризация по центрам пикселов (*б*) и усреднением цветов внутри пиксела (*в*)

а)

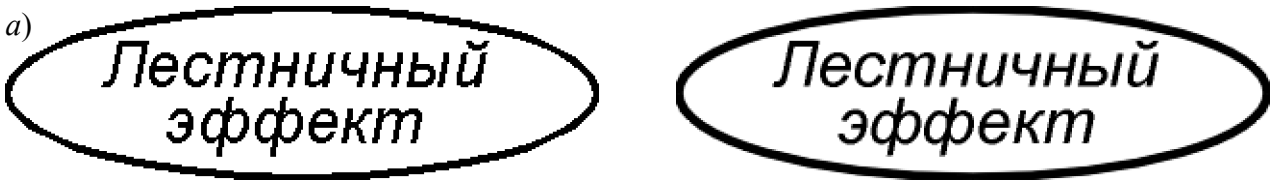
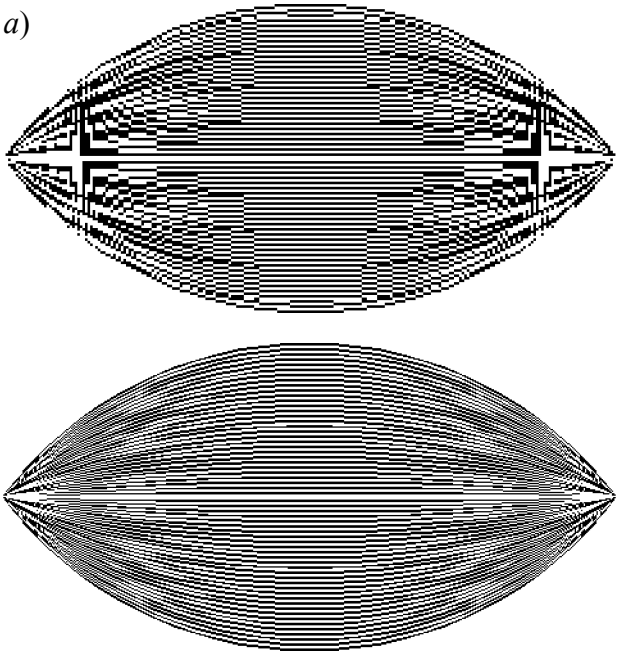


Рисунок 5.20 – Пример изображения до (*а*) и после (*б*) устранения лестничного эффекта

Стороны области пикселя образуют *отсекающее окно*. Для модулирования интенсивности пикселя используется отношение площади полученного в результате отсечения многоугольника к площади пикселя. Если внутри пикселя находится несколько многоугольников, то используется среднее взвешенное их атрибутов для модулирования атрибутов пикселя, рис. 5.19 *в*. В результате на границе закрашенной и незакрашенной зон создается плавный полутоновой переход, который визуально сглаживает ступенчатость — граница зрительно воспринимается гладкой. Пример изображения до и после устранения ступенчатости приведен на рис. 5.20 *а* и *б*.

Второй тип дефектов растра — *некорректная визуализация тонких деталей или фактуры* — наиболее заметно проявляется при отображении структур, состоящих из тонких линий, расположенных параллельно или под некоторым углом. Если толщина линий и расстояние между ними меньше или близки к размеру пикселя, линии сливаются, образуя некоторые закрашенные или незакрашенные зоны сложной формы, т.н. *муаровые узоры*, рис. 5.21 *а*. Эффективного метода борьбы с такими эффектами практически не существует, поскольку и метод усреднения, и увеличение разрешения растра могут не дать необходимого эффекта, рис. 5.21 *б*, *в*. Поэтому при создании растровых изображений следует избегать подобных регулярных структур.

а)



б)

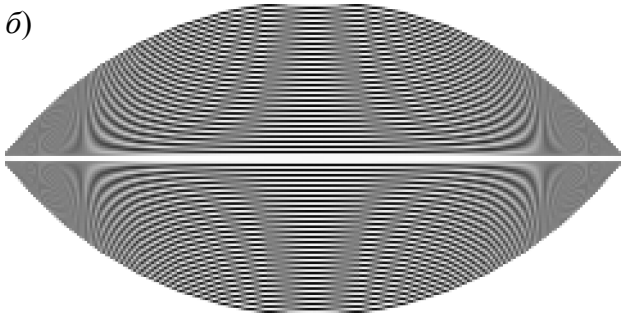


Рисунок 5.21 – Пример дефектов изображения регулярной структуры без усреднения (*а*), после усреднения (*б*) и при увеличенной в 2 раза разрешающей способности растра (*в*)

6. ОСНОВЫ ТРЕХМЕРНОЙ ГРАФИКИ

6.1 Преобразование координат в пространстве

В основе любого метода построения моделей в пространстве, как и для плоскости, лежит ***координатный метод***. Положение точки в пространстве определяется тремя координатами. В *прямоугольной* или ***Декартовой*** системе координат это *проекции вектора*, проведенного из начала координат в данную точку, на координатные оси — x, y и z.

Положение ***объекта*** в пространстве может быть задано координатами некоторой базовой точки, являющейся центром его собственной (*локальной*) системы координат, и *тремя углами* поворота этой системы координат относительно каждой из осей *глобальной системы координат*.

Преобразование координат в пространстве выполняется аналогично преобразованию на плоскости и включает *масштабирование*, *сдвиг* и *поворот*. В общем виде *аффинные преобразования в пространстве* могут быть представлены системой уравнений:

$$\begin{cases} X = A\,x + B\,y + C\,z + D, \\ Y = E\,x + F\,y + G\,z + H, \\ Z = K\,x + L\,y + M\,z + N; \end{cases} \tag{6.1}$$

или в матричном виде:

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} A & B & C & D \\ E & F & G & H \\ K & L & M & N \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}, \tag{6.2}$$

где x, y и z – координаты точки в локальной системе координат объекта; X, Y и Z – координаты этой же точки в глобальной системе координат; A, B … N – константы преобразования. Рассмотрим частные случаи аффинного преобразования.

1. ***Сдвиг*** центра локальной системы координат в точку (X_0, Y_0, Z_0), рис. 6.1 *а*:

$$\begin{cases} X = x + X_0, \\ Y = y + Y_0, \\ Z = z + Z_0, \end{cases} \qquad \text{или} \qquad \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & X_0 \\ 0 & 1 & 0 & Y_0 \\ 0 & 0 & 1 & Z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}. \tag{6.3}$$

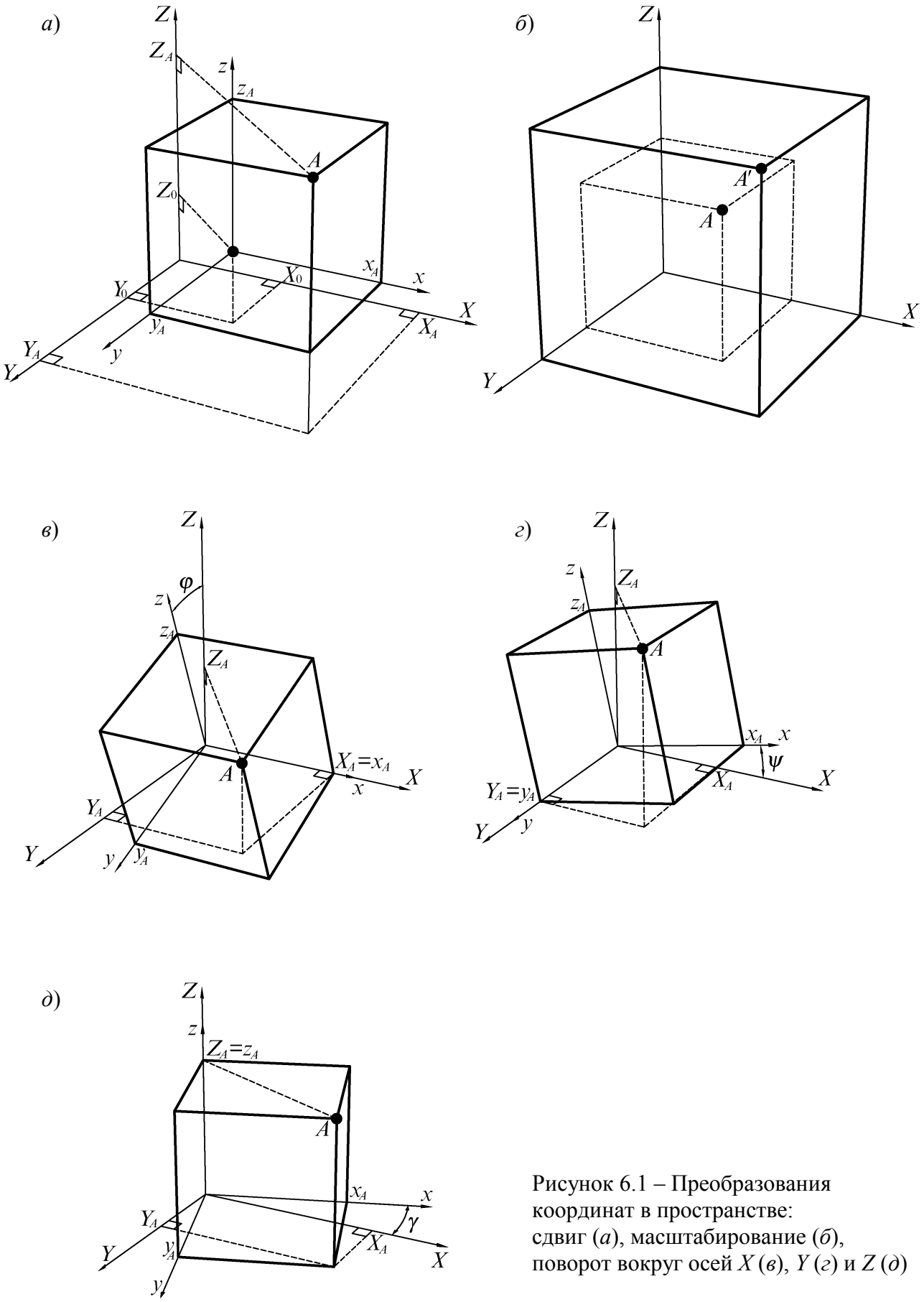


Рисунок 6.1 – Преобразования координат в пространстве: сдвиг (*а*), масштабирование (*б*), поворот вокруг осей *X* (*в*), *Y* (*г*) и *Z* (*д*)

2. ***Масштабирование*** (*сжатие/растяжение*) с центром в начале координат и коэффициентами масштабирования k_x, k_y и k_z, рис. 6.1 *б*:

$$\begin{cases} X = x\,k_x, \\ Y = y\,k_y, \\ Z = z\,k_z, \end{cases} \quad \text{или} \quad \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} k_x & 0 & 0 & 0 \\ 0 & k_y & 0 & 0 \\ 0 & 0 & k_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}. \tag{6.4}$$

3. ***Повороты***:

— вокруг оси X на угол φ, рис. 6.1 *в*:

$$\begin{cases} X = x, \\ Y = y\cos\varphi - z\sin\varphi, \\ Z = y\sin\varphi + z\cos\varphi, \end{cases} \quad \text{или} \quad \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\varphi & -\sin\varphi & 0 \\ 0 & \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}. \tag{6.5}$$

— вокруг оси Y на угол ψ, рис. 6.1 *б*:

$$\begin{cases} X = x\cos\psi - z\sin\psi, \\ Y = y, \\ Z = x\sin\psi + z\cos\psi, \end{cases} \quad \text{или} \quad \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\psi & 0 & -\sin\psi & 0 \\ 0 & 1 & 0 & 0 \\ \sin\psi & 0 & \cos\psi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}. \tag{6.6}$$

— вокруг оси Z на угол γ, рис. 6.1 *г*:

$$\begin{cases} X = x\cos\gamma - y\sin\gamma, \\ Y = x\sin\gamma + y\cos\gamma, \\ Z = z, \end{cases} \quad \text{или} \quad \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\varphi & -\sin\varphi & 0 \\ 0 & \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}. \tag{6.7}$$

Как можно видеть, при повороте вокруг некоторой оси глобальной системы координат координата точки по этой оси остается неизменной, остальные две координаты преобразуются аналогично повороту на плоскости (5.2) и (5.9).

Последовательно выполняя операции масштабирования, сдвига и поворотов вокруг осей X, Y и Z, можно расположить некоторый объект в произвольном положении в некоторой области пространства.

6.2 Проекции

Подавляющее большинство современных средств отображения компьютерной графики (дисплеи, принтеры) синтезируют изображение *на плоскости*. При создании изображений трехмерных объектов используют метод ***проекций***.

Проекция объекта — это изображение, полученное как результат пересечения пучком параллельных или сходящихся *проецирующих лучей*, идущих от этого объекта, поверхности проецирования (плоскости или, реже, цилиндрической, сферической поверхностей).

В компьютерной графике наиболее распространены ***параллельная*** и ***центральная*** проекции. *Параллельная проекция* создается пучком лучей, параллельных друг другу, рис. 6.2 *а*. *Центральная проекция*, называемая также ***перспективной***, получается, когда проецирующие лучи исходят из одной точки пространства, рис. 6.2 *б*. *Перспективная проекция* позволяет создать изображение, которое ближе к той картинке, которую воспринимает наш глаз, а потому выглядит реалистичнее.

При отображении пространственных объектов на экране или листе бумаги с помощью принтера, необходимо знать координаты проекций точек объекта на экран или лист. Обычно используют две системы координат — *мировую* и *экранную*.

Мировые координаты описывают истинное положение объектов в пространстве. В большинстве случаев это *трехмерная* прямоугольная система координат.

Экранные координаты — это система координат (обычно *двумерная*) устройства отображения, которое осуществляет вывод изображений объектов в заданной проекции.

Для получения изображения трехмерного объекта на устройстве отображения необходимо задать *преобразование координат* из *мировых* в *экранные*.

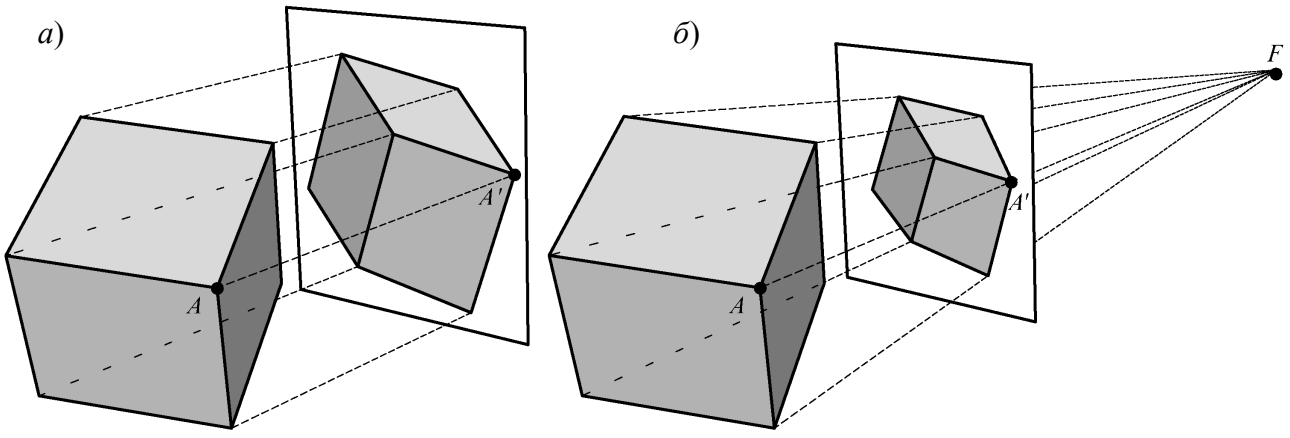


Рисунок 6.2 – Параллельная (*а*) и центральная (*б*) проекции

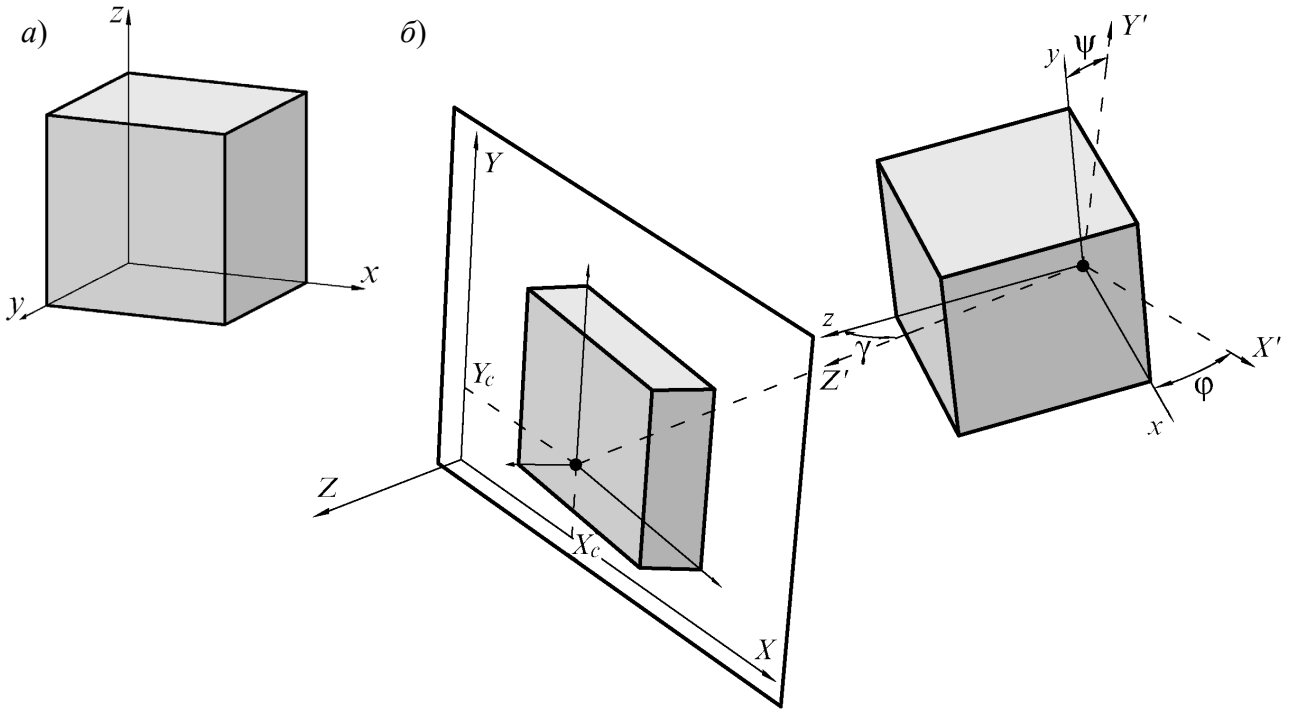


Рисунок 6.3 – Аксонометрическая проекция куба (*а*) и преобразование координат (*б*)

Для создания упрощенных изображений объектов (например, в процессе трехмерного моделирования) используют ***аксонометрическую проекцию*** — разновидность параллельной проекции, когда все проецирующие лучи направлены под прямым углом к плоскости проецирования.

Аксонометрическая проекция, рис. 6.3 *а*, имеет такие особенности:

— проекции отрезков параллельных прямых *всегда параллельны*;

— *длины проекций* одинаковых отрезков параллельных прямых *всегда одинаковы*, независимо от их удаленности от плоскости проецирования.

Несмотря на то, что аксонометрическая проекция «точно» воспроизводит объект, человек воспринимает такое изображение как «плоское», лишенное объема.

Рассмотрим *преобразование координат* для получения аксонометрической проекции. Пусть необходимо получить аксонометрическую проекцию некоторого объекта (например, куба), заданного в мировой системе координат (x, y, z), на расположенную произвольно плоскость экрана, рис. 6.3 *б*. Имеем экранную систему координат (X, Y, Z), ось Z направлена перпендикулярно плоскости экрана. Относительно *экранной* системы координат *мировая* система смещена на величины X_C, Y_C и Z_C вдоль соответствующих осей и повернута на углы φ, ψ и γ вокруг осей X, Y и Z соответственно.

Введем вспомогательную систему координат (X', Y', Z'), оси которой параллельны соответствующим осям *экранной системы*, а центр совпадает с центром *мировой системы*. Такую систему координат называют ***видовой***. Она определяет *ракурс* показа объекта.

Для преобразования координат объекта из мировой в видовую систему, необходимо выполнить умножение матрицы координат на матрицы поворота вокруг каждой из осей (6.5), (6.6) и (6.7). Условно это можно записать так:

$$\begin{bmatrix} X' \\ Y' \\ Z' \\ 1 \end{bmatrix} = \begin{bmatrix} \text{Поворот} \\ \text{вокруг} \\ \text{оси } Z' \\ \text{на } \gamma \end{bmatrix} \begin{bmatrix} \text{Поворот} \\ \text{вокруг} \\ \text{оси } Y' \\ \text{на } \psi \end{bmatrix} \begin{bmatrix} \text{Поворот} \\ \text{вокруг} \\ \text{оси } X' \\ \text{на } \varphi \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}. \tag{6.8}$$

Чтобы получить экранные координаты, нужно выполнить смещение объекта на величины X_C и Y_C вдоль соответствующих осей. Смещение вдоль координаты Z выполнять необязательно, т.к. для построения проекции достаточно координат X и Y:

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} \text{Сдвиг} \\ \text{на} \\ X_C \text{ и } Y_C \\ \end{bmatrix} \begin{bmatrix} X' \\ Y' \\ Z' \\ 1 \end{bmatrix}. \tag{6.9}$$

Полная формула преобразования координат объекта выглядит следующим образом:

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} \text{Сдвиг} \\ \text{на} \\ X_C \text{ и } Y_C \\ \end{bmatrix} \begin{bmatrix} \text{Поворот} \\ \text{вокруг} \\ \text{оси } Z' \\ \text{на } \gamma \end{bmatrix} \begin{bmatrix} \text{Поворот} \\ \text{вокруг} \\ \text{оси } Y' \\ \text{на } \psi \end{bmatrix} \begin{bmatrix} \text{Поворот} \\ \text{вокруг} \\ \text{оси } X' \\ \text{на } \varphi \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}. \tag{6.10}$$

Для получения проекции объекта на плоскость экрана теперь достаточно приравнять координату Z каждой его точки к нулю. Тогда оставшиеся две координаты X и Y зададут положение проекции этой точки на плоскости экрана.

Произведение всех матриц сдвига и поворотов, необходимых для преобразования координат объекта из мировых в экранные, обычно называют ***матрицей видового аффинного преобразования***. Тогда (6.10) можем записать:

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} \text{Матрица} \\ \text{видового} \\ \text{аффинного} \\ \text{преобразования} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}. \tag{6.11}$$

Поскольку отображаемый объект может быть намного больше или, наоборот, намного меньше области экрана, в матрицу видового преобразования обычно вводят *матрицу масштабирования* (6.4).

Изображения, получаемые с помощью фото- и кинокамер, так же, как и изображение, воспринимаемое глазом человека, отличаются от аксонометрических, поскольку относятся к *центральным* или ***перспективным проекциям***. Поэтому для получения реалистичного изображения трехмерных объектов необходимо генерировать их ***перспективную проекцию*** на плоскость экрана. Такую проекцию можно представить себе как изображение на стекле, через которое смотрит наблюдатель, расположенный в точке *F*, рис. 6.4. Рассмотрим методику получения такой проекции.

Пусть некоторый объект задан в *мировых* координатах (x, y, z). Расположим *видовую систему координат* (X, Y, Z) так, чтобы точка *F* лежала на оси *Z* и имела координаты (0, 0, Z_F). Плоскость экрана пересекает ось *Z* в точке *3* с координатой $Z_{пл}$.

Преобразуем координаты объекта в *видовые*, умножив на матрицу видового аффинного преобразования (6.11). Произвольная точка *A* объекта, имеющая видовые координаты (X_A, Y_A, Z_A), проецируется лучом *AF* в точку *A′* на плоскости экрана, имеющую экранные координаты ($X_{пA}$, $Y_{пA}$). Исходя из подобия треугольников *1F3* и *1′F3′* а также *2F3* и *2′F3′* запишем пропорции:

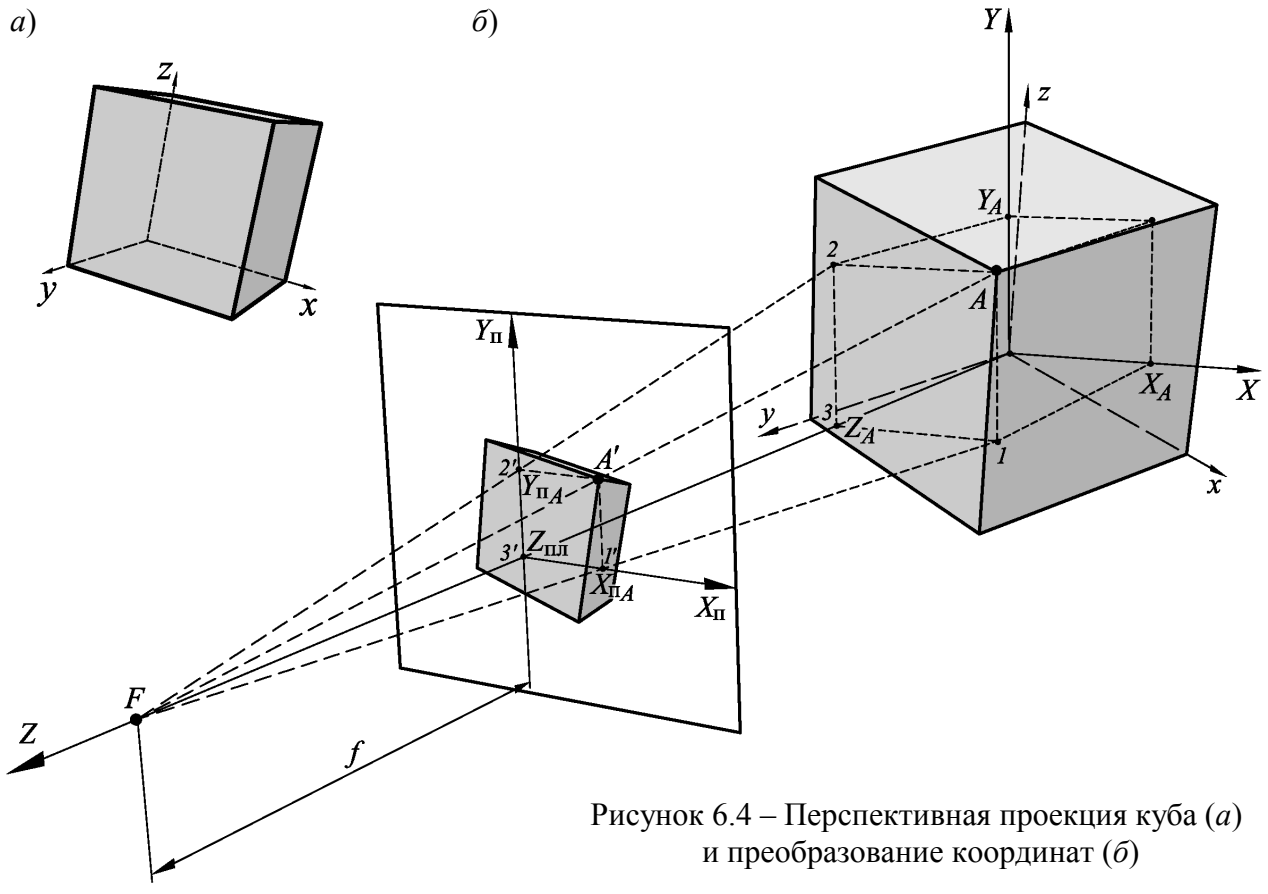


Рисунок 6.4 – Перспективная проекция куба (*а*) и преобразование координат (*б*)

$$\begin{cases} \dfrac{X_{\text{п}A}}{Z_F - Z_{\text{пл}}} = \dfrac{X_A}{Z_F - Z_A}, \\ \dfrac{Y_{\text{п}A}}{Z_F - Z_{\text{пл}}} = \dfrac{Y_A}{Z_F - Z_A}. \end{cases} \tag{6.12}$$

Найдем координаты проекции:

$$\begin{cases} X_{\text{п}A} = X_A \dfrac{Z_F - Z_{\text{пл}}}{Z_F - Z_A}, \\ Y_{\text{п}A} = Y_A \dfrac{Z_F - Z_{\text{пл}}}{Z_F - Z_A}. \end{cases} \tag{6.13}$$

Расстояние $f = Z_F - Z_{\text{пл}}$ от точки схода лучей (***фокуса***) до плоскости экрана называют ***фокусным расстоянием***. Тогда можем записать

$$\begin{cases} X_{\text{п}A} = \dfrac{f\, X_A}{Z_F - Z_A}, \\ Y_{\text{п}A} = \dfrac{f\, Y_A}{Z_F - Z_A}, \end{cases} \quad \text{или} \quad \begin{cases} X_{\text{п}A} = \boldsymbol{\Pi}_x\left(f, X_A, Z_A\right), \\ Y_{\text{п}A} = \boldsymbol{\Pi}_y\left(f, Y_A, Z_A\right), \end{cases} \tag{6.14}$$

где $\boldsymbol{\Pi}$ – функция перспективного преобразования координат. В отличие от видового преобразования, перспективное преобразование — *нелинейное*. Значения коэффициентов преобразования зависят от координаты Z, находящейся в знаменателе.

Перспективная проекция имеет следующие особенности, см. рис. 6.4 *а*:

— *прямые линии* изображаются прямыми линиями;

— *параллельные прямые*, которые параллельны плоскости проецирования, изображаются параллельными прямыми;

— *параллельные прямые*, непараллельные плоскости проецирования, изображаются прямыми, сходящимися в одной точке;

— *длины проекций* одинаковых отрезков параллельных прямых будут различными в зависимости от их удаленности от плоскости проецирования.

Вид перспективной проекции определяется взаимным расположением пространственного объекта, плоскости проецирования и точки фокуса. Для того, чтобы перспективное изображение объекта воспринималось наиболее естественно, необходимо, чтобы угол α, под которым человек рассматривает изображение (на экране, листе бумаги и др.) соответствовал тому углу α', под которым мы видели бы изображаемый трехмерный объект, существуй он в действительности, рис. 6.5.

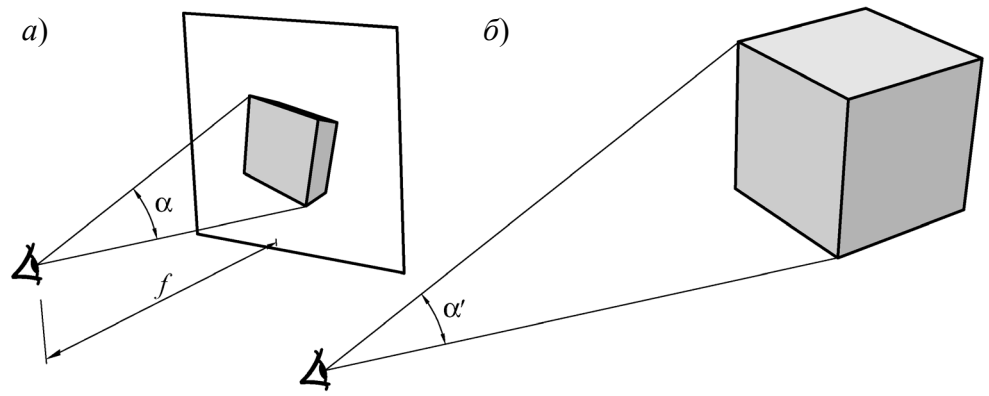


Рисунок 6.5 – Угол обзора изображения (*а*) и объекта (*б*)

Если фокусное расстояние слишком мало и угол α' обзора объекта из точки фокуса значительно больше угла α обзора изображения наблюдателем, получим искаженное изображение с утрированной, чрезмерной перспективностью, рис. 6.6 *а*. Если же фокусное расстояние слишком велико а угол α' слишком мал, изображение будет близким к аксонометрическому и не будет передавать ощущения глубины пространства, рис. 6.6 *б*. При равенстве углов α' и α изображение будет наиболее естественным, рис. 6.6 *в*.

Проецирование на плоскость изображений с большим углом обзора приводит к значительным искажениям, что видно на рис. 6.6 *а*. Поэтому для получения изображений, охватывающих значительную область пространства вокруг зрителя (*панорамных*) используют проецирование на криволинейные поверхности — *цилиндрическую* и *сферическую*.

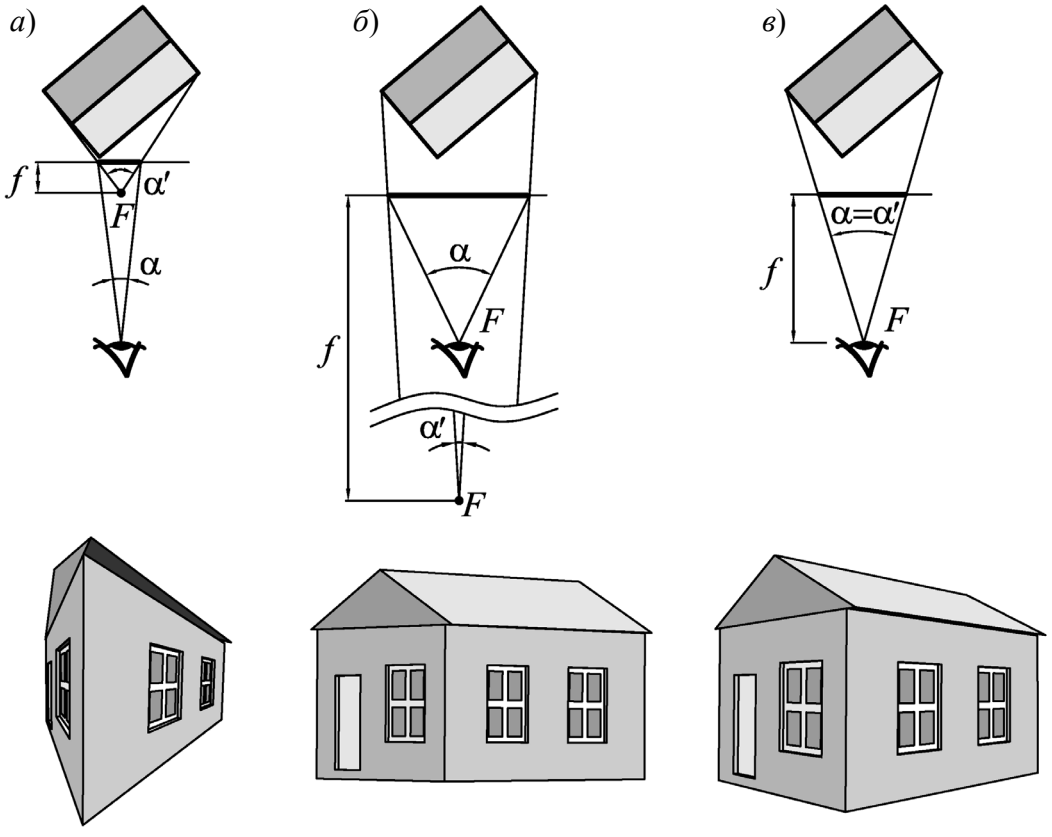


Рисунок 6.6 – Перспективное изображение, построенное при $\alpha' > \alpha$ (*а*), $\alpha' < \alpha$ (*б*) и $\alpha' = \alpha$ (*в*)

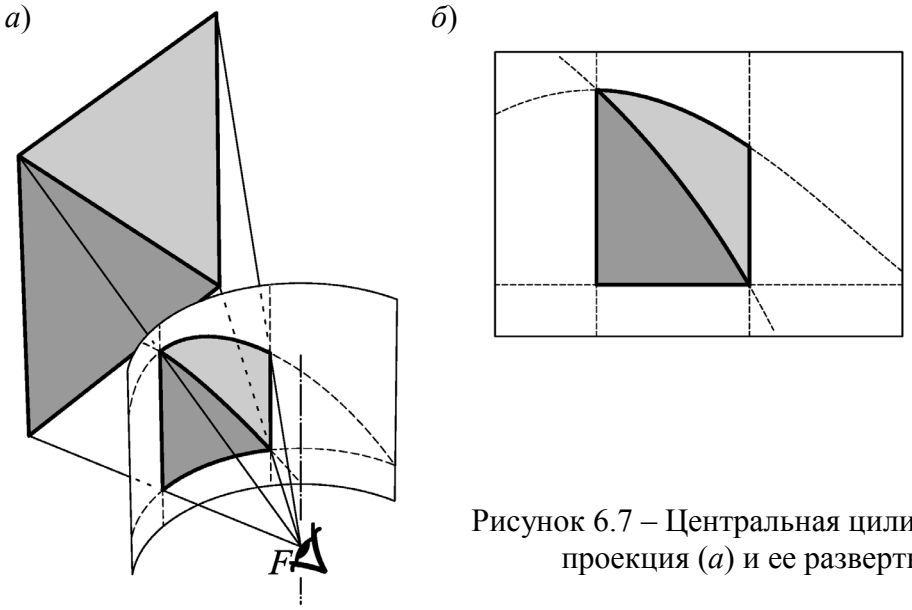


Рисунок 6.7 – Центральная цилиндрическая проекция (*а*) и ее развертка (*б*)

В ***цилиндрической проекции*** точка фокуса располагается на оси цилиндра (обычно вертикальной), а изображение строится по точкам пересечения лучей, идущих из точки фокуса к проецируемым объектам, с поверхностью цилиндра, рис. 6.7. Пример развернутого изображения, полученного цилиндрической проекцией, показан на рис. 6.8.

Характерным для цилиндрической проекции является то, что:

— *размеры* изображений объектов уменьшаются по мере их удаления от оси цилиндра *независимо от направления*, при этом небольшие объекты изображаются без значительных искажений, даже если они расположены под большим углом по горизонтали к оси взгляда;

— *прямыми* на плоской развертке проекции отображаются только отрезки прямых, лежащих в плоскости, которая проходит через ось цилиндра или через точку фокуса перпендикулярно оси;

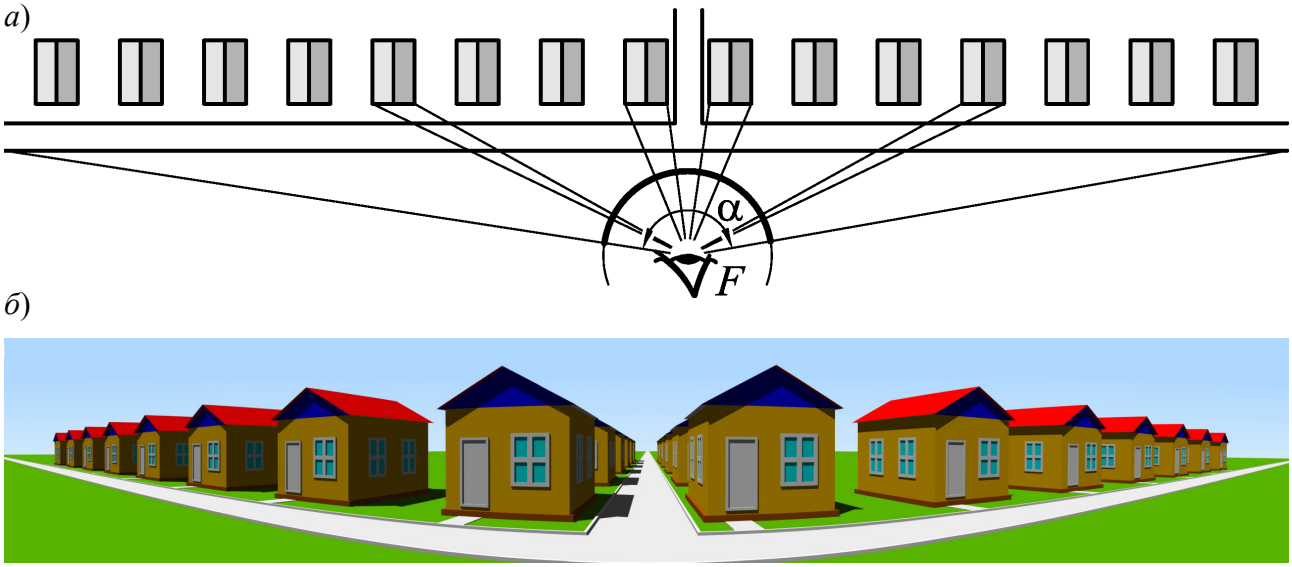


Рисунок 6.8 – Получение панорамного изображения в центральной цилиндрической проекции (*а*) и его развертка (*б*)

— отрезки любых других прямых на поверхности цилиндра образуют *дуги эллипса*, а после развертывания изображения на плоскости отображаются участками *синусоиды*.

Основным преимуществом цилиндрической проекции перед проекцией на плоскость является большой угол съемки, который может достигать 360° — в таком случае получаем *круговую панораму*. Следует также отметить, что искривление прямых проявляется лишь на плоской развертке. Для зрителя, который видит изображение на цилиндрическом экране, расположившись в точке фокуса (или достаточно близко к ней), любые искажения отсутствуют. Поэтому цилиндрическая проекция используется в широкоэкранных «панорамных» кинотеатрах, где изображение проецируется на изогнутый экран — зритель видит события, происходящие не только перед ним, но и слева, справа и даже сзади.

В ***сферической проекции*** поверхностью проецирования служит *сфера*, а точка фокуса располагается в ее центре. Сферическая проекция во многом сходна с цилиндрической, но позволяет получить большой угол съемки как по горизонтали, так и по вертикали — вплоть до отображения всего окружающего пространства. Сферическая проекция позволяет наиболее точно и полно отобразить то, что видит человек, и, возможно, будет использоваться в панорамных кинотеатрах будущего. В настоящее время сферическая проекция используется в планетариях для имитации звездного неба на полусферическом экране.

6.3 Модели описания поверхностей

Для того, чтобы синтезировать на компьютере изображение некоторого трехмерного объекта (или группы объектов), необходимо иметь описание этого объекта — ***математическую модель***. Математическое моделирование трехмерных объектов используется в САПР, художественной, кинематографической графике, 3D-играх и симуляторах, при этом подходы к моделированию поверхностей объектов могут быть различными. В настоящее время известны следующие методы.

1. ***Аналитическая модель*** — описание поверхности математическими формулами. Поверхность может быть задана в виде *функции двух аргументов*:

$$z = f\left(x, y\right), \tag{6.15}$$

или виде *уравнения*:

$$F\left(x, y, z\right) = 0, \tag{6.16}$$

но наиболее часто встречается *параметрическая форма* описания поверхности:

$$\begin{cases} x = f_x(s,t), \\ y = f_y(s,t), \\ z = f_z(s,t), \end{cases} \tag{6.17}$$

где f_x, f_y, f_z – функции, определяющие форму поверхности; s и t – параметры этих функций. Параметрическая форма позволяет проще описывать сложные поверхности, в том числе замкнутые и поверхности, соответствующие неоднозначным функциям. Например, сферу можно описать как функцию двух аргументов, уравнением и в параметрической форме следующим образом:

$$z = \pm\sqrt{R^2 - x^2 - y^2}\,; \tag{6.18}$$

$$x^2 + y^2 + z^2 - R^2 = 0; \tag{6.19}$$

$$\begin{cases} x = R\sin s\cos t, \\ y = R\sin s\sin t, \\ z = R\cos t. \end{cases} \tag{6.20}$$

Для описания сложных поверхностей часто используют ***сплайны*** — специальные функции, аппроксимирующие отдельные фрагменты сложной поверхности. При этом модель сложной поверхности образуется несколькими сплайнами.

Обычно используют *кубические* сплайны, заданные в параметрической форме, поскольку третья степень является наименьшей, позволяющей описывать любую форму и при стыковке сплайнов обеспечивать непрерывную вторую производную — получать гладкую поверхность без изломов. В САПР и 3D-графике получили распространение *сплайны Безье* и *NURBS*.

Поверхность Безье в общем виде описывается зависимостью

$$\begin{cases} x(s,t) = \sum\limits_{i=0}^{m}\sum\limits_{j=0}^{n} C_i s^i (1-s)^{m-i} C_j t^j (1-t)^{n-i} x_{ij}, \\ y(s,t) = \sum\limits_{i=0}^{m}\sum\limits_{j=0}^{n} C_i s^i (1-s)^{m-i} C_j t^j (1-t)^{n-i} y_{ij}, \\ z(s,t) = \sum\limits_{i=0}^{m}\sum\limits_{j=0}^{n} C_i s^i (1-s)^{m-i} C_j t^j (1-t)^{n-i} z_{ij}, \end{cases} \tag{6.21}$$

где x_{ij}, y_{ij} и z_{ij}– координаты точек-ориентиров P_{ij}; C_i и C_j – коэффициенты бинома Ньютона. Зависимость имеет степенть $m \times n$ и однозначно описывается сеткой из $(m + 1)(n + 1)$ опорных точек. Параметры s и t изменяются в диапазоне от 0 до 1; каждой паре их значений соответствует некоторая точка на поверхности.

Кубический сплайн Безье соответствует $m = n = 3$:

$$\begin{cases} x(s,t) = (1-s)^3(1-t)^3 x_{0,0} + 3(1-s)^3 t\,(1-t)^2 x_{0,1} + 3(1-s)^3 t^2(1-t)\,x_{0,2} + \\ \quad + (1-s)^3 t^3 x_{0,3} + 3s(1-s)^2(1-t)^3 x_{1,0} + 9st(1-s)^2(1-t)^2 x_{1,1} + \\ \quad + 9st^2(1-s)^2(1-t)\,x_{1,2} + 3st^3(1-s)^2 x_{1,3} + 3s^2(1-s)(1-t)^3 x_{2,0} + \\ \quad + 9s^2 t(1-s)(1-t)^2 x_{2,1} + 9s^2 t^2(1-s)(1-t)\,x_{2,2} + 3s^2 t^3(1-s)x_{2,3} + \\ \quad + s^3(1-t)^3 x_{3,0} + 3s^3 t\,(1-t)^2 x_{3,1} + 3s^3 t^2(1-t)\,x_{3,2} + s^3 t^3 x_{3,3}, \\ y(s,t) = (1-s)^3(1-t)^3 y_{0,0} + 3(1-s)^3 t\,(1-t)^2 y_{0,1} + \ \ldots + 3s^3 t^2(1-t)\,y_{3,2} + s^3 t^3 y_{3,3}, \\ z(s,t) = (1-s)^3(1-t)^3 z_{0,0} + 3(1-s)^3 t\,(1-t)^2 z_{0,1} \ + \ldots + 3s^3 t^2(1-t)\,z_{3,2} + s^3 t^3 z_{3,3}, \end{cases} \tag{6.22}$$

и может быть задан сеткой из $4 \times 4 = 16$ опорных точек, рис. 6.9.

Кубическая поверхность Безье может принимать весьма разнообразные формы, что позволяет использовать ее для моделирования сложных поверхностей.

Поверхности NURBS сходны с поверхностями Безье, но задаются сеткой из произвольного, в т.ч. весьма большого числа опорных точек, могут быть замкнутыми. Сплайны *NURBS* позволяют аппроксимировать сложные поверхности (например, поверхности живых объектов) проще, чем сплайны Безье.

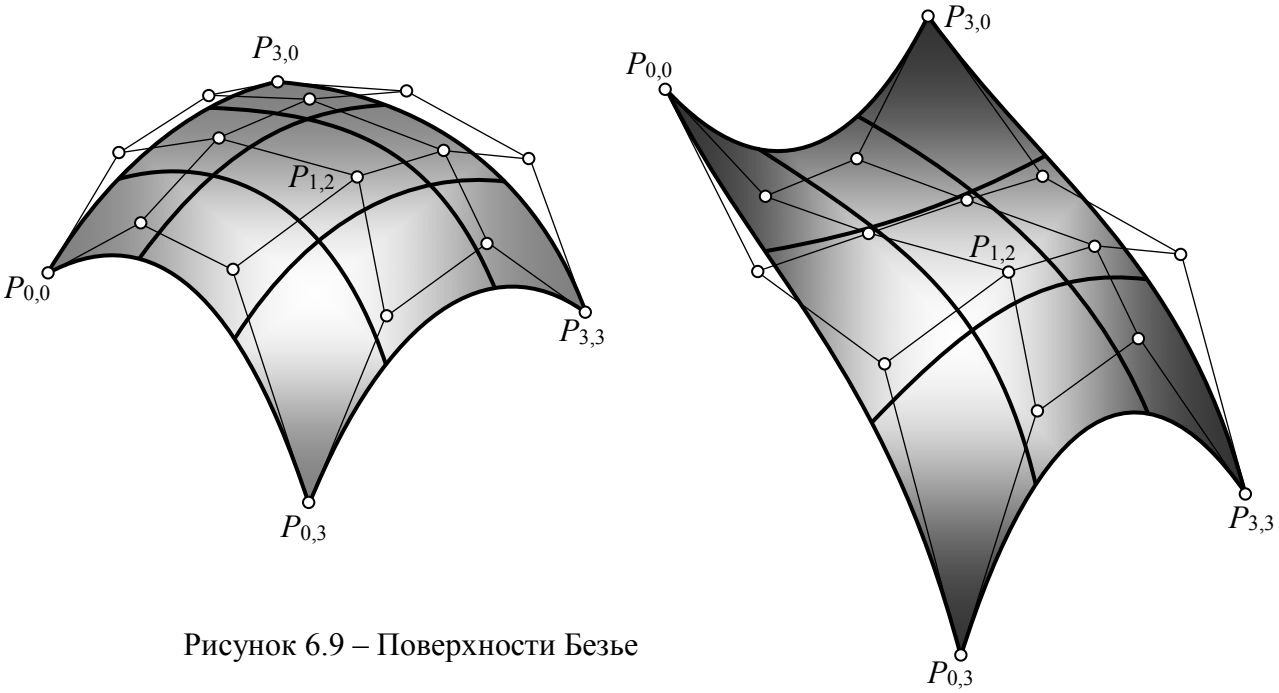


Рисунок 6.9 – Поверхности Безье

Существуют несколько вариаций аналитического метода описания поверхностей и тел, используемых в различных областях моделирования. В САПР для создания моделей тел типа деталей машин, которые характеризуются относительно простыми формами, применяется ***твердотельное моделирование***. Известны два метода твердотельного моделирования: *метод геометрических примитивов* и *кинематический метод*.

Метод геометрических примитивов предполагает создание модели объекта из набора стандартных объемных фигур, которые достаточно просто описываются аналитически. Обычно это прямоугольный параллелепипед, цилиндр, конус, шар, тороид, рис. 6.10 *а*. Модель объекта создается путем выполнения топологических операций над примитивами: сложением, вычитанием, пересечением, усечением плоскостью, рис. 6.10 *б*, *в*.

Кинематический метод позволяет описать поверхности как след в пространстве от движения некоторой линии (образующей) по заданной траектории. Обычно используют движение вдоль отрезка прямой (*выдавливание*), *вращение*, а также движение вдоль *произвольной траектории*, рис. 6.10 *г*

Твердотельные модели дают информацию не только о поверхности объекта, но и о его объеме, массе. Кроме того, твердотельные аналитические

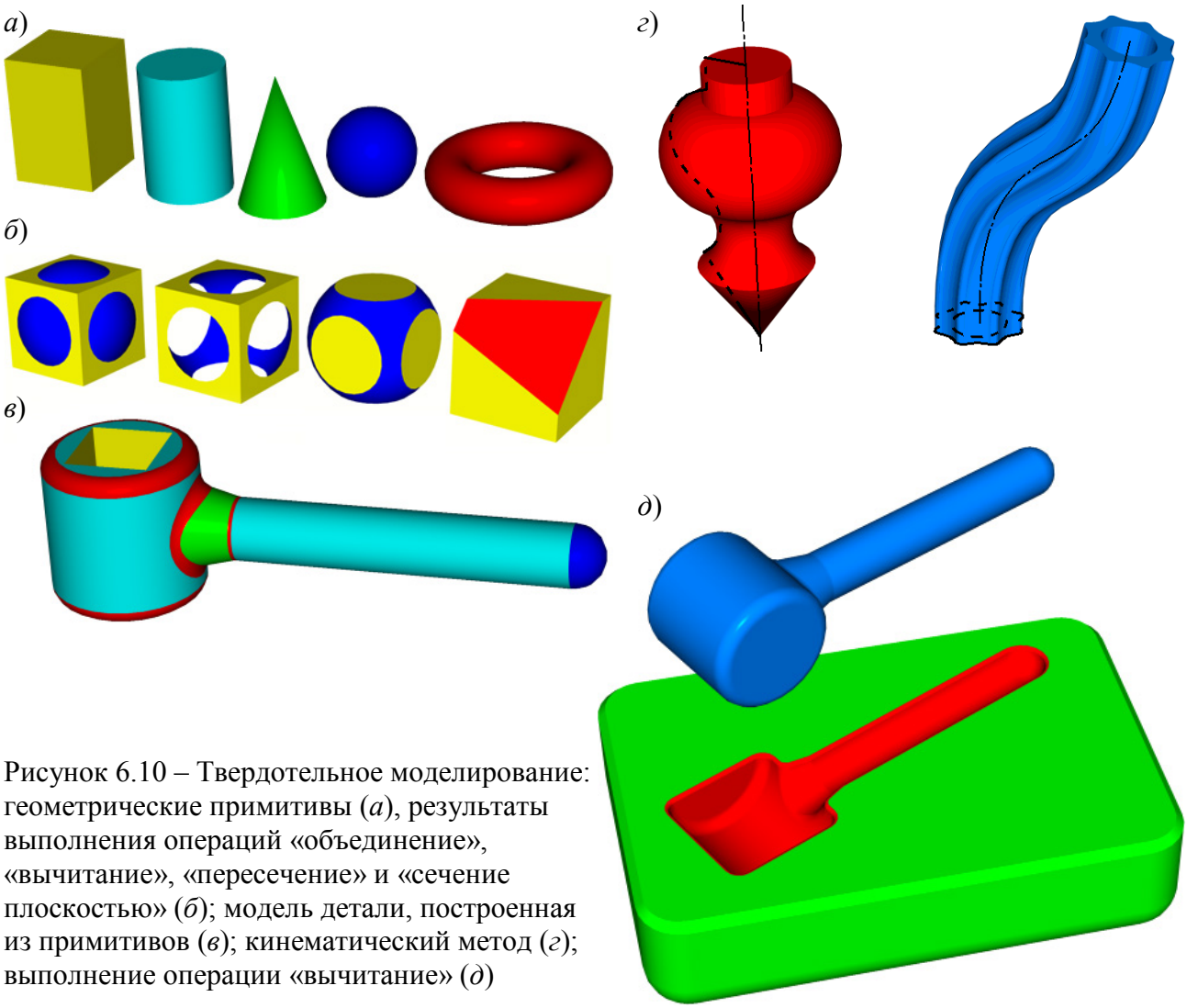


Рисунок 6.10 – Твердотельное моделирование: геометрические примитивы (*а*), результаты выполнения операций «объединение», «вычитание», «пересечение» и «сечение плоскостью» (*б*); модель детали, построенная из примитивов (*в*); кинематический метод (*г*); выполнение операции «вычитание» (*д*)

модели позволяют выполнять *топологические операции* над объектами: получать разрезы, сечения, суммировать и вычитать объекты. Например, чтобы получить модель прессформы, достаточно из ее заготовки вычесть модель готовой детали, рис. 6.10 *г* Твердотельные модели дают возможность находить области взаимного пересечения объектов, например, для анализа ошибок взаимного расположения деталей в проектируемом узле.

В целом *аналитические модели* наиболее пригодны для операций анализа поверхностей, позволяют достаточно просто рассчитать координаты вектора нормали к каждой точке поверхности. Недостатки аналитических моделей — сложность формул описания, необходимость выполнения больших объемов вычислений для построения реалистических изображений поверхностей. Поэтому в интерактивной 3D-анимации (играх, симуляторах) аналитические модели поверхностей не используются. Основная область применения аналитических моделей — САПР и высококачественная 3D-анимация (художественные фильмы).

Для ускорения отображения объектов, заданных аналитическими моделями, большинство программ выполняют их предварительное преобразование в *полигональную модель*, которая визуализируется с помощью более простых и быстродействующих алгоритмов.

2. ***Полигональные модели*** аппроксимируют поверхности участками *плоскостей*, ограниченными прямыми линиями — ***полигонами***.

Чаще всего, полигон представляет собой *треугольник*, заданный координатами трех его ***вершин***, рис. 6.11 *а*. Представление поверхностей треугольными гранями широко используется в различных областях — компьютерных играх, художественной графике, САПР. Треугольник — базовый элемент для современных видеоадаптеров с 3D-акселераторами. Объясняется это, прежде всего, простотой алгоритмов построения изображения треугольника, определения нормали, освещенности, отсечения невидимых участков и т.д.

Чтобы создать *полигональную модель* некоторой криволинейной поверхности (ее называют еще *полигональной сеткой*), необходимо задать в пространстве набор полигонов, соприкасающихся друг с другом по граням и имеющих общие вершины. Наиболее просто это сделать, задав ***массив координат вершин*** размером $m \times n$, при этом вершинами некоторого полигона будут точки $P_{i,j}$, $P_{i+1,j}$ и $P_{i,j+1}$, рис. 6.11 *б*. Такая модель наиболее экономна по занимаемой памяти, поскольку координаты вершин для всех смежных полигонов записываются один раз. Но при этом необходимость задания прямоугольного массива накладывает некоторые ограничения на возможности моделирования: усложняет стыковку с другими поверхностями, не позволяет получать разрывы, «отверстия» в поверхности, на некоторых участках приводит к переизбытку вершин.

Более гибкой является ***векторная полигональная модель***, в которой каждый полигон представляет собой отдельный элемент, рис. 6.11 *в*. Чтобы избежать многократной перезаписи координат вершин для смежных полигонов, первоначально в памяти формируется *массив координат вершин*, каждой из которых присваивается номер. Теперь, чтобы задать полигон, достаточно

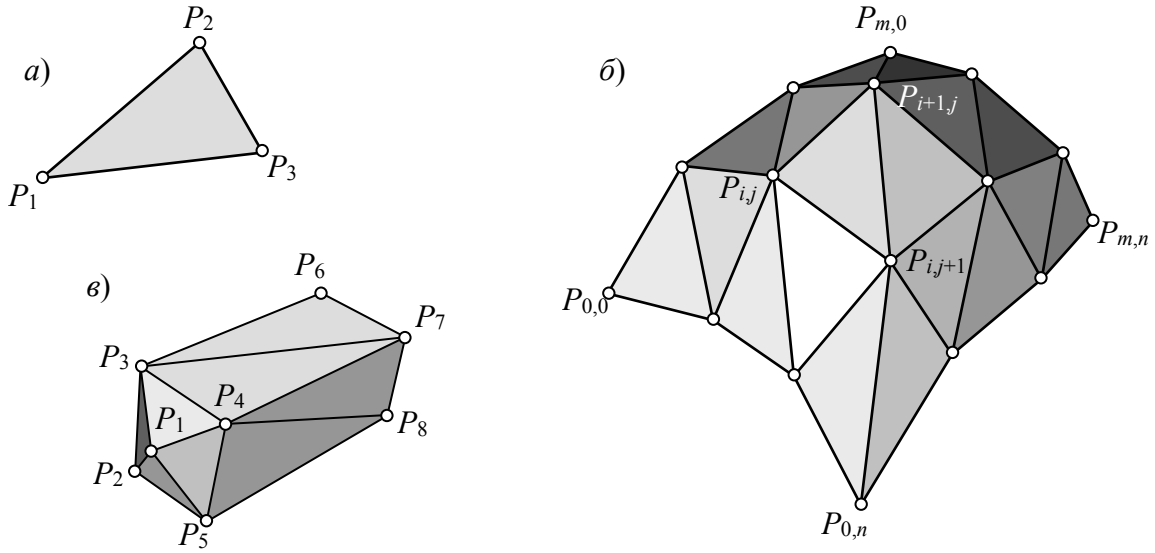


Рисунок 6.11 – Полигон (*а*), полигональная сетка (*б*) и линейно-узловая модель (*в*)

указать номера его вершин. Кроме того, такой способ облегчает редактирование полигональной модели — изменив координату некоторой вершины, мы автоматически изменим все смежные с ней полигоны.

Часто используют более сложную ***линейно-узловую модель***, в которой кроме массива вершин создается массив ребер — каждое ребро задается номерами двух вершин. Полигон же задается указанием номеров ограничивающих его ребер. Такая модель позволяет задавать свойства ребер, необходимые для построения реалистических изображений поверхностей (например, нужно ли это ребро «сглаживать», или оно должно быть острым), а также исключает многократную прорисовку ребер при отображении «проволочной модели» поверхности.

Положительные черты полигональных моделей следующие:

— удобство перемещения, вращения и масштабирования объектов — достаточно для всех вершин выполнить соответствующее преобразование координат;

— простые и быстродействующие алгоритмы визуализации, аппаратная поддержка этих алгоритмов в современных графических системах, что позволяет реализовать интерактивную анимированную 3D-графику — создавать 10…20 изображений (кадров) в секунду.

Наряду с этим следует выделить недостатки полигональных моделей:

— погрешности моделирования при аппроксимации плоскими гранями криволинейных поверхностей; для получения достаточно точных моделей необходимо использовать очень большое число полигонов, что увеличивает объем памяти, занимаемый описанием объекта и снижает скорость построения изображения;

— для создания реалистических изображений объектов необходимо использовать специальные алгоритмы, «скрывающие» грани;

— существенно усложняется выполнение топологических операций над объектами: получение разрезов и сечений, нахождение областей пересечения объектов и т.п.

3. ***Воксельная модель*** — трехмерный растр.

Воксел — элемент объема (*voxel — volume element*). По аналогии с пикселами обычного растра вокселы заполняют объем в трехмерном растре, располагаясь в узлах равномерной сетки. Любой воксел может иметь свой цвет и, кроме того, прозрачность. Полная прозрачность воксела означает пустоту соответствующей точки объема.

Основная характеристика воксельной модели — *разрешающая способность* — количество вокселов в определенном объеме. Она определяет точность моделирования трехмерных объектов.

Представление объемных объектов в виде воксельных моделей используют в компьютерных системах, предназначенных для медицинских целей. Например, при сканировании томографом получаются изображения срезов объекта, которые потом объединяются в виде объемной модели для дальнейшего анализа. Воксельный метод используется также для графических устройств, создающих действительно объемные изображения.

Воксельная модель обеспечивает простое описание сложных объектов и сцен, упрощает процедуру отображения (для отображения воксельной модели достаточно изобразить «кубики» соответствующего цвета во всех точках сетки растра, кроме «прозрачных» точек), не требует выполнения процедур отсечения невидимых частей объектов (достаточно отображать слои вокселов в порядке от дальнего к ближнему, при этом ближние вокселы перекроют дальние). Кроме того, упрощается выполнение топологических операций, например — выполнение разрезов, рис. 6.11, — нужно просто сделать «невидимыми» определенные группы вокселов.

Основной недостаток воксельной модели, ограничивающий область ее применения — чрезвычайно большое количество информации, необходимое для представления объемных данных с достаточно большим разрешением. Например, объем 256×256×256 имеет не слишком большую разрешающую способность, но содержит свыше 16 миллионов вокселов. Значительные затраты памяти ограничивают разрешающую способность, точность моделирования, скорость вывода изображения. Кроме того, как и обычный растр, воксельная модель затрудняет редактирование, масштабирование объектов.

Несмотря на указанные недостатки, с ростом возможностей компьютерной техники воксельные модели могут найти значительно большее применение в различных областях.

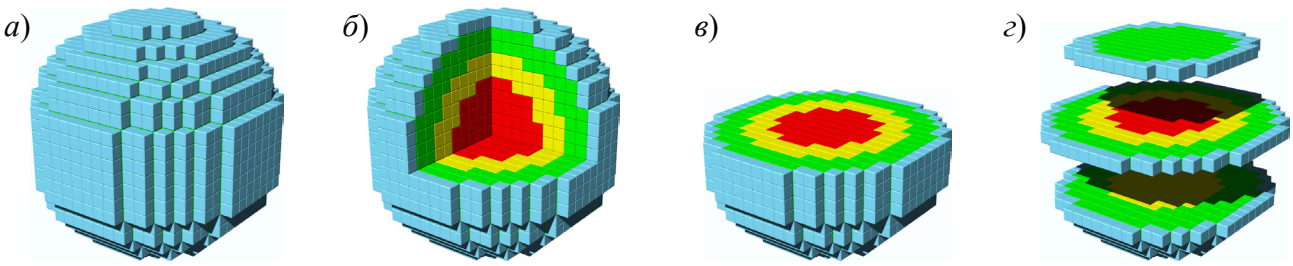


Рисунок 6.12 – Воксельная модель размером 20×20×20 (*а*) и ее разрезы (*б*, *в*, *г*)

7. ВИЗУАЛИЗАЦИЯ ТРЕХМЕРНЫХ ОБЪЕКТОВ

7.1 Способы и уровни визуализации

Любой трехмерный объект может быть изображен по-разному, в зависимости от назначения создаваемого изображения и используемого при этом алгоритма. ***Способы визуализации*** по характеру получаемых изображений и степени сложности используемых алгоритмов можно разделить на такие уровни:

1. ***Каркасная визуализация*** («*проволочноая модель*») заключается в отрисовке всех элементов каркаса модели — ребер полигонов, дуг эллипсов, сплайновых кривых и т.п. При этом поверхности модели «прозрачны» и видны все элементы объектов, рис. 7.13 *а*.

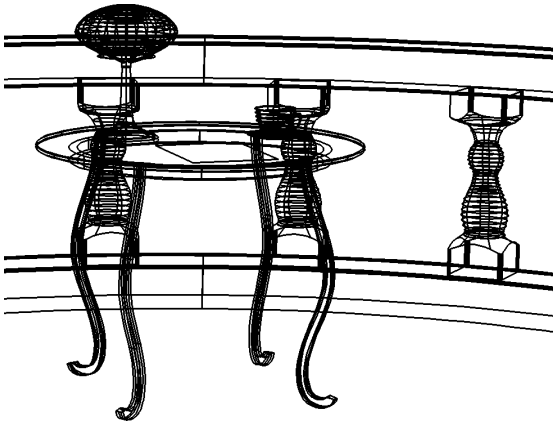
Для построения изображения необходимо преобразовать координаты вершин полигонов и базовых точек кривых из мировых в экранные координаты, после чего используются простые растровые алгоритмы вывода линий. Каркасная визуализация — наиболее быстрая и используется для отображения моделей объектов в процессе редактирования.

2. ***Визуализация с удалением невидимых точек*** предусматривает отрисовку только тех элементов объекта, которые не скрыты от наблюдателя другими объектами, рис. 7.13 *б*. Существует несколько методик удаления невидимых элементов.

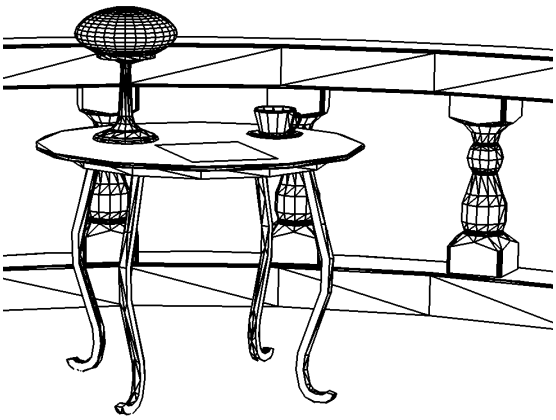
Наиболее простой метод, применяемый для полигональных моделей — ***сортировка граней по глубине***. Полигоны рисуются в порядке от самых дальних к самым ближним; при заливке ближних полигонов стираются невидимые части дальних. Метод наиболее эффективен при построении поверхностей, заданных функциями $z = f(x, y)$, рис. 7.14 *а*, и используется в основном математических пакетах. При отрисовке сложных объектов, грани которых могут располагаться произвольным образом, пересекаться, рис. 7.14 *б*, метод дает ошибки. Кроме того, метод требует обязательного заполнения полигонов, что замедляет процесс отображения.

В ***методе плавающего горизонта*** грани выводятся в обратном порядке — от самых ближних к самым дальним. На каждом шаге границы граней образуют две ломанные линии — *верхний* и *нижний горизонты*, которые запоминаются в виде двух одномерных массивов. Каждому значению координаты y растра соответствуют значения верхней и нижней границы. При выводе следующей грани рисуется только то, что выше верхнего или ниже нижнего горизонтов. Каждая следующая грань поднимает верхний и опускает нижний горизонты соответственно своим границам. Метод имеет те же ограничения и ту же область применения, что и сортировка граней по глубине, но более быстродействующий, поскольку не требует заливки граней (можно рисовать только ребра) и позволяет не тратить время на отрисовку невидимых элементов. Метод также применяется при изображении математических поверхностей плавными кривыми, а не полигонами, рис. 7.14 *в*.

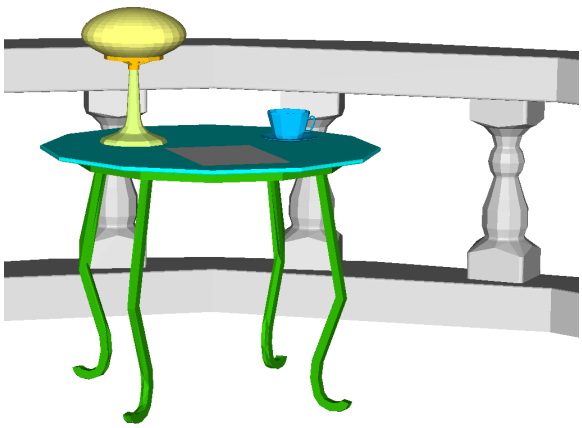
а)



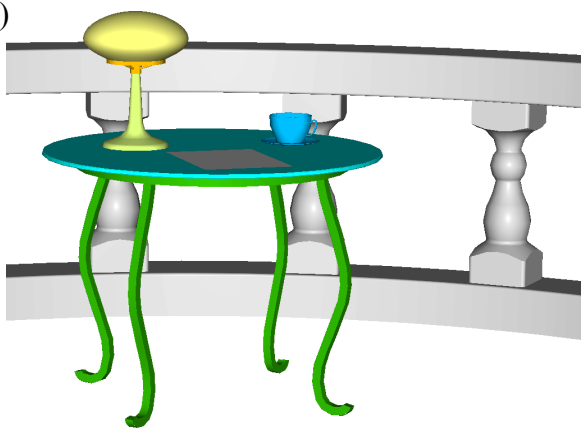
б)



в)



г)



д)

Рисунок 7.13 Уровни визуализации 3D-модели: каркасная (*а*); с удалением невидимых точек (*б*); закрашивание граней с учетом освещения (*в*); имитация гладких поверхностей (*г*); наложение текстуры, имитация теней, отражения и преломления, трассировка лучей (*д*)

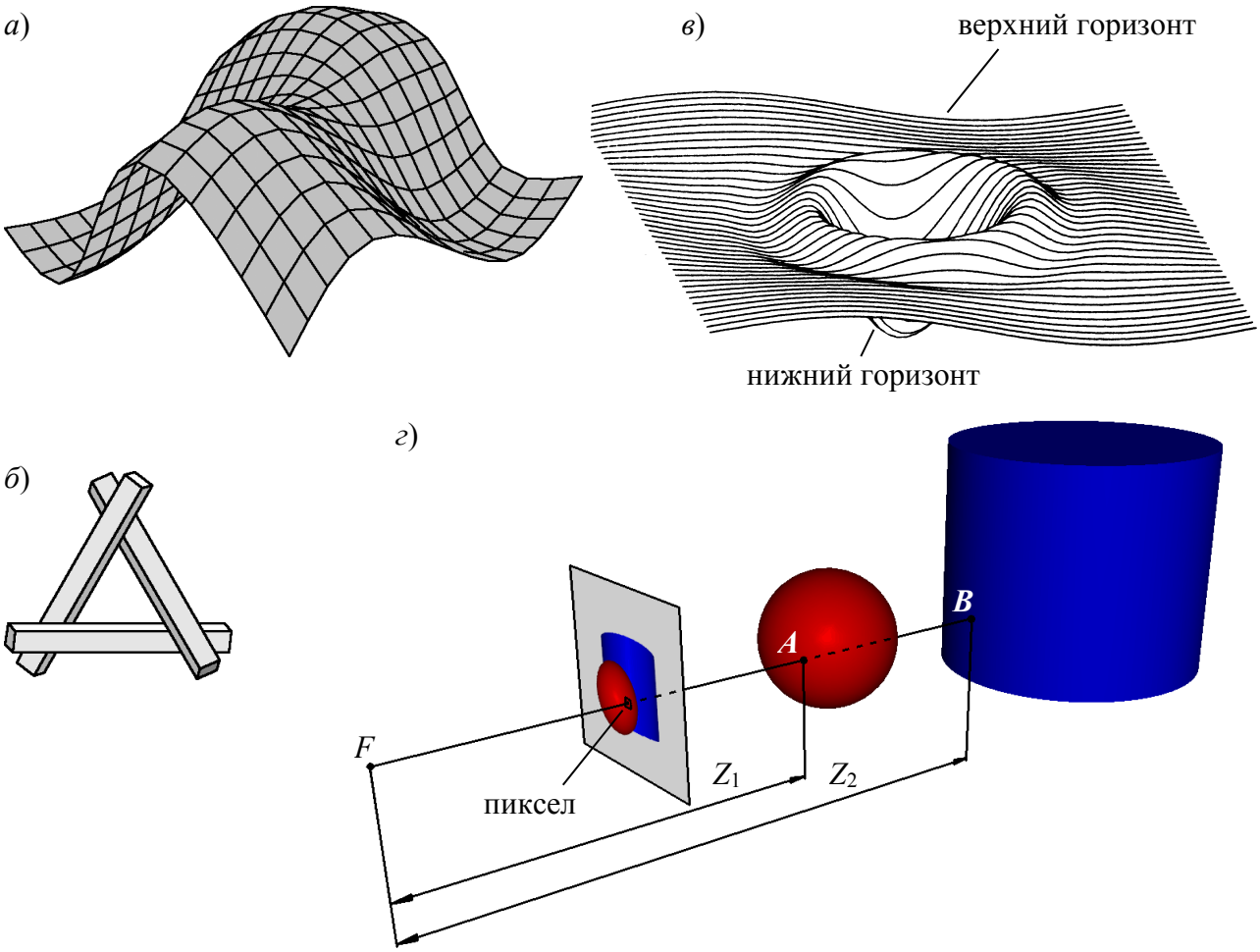


Рисунок 7.14 – Удаление невидимых элементов методом сортировки граней по глубине (*а*) и объект, который невозможно отобразить этим методом (*б*); методы плавающего горизонта (*в*) и *Z*-буфера (*г*)

Метод Z-буфера основан на использовании специального участка памяти, в котором для каждого пиксела изображения записывается значение координаты *z* (в видовых координатах) точки поверхности трехмерного объекта, проецируемой в область этого пиксела.

Рассмотрим пример, изображенный на рис. 7.14 *г*. Перед построением изображения *Z*-буфер инициализируется — заполняется максимальными значениями координаты *Z*. Проекционный луч, проходящий через некоторый пиксел растра, пересекает объекты *сфера* и *цилиндр*. Пусть первым изображается сфера. В *Z*-буфер записывается координата Z_1 точки пересечения проекционного луча с поверхностью сферы (точка *A*). При построении изображения цилиндра в этот же пиксел будет проецироваться точка *B* с координатой Z_2, большей, чем Z_1. Следовательно, эта точка будет проигнорирована.

Метод *Z*-буфера сейчас основной в 3D-графике, поскольку позволяет правильно изображать пересекающиеся поверхности, поддерживается аппаратно современными графическими системами. Недостатком метода можно признать большой объем вычислений при построении изображений, поскольку

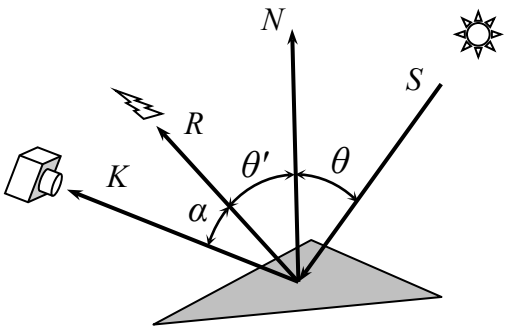


Рисунок 7.15 – Расположение векторов нормали, источника света, отраженного луча и направления взгляда

для каждого пиксела нужно находить точки пересечения проецирующего луча со всеми отображаемыми поверхностями.

3. ***Закрашивание поверхностей*** предполагает использование алгоритмов, имитирующих отражение света реальными поверхностями тел для создания более-менее реалистических изображений.

При вычислении цвета точек поверхности используется ***алгебра векторов***. Расположение поверхности характеризуется ***вектором нормали*** N к поверхности в данной точке, рис. 7.15. Направление лучей падающего и отраженного света, а также направление взгляда (луча проецирования) также задается векторами S, R и K соответственно.

Поверхности реальных объектов обладают рядом характеристик, влияющих на характер отражения ими света: *светлота* и *цвет* объекта, *шероховатость* поверхности, прозрачность и др. В компьютерной графике для имитации этих свойств используются несколько ***моделей отражения***.

Зеркальное отражение света характерно для *гладких поверхностей*. Угол θ между падающим лучом и нормалью равен углу θ' между нормалью и отраженным лучом. Падающий луч, отраженный луч и нормаль расположены в одной плоскости.

Поверхность считается *идеально гладкой*, если размер шероховатостей на ней намного меньше длины световой волны. В таком случае вся световая энергия падающего луча отражается только по линии отраженного луча, любое рассеивание в стороны от этой линии отсутствует, рис. 7.16 *а*.

Луч света, попадающий на поверхность *неидеального зеркала*, порождает не один, а множество отраженных лучей, рассеянных вокруг направления идеально отраженного луча, рис. 7.16 *б*. Интенсивность луча, отраженного в некотором направлении (например, в направлении камеры) зависит от угла α, см. рис. 7.15. Эта зависимость описывается эмпирической ***моделью Фонга***:

$$I_з = I \cos^p \alpha, \tag{7.23}$$

где I и $I_з$ – интенсивность падающего и отраженного света; p – показатель, зависящий от качества полировки поверхности, рис. 7.17 *а*, *б*.

Диффузное отражение присуще матовым поверхностям, имеющим шероховатости большие, чем длина световой волны. Примерами могут служить

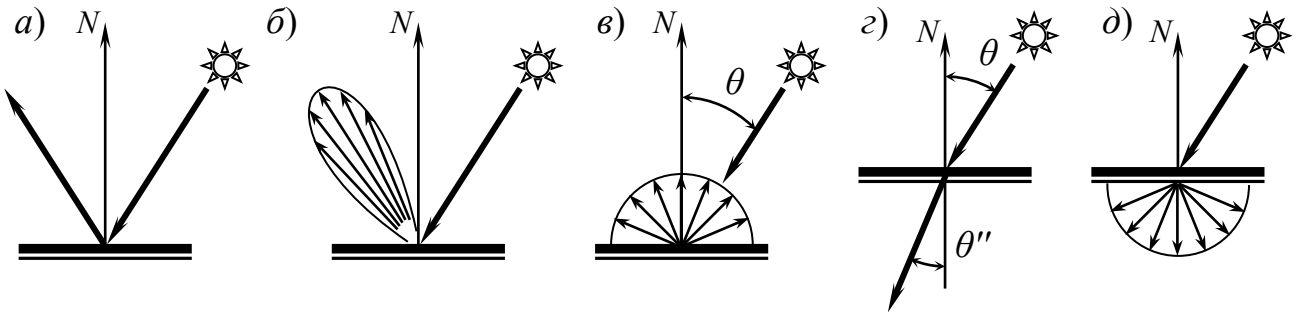


Рисунок 7.16 – Оптические модели: идеальное зеркало (*а*), неидеальное зеркало Фонга (*б*); диффузное отражение (*в*), идеальное преломление (*г*) и диффузное преломление (*д*)

бумага, гипс, песок. Падающий на такую поверхность луч света рассеивается равномерно во все стороны, рис. 7.16 *в*. Диффузное отражение описывается ***законом Ламберта***, согласно которому интенсивность отраженного света пропорциональна косинусу угла θ между лучом падающего света и нормалью:

$$I_{\text{д}} = I\cos\theta\,. \tag{7.24}$$

Пример объекта с диффузным отражением приведен на рис. 7.17 *в*.

Зеркальная и диффузная модели при некоторых углах α и θ дают нулевые значения интенсивности. Это значит, что участки объекта будут абсолютно черными. Поскольку в реальных сценах помимо прямого освещения обычно присутсвует *рассеянный свет*, идущий от неба или отраженный от других объектов, в модель вводится имитация ***рассеянного освещения*** с некоторой интенсивностью, не зависящей от направления источника освещения — все точки поверхности объекта освещаются равномерно, рис. 7.17 *г*.

Для прозрачных объектов используются модели ***идеального*** и ***диффузного преломления***, рис. 7.16 *г* и *д*. Характеристикой прозрачной поверхности с идеальным преломлением является ***коэффициент преломления*** n, определяющий отклонение преломленного луча

$$\cos\theta'' = \frac{\cos\theta}{n}. \tag{7.25}$$

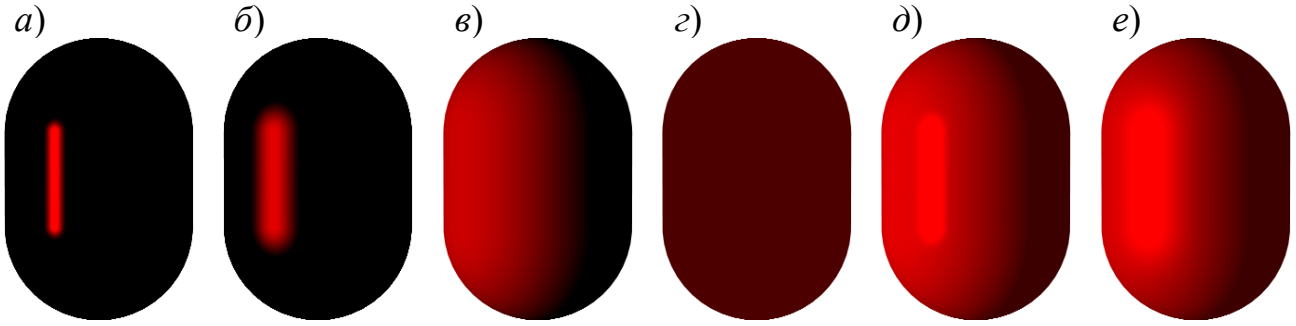


Рисунок 7.17 – Закрашивание поверхностей с отражением по модели Фонга при различной степени шероховатости (*а*, *б*), по диффузной модели (*в*); рассеянное освещение (*г*); сочетание моделей отражения Фонга, диффузной и рассеянной с различными параметрами (*д*, *е*)

где θ'' – угол преломленного луча.

Параметры отражения света поверхностями объектов моделируются как сумма интенсивностей света, рассчитанных по различным моделям, с учетом весовых коэффициентов, определяющих степень влияния каждой модели, рис. 7.17 *д*, *е*:

$$I_{\text{о}} = k_{\text{з}} I_{\text{з}} + k_{\text{д}} I_{\text{д}} + k_{\text{р}} I_{\text{р}} + k_{\text{п}} I_{\text{п}}, \qquad (7.26)$$

где $I_{\text{о}}$ – цвет точки объекта; $I_{\text{р}}$ – интенсивность рассеянного освещения; $I_{\text{п}}$ – интенсивность преломленного луча; $k_{\text{з}}$, $k_{\text{д}}$, $k_{\text{р}}$, $k_{\text{п}}$ – коэффициенты зеркальности, шероховатости, рассеянного освещения и прозрачности объекта.

4. ***Имитация гладких поверхностей*** *полигональных моделей*.

Поскольку полигональные модели (являющиеся наиболее распространенным типом моделей 3D-поверхностей) представляют собой *набор плоских граней*, наиболее простой метод их закрашивания следующий. Для каждой грани определяем направление *вектора нормали*, рис. 7.18 *а*, и в соответствии с моделью оптических свойств материала поверхности и направлением луча освещения вычисляем *цвет* этой грани. Затем с помощью простого алгоритма закрашивания полигона (см п. 5.4) заполняем контур грани этим цветом.

Полученное таким методом изображение получается «*граненым*», каждая из граней полигональной сетки четко видна, рис. 7.18 *д* и 7.13 *в*. Поэтому метод используется лишь на этапе разработки модели для ее визуализации в режиме реального времени — он дает наибольшую скорость генерации кадров изображения.

Для создания более реалистичных изображений необходимо «сгладить» неровности полигональной аппроксимации. Увеличение количества граней при аппроксимации криволинейных объектов дает эффект только в том случае, если видимый размер полигонов приближается к размеру пиксела. При этом модель получится слишком сложной.

Существуют методы, позволяющие «сгладить» границы между гранями, создав иллюзию гладкой поверхности: методы *Гуро* и *Фонга*.

Метод Гуро предполагает следующую последовательность операций:

— вычисляются нормали к граням, рис. 7.18 *а*;

— вычисляются «нормали» в вершинах полигональной сетки как усреднение векторов нормалей примыкающих к этой вершине граней, рис. 7.18 *б*;

— опираясь на «нормали» вершин, с учетом направления падающего света и оптических свойств материала поверхности для каждой вершины вычисляется цвет;

— полигон грани закрашивается цветами; соответствующими *линейной интерполяции* цветов его вершин, рис. 7.18 *в.*

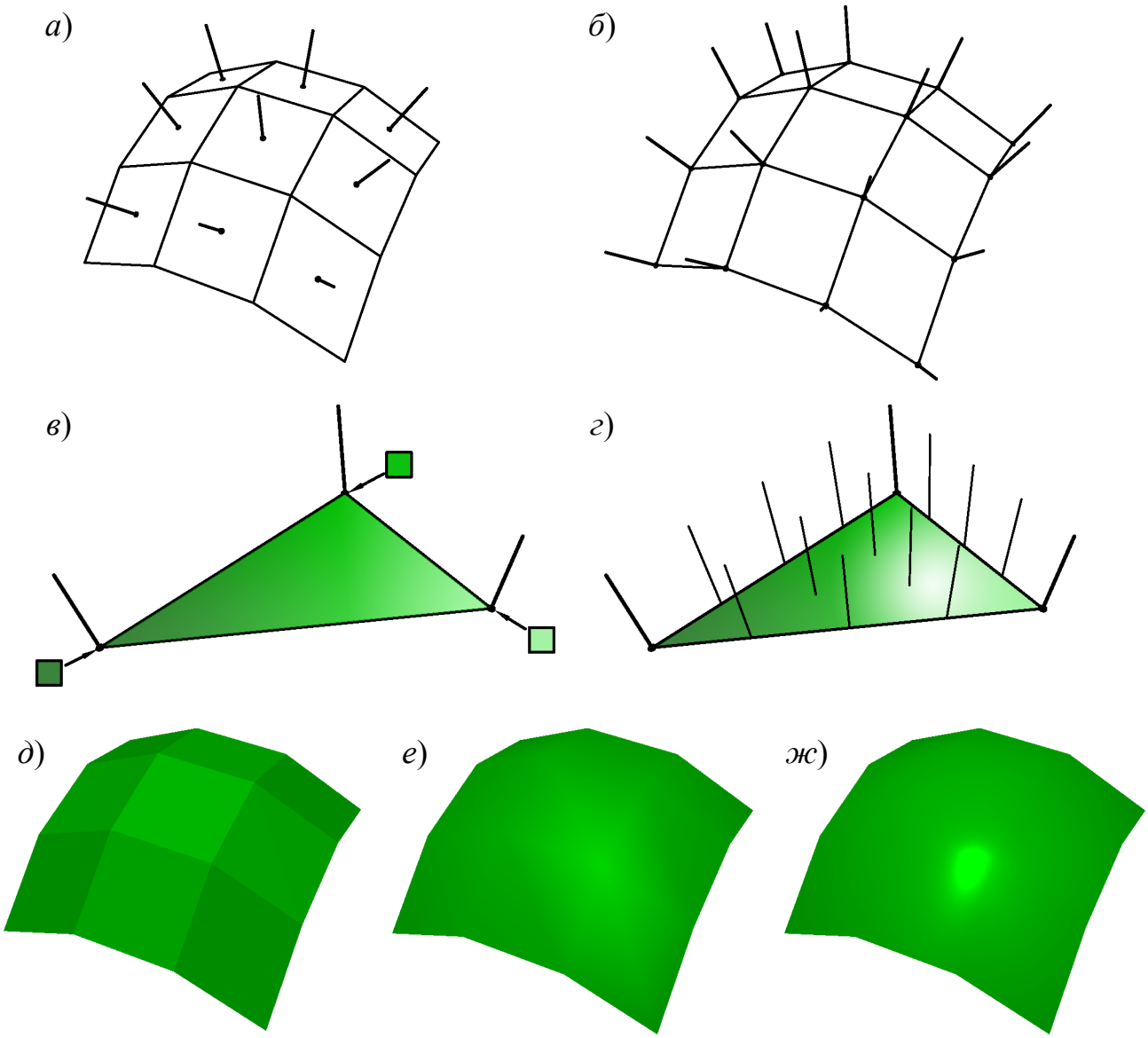


Рисунок 7.18 – Закрашивание полигональных моделей: нормали к граням (*а*) и усредненные нормали в вершинах (*б*); закрашивание полигона методом Гуро (*в*) и методом Фонга (*г*); поверхность, закрашенная по нормалям к граням (*д*), по методам Гуро (*е*) и Фонга (*ж*)

В результате грань заполняется равномерным цветовым переходом и кажется выпуклой. Цветовой переход вдоль ребра грани определяется только цветами двух вершин, которые соединяет это ребро, и не зависит от цветов других вершин. Поэтому у двух соседних граней цвета вдоль общего ребра будут одинаковыми — граница между гранями становится невидимой. Создается иллюзия гладкой криволинейной поверхности, рис. 7.18 *е* и 7.13 *г*.

Закрашивание по методу Гуро не требует большого числа вычислений и выполняется с помощью простого алгоритма заполнения полигона, в который добавлена процедура линейной интерполяции цвета. Поэтому метод весьма быстродействующий и используется при создании движущихся изображений в режиме реального времени. Метод Гуро поддерживается на аппаратном уровне современными графическими системами.

Вместе с тем, поскольку моделирование процесса отражения света выполняется только для некоторых точек поверхности (вершин), метод Гуро дает достаточно реалистичное изображение только для матовых поверхностей и не позволяет получить блики или зеркальное отражение.

Для создания более реалистических изображений поверхностей, в т.ч. блестящих, зеркальных, прозрачных, используется ***метод Фонга***. По методу Фонга также вычисляются нормали к граням и «нормали» в вершинах как усреднение нормалей соседних граней. Но вычисление цвета (согласно модели отражения, интенсивности и направленности освещения) выполняется *отдельно для каждого пиксела* изображения грани. При этом выполняется *линейная интерполяция векторов нормалей* — для точки поверхности грани, проецируемой в данный пиксел изображения, рассчитывается усредненный вектор «нормали» пропорционально расстоянию до векторов «нормалей» в вершинах. Поэтому грань закрашивается так же, как если бы она была действительно выпуклой или вогнутой, рис. 7.18 *г*.

Метод Фонга требует значительно большего количества вычислений и применяется при окончательной отрисовке объектов, позволяя создать качественное реалистичное изображение, рис. 7.18 *ж* (см. также рис. 7.13 *д*).

Следует отметить, что эффект «сглаживания» граней как методом Гуро, так и Фонга проявляется только внутри контура поверхности. Крайние, контурные грани объекта в любом случае будут заметны — на рис. 7.18 *д*, *е* и *ж* контур поверхности состоит из ломаных линий. Единственный способ сделать контур объекта более плавным — увеличить количество полигонов при аппроксимации поверхности.

7.2 Имитация мелких деталей и микрорельефа

В некоторых областях применения компьютерной графики (например, в САПР) вполне достаточно визуализировать 3D-модели поверхностей закрашиванием их заданным цветом по методу Гуро, поскольку необходимо лишь иметь представление о форме и внешнем виде объекта проектирования. В таких же областях, как синтез изображений художественного характера, создание спецэффектов в фильмах, компьютерных мультфильмов и рекламных роликов, необходимо создавать изображение, максимально близкое к реальному. В этом случае необходимо решить задачи *трассировки световых лучей* и *имитации мелких деталей и микрорельефа*.

Реальные объекты редко имеют идеально гладкую одноцветную поверхность. Исключение составляют лишь объекты искусственного происхождения (например, кузов автомобиля, белая тарелка и т.п.). Большинство же объектов как природного, так и искусственного происхождения (ствол дерева, гора, кожа человека, стена дома, дорога) имеют поверхности чрезвычайно сложного строение из-за множества *мелких деталей* — неровностей, областей различного цвета и т. п. Если попытаться передать все мелкие детали с помощью полигональной модели, такая модель окажется слишком сложной и объемной. Однако мелкие детали можно имитировать с помощью *текстур* и *карт микрорельефа*.

Текстурой называется *растровое изображение*, которое используется для заполнения поверхности трехмерного объекта. В 3D-графике обычно используются ***проективные текстуры*** — растры, которые накладываются (проецируются) на поверхность объекта. Элемент растра текстуры называется ***текселом***.

Текстура накладывается на поверхность следующим образом. *Расположение растра текстуры* задается углами поворота ее системы координат ($x_т$, $y_т$, $z_т$,) относительно системы координат объекта (x, y, z,), рис. 7.19 *а*. Также указывается размер растра (в единицах размеров объекта) и его смещение. В большинстве программных пакетов для удобства наложения текстуры ее размер и смещение указываются в долях габарита объекта. Если растр текстуры меньше размера объекта, возможно *копирование* текстуры по всей его поверхности. Так, на рис. 7.13 *д* рисунок пола задан растром для одной плитки — остальные получены копированием текстуры.

Каждый *тексел* трактуется как прямоугольник, закрашенный некоторым цветом (прямоугольник 1 2 3 4 на рис. 7.19 *а*). Наложение текстуры выполняется проецированием областей текселов на поверхность. В результате на поверх-

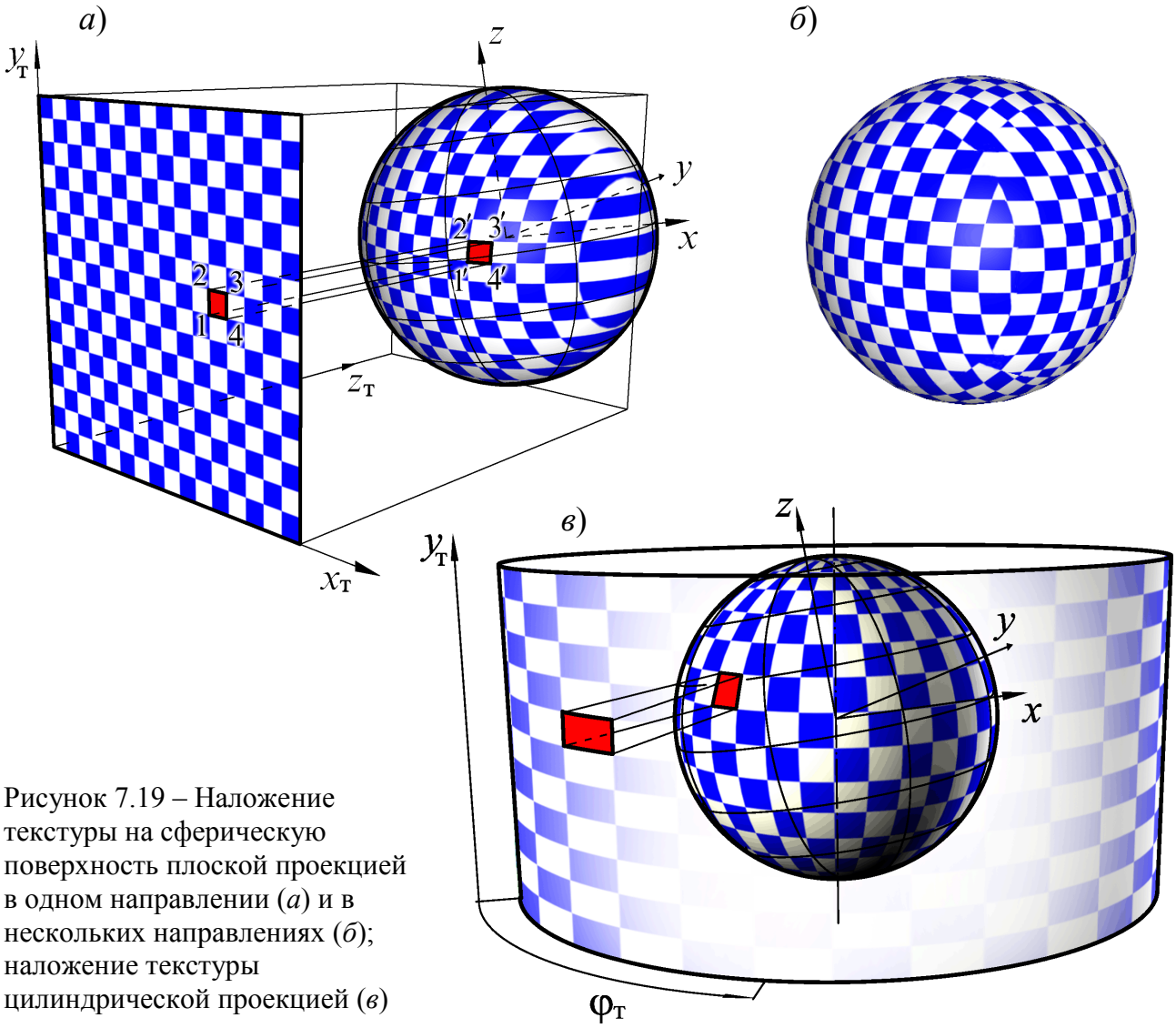


Рисунок 7.19 – Наложение текстуры на сферическую поверхность плоской проекцией в одном направлении (*а*) и в нескольких направлениях (*б*); наложение текстуры цилиндрической проекцией (*в*)

ности образуется некоторая фигура $1'2'3'4'$. При проецировании этого участка поверхности на плоскость экрана ему присваивается цвет данного тексела.

Как видно из рис. 7.19 *а*, при больших углах наклона поверхности относительно плоскости текстуры форма проекций текселов сильно искажается. Существует два способа текстурирования поверхностей с большими углами охвата. Первый — наложение *нескольких* одинаковых или разных текстур, каждая из которых расположена под небольшими углами к той части поверхности объекта, на которую проецируется. На рис. 7.19 *б* различные участки сферы заполнены шестью одинаковыми текстурами, расположенными во взаимно перпендикулярных плоскостях. Видно, что в местах стыковки текстур имеются дефекты.

Второй способ — наложение текстуры с использованием *цилиндрической* или *сферической* проекций. При ***цилиндрической проекции*** текстуры одна из координат растра трактуется как угол поворота $\varphi_т$, рис. 7.19 *в*. Проецирование выполняется лучами, пересекающими ось цилиндра и перпендикулярными этой оси. ***Сферическая проекция*** отличается тем, что обе координаты растра рассматриваются как угловые, а проецирующие лучи проходят через центр сферы. Цилиндрическая и сферическая проекции позволяют закрасить всю поверхность объекта.

При недостаточно большом разрешении растра, а также при рассмотрении всего объекта или некоторой его части со слишком близкого расстояния становятся заметны отдельные текселы текстуры, рис. 7.20 *а*. Для улучшения вида текстурированной поверхности используются методы ***фильтрации*** (сглаживания) растров текстур, например *билинейную фильтрацию*, рис. 7.20 *б*. Изображение становится слегка размытым, но отдельные текселы не заметны.

Применение текстуры для закрашивания поверхности не исключает задания для нее других оптических свойств — зеркального отражения, прозрачности и др. При этом текстура обычно применяется с диффузной моделью отражения; цветовые эффекты от текстуры и других оптических моделей складываются согласно (7.26).

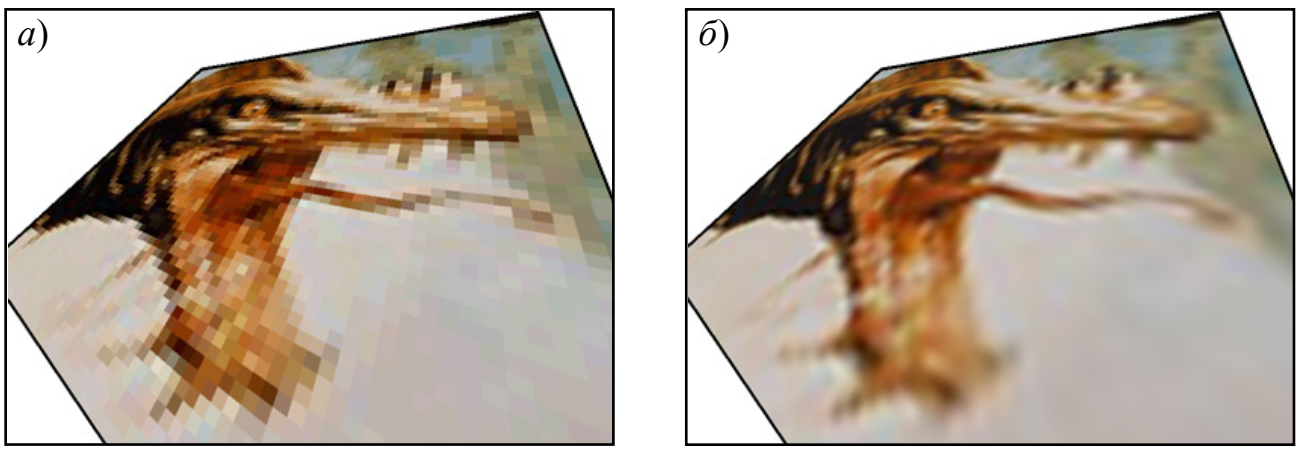


Рисунок 7.20 – Наложение текстуры без фильтрации (*а*) и с билинейной фильтрацией (*б*)

Карты текстур применяются не только для окрашивания поверхности, но и для моделирования других ее свойств, например ***микрорельефа*** — мелких неровностей на поверхности. Моделирование таких неровностей с помощью полигональной сетки требует значительного усложнения модели. Использование *карт микрорельефа* позволяет решить эту проблему с гораздо меньшими затратами времени.

Карта микрорельефа (*bump map*) — это растр, позволяющий имитировать микрорельеф. В качестве карт микрорельефа используются полутоновые или полноцветные изображения (в последнем случае используется только интенсивность, а цвет игнорируется).

Микрорельеф имитируется следующим образом. Интенсивность текселов растра микрорельефа трактуется как *отклонение поверхности* микрорельефа в данной точке от поверхности грани, рис. 7.21. Для каждой точки поверхности вычисляется вектор «нормали», отклоненный от действительной нормали грани в соответствии с картой микрорельефа. Дальнейший расчет параметров отражения и преломления света данным участком поверхности выполняется с использованием отклоненного вектора — в результате создается иллюзия «наклона» участка поверхности. На поверхности появляются «выступы» и «впадины» — микрорельеф, рис. 7.22 *а*.

Использование микрорельефа позволяет создавать эффекты неровной зеркальной или прозрачной поверхности, например, поверхность плитки пола и абажура лампы на рис. 7.13 *д*, волн на рис. 7.22 *б*. Следует заметить, что это не реальный рельеф, а лишь его иллюзия, созданная цветовым рисунком на грани — в действительности грань остается плоской.

Текстурные карты могут использоваться также для задания других свойств поверхности: прозрачности, зеркальности, шероховатости и др. Например, на рис. 7.23 карта текстуры задает прозрачность поверхности.

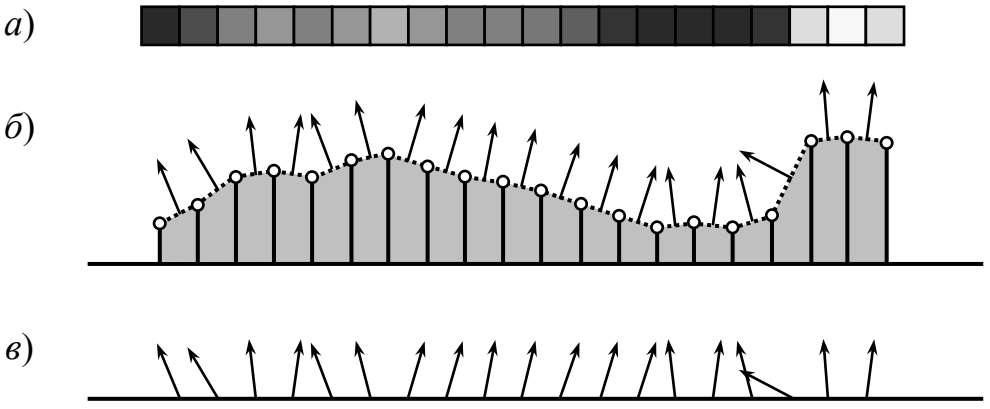
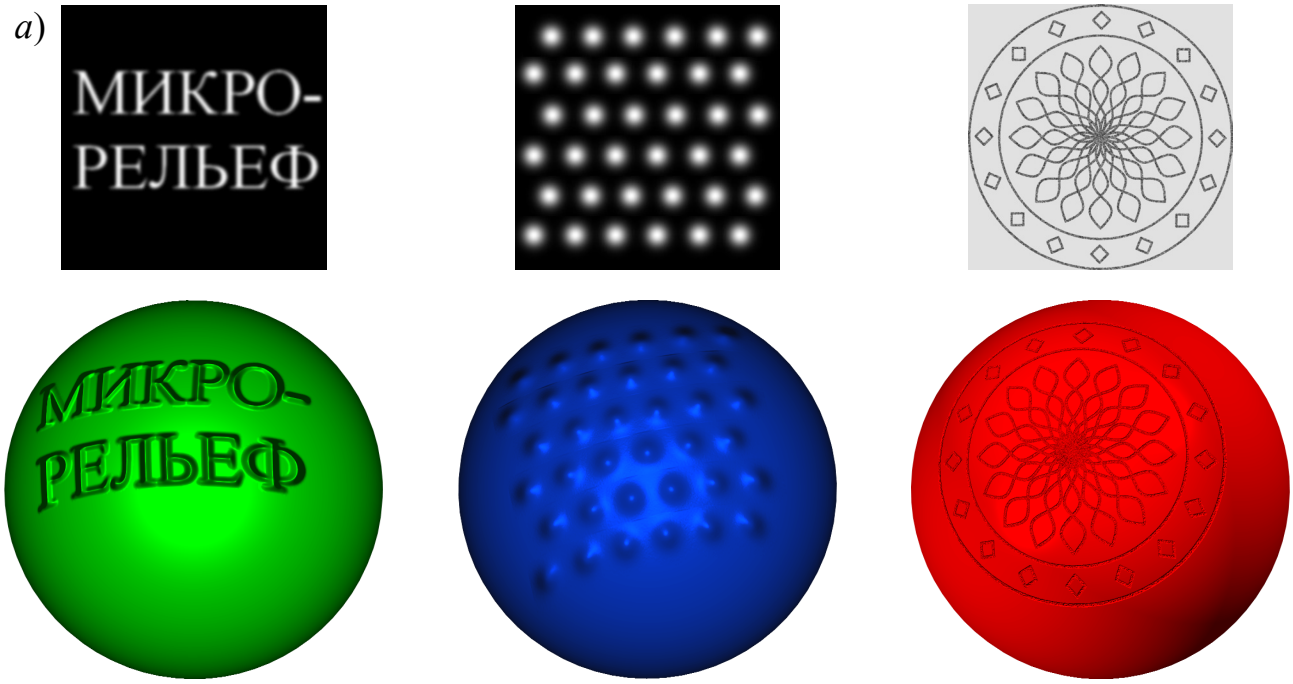


Рисунок 7.21 – Имитация микрорельефа: строка растра (*а*), интерпретация интенсивностей текселов как высот микрорельефа (*б*), отклонение векторов нормалей на поверхности (*в*)

а)



б)

Рисунок 7.22– Наложение карт рельефа на шар (*а*) и на плоскую зеркальную поверхность (*б*)

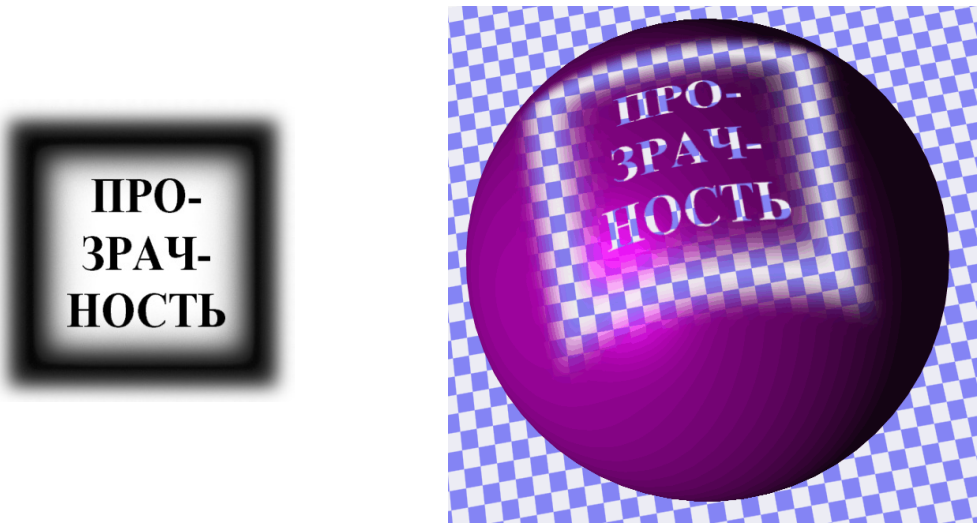


Рисунок 7.23 – Карта прозрачности

7.3 Освещение и тени

Важную роль в создании реалистических изображений играет имитация реальных условий ***освещения*** светоотражающих объектов. При создании упрощенных изображений в процессе разработки модели обычно используют источник света, направленный на объект от зрителя (*фронтальное освещение*), рис. 7.24 *а*, что позволяет подсветить все элементы объекта.

При построении более сложных сцен используют различные ***модели источников света***. Получили распространение такие модели:

1. ***Удаленный источник***, рис. 7.24 *б*, имитирует источник света, расположенный настолько далеко, что идущие от него лучи параллельны. Таким свойством обладает свет, идущий от небесных светил — Солнца, Луны. Интенсивность света с расстоянием не меняется. Источник задается *вектором*, определяющим направление лучей света. В некоторых программных пакетах предусмотрена специальная модель *солнечного освещения*, являющаяся разновидностью удаленного источника и позволяющая вычислить направление вектора освещения в зависимости от времени суток, широты и долготы местности.

2. ***Точечный источник***, рис. 7.24 *в*, имитирует в основном искусственные источники света (электролампу, свечу) и задается расположением в пространстве одной *точки*. Световые лучи выходят из этой точки во всех направлениях и освещают окружающее пространство. С удалением от точечного источника *интенсивность света уменьшается*. Для моделирования этого эффекта используются обратная линейная или обратная квадратичная зависимости:

$$I = \frac{I_0}{L}; \qquad I = \frac{I_0}{L^2}, \tag{7.27}$$

где I_0 – интенсивность источника; L – расстояние от источника до объекта.

Обратная квадратичная зависимость соответствует действительному оптическому закону уменьшения интенсивности света, но часто не дает нужного визуального эффекта, поскольку способность графических устройств передавать различные уровни яркости значительно уступает человеческому зрению.

3. ***Направленный источник***, рис. 7.24 *г*, является разновидностью точечного, но его световой поток ограничен бесконечным конусом с вершиной в точке расположения источника. Направленный источник задается в пространстве расположением его *точки*, а также *вектором*, определяющим направление оси конуса и *углом* конуса. Для моделирования уменьшения интенсивности используются законы (7.27).

Кроме расположения и направления, источники света характеризуются ***интенсивностью*** и ***цветом*** — свет может быть *ахроматическим* (белым) и *хроматическим* (окрашенным в некоторый цвет). В последнем случае

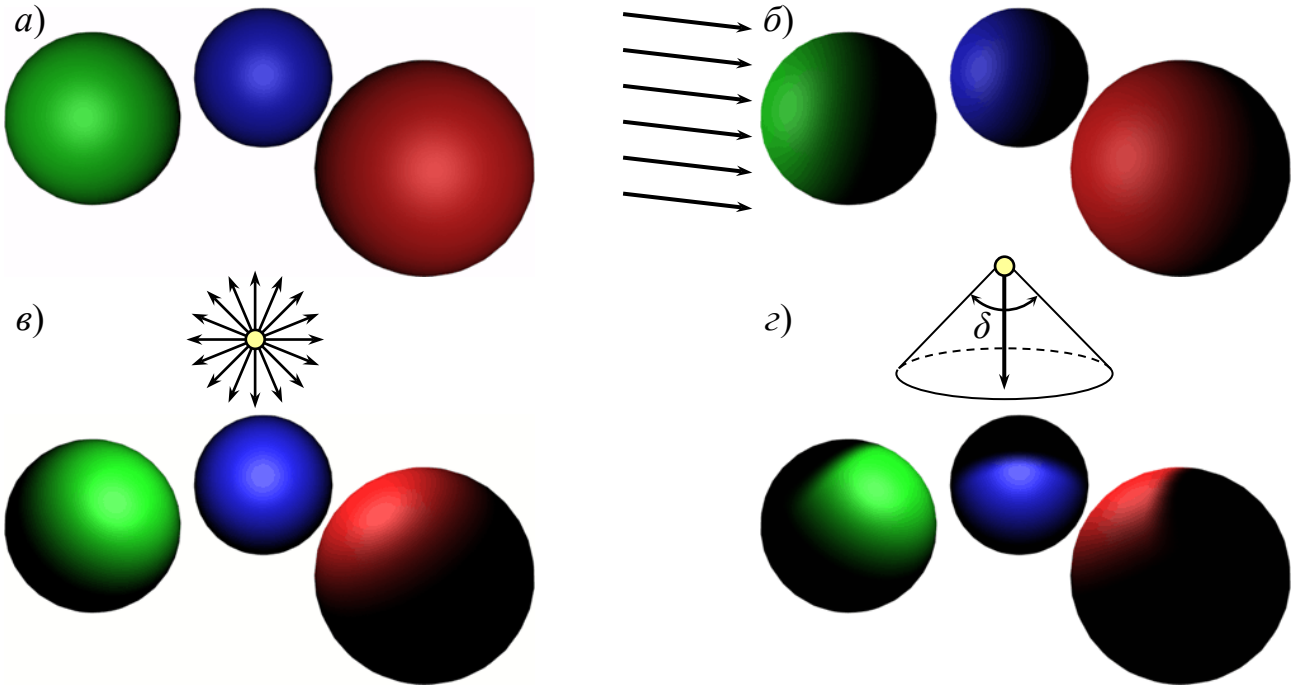


Рисунок 7.24 – Модели освещения объектов: фронтальное (*а*), удаленный источник (*б*); точечный источник (*в*) и направленный источник (*г*)

изображение поверхности не будет соответствовать по цвету характеристике самого объекта — объект будет окрашиваться.

Реальные объекты при освещении их некоторым источником света отбрасывают ***тень*** — экранируют другие объекты, расположенные дальше от источника, от попадания на них прямых световых лучей. Имитация явления *тени* — весьма сложная задача, решение которой необходимо при создании действительно реалистических изображений.

Тени, отбрасываемые объектами, образуют на поверхности других объектов весьма сложные фигуры, которые определяются формой обеих поверхностей и их взаимным расположением, рис. 7.25. Задачу нахождения теней можно рассматривать как вариант задачи нахождения невидимых частей объектов, если рассматривать их из точки расположения источника света. Получили распространение три метода моделирования теней.

Первый метод предполагает создание до визуализации сцены ***карт теней*** — двухмерных массивов, задающих расстояние от источника света до ближайшей к нему поверхности. Этот метод несколько напоминает метод Z-буфера, но при проецировании не на плоскую, а на сферическую поверхность с центром в точке расположения источника света. Область пространства, через которую проходит световой поток, разбивается на элементы в виде бесконечных конусов с квадратным поперечным сечением, каждый из которых рассматривается как отдельный «луч», рис. 7.25 *а*. Весь световой поток представляет собой совокупность таких «лучей» — своеобразный растр. Для каждого «луча» методом перебора всех объектов сцены определяются те объекты, которые он пересекает. Из этих объектов луч освещает только тот,

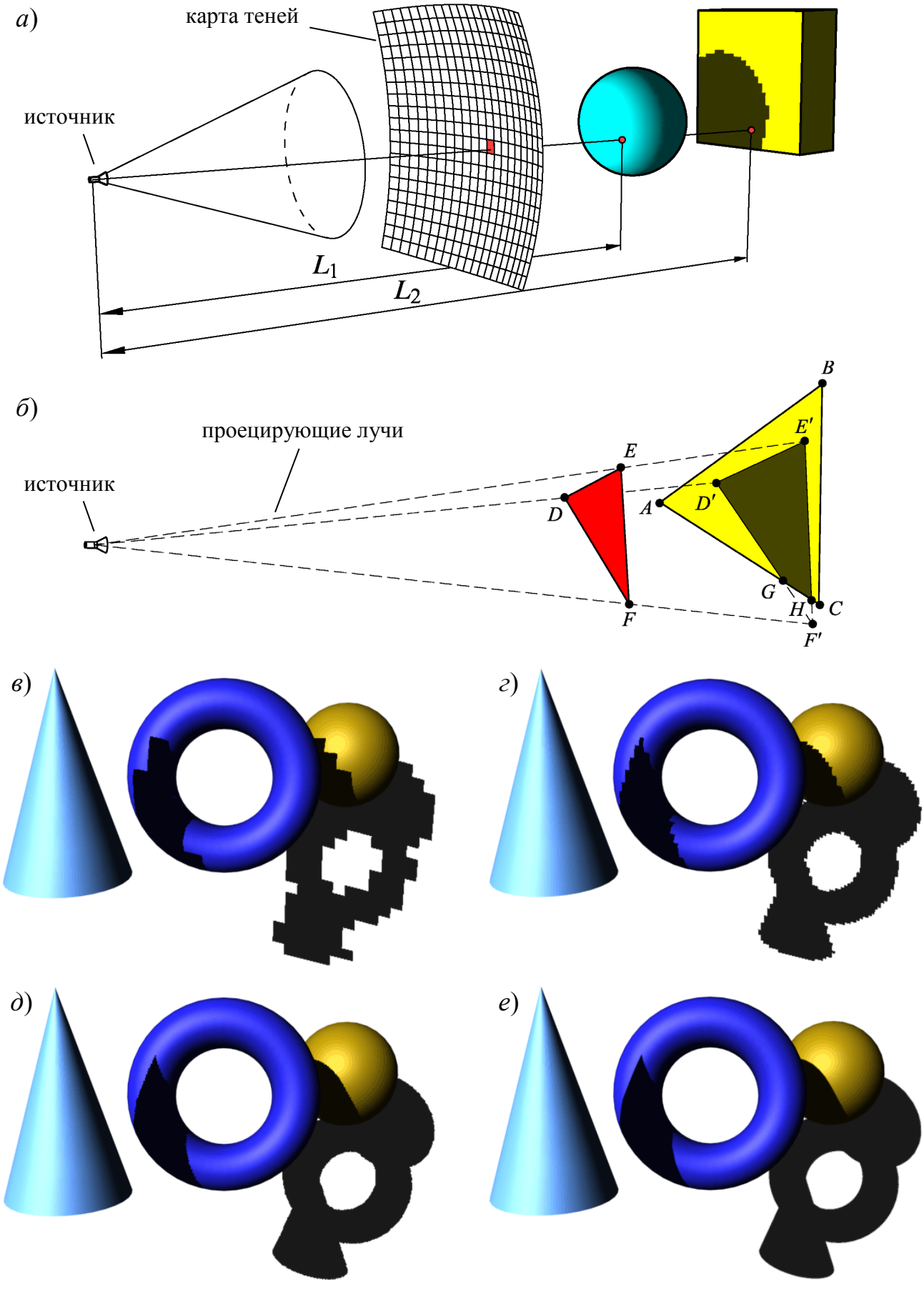


Рисунок 7.25 – Создание теней при помощи карты теней (*а*) и методом проецирования (*б*); тени, созданные при размере карты 32×32 (*в*), 128×128 (*г*), 1024×1024 (*д*) и трассировкой лучей (*е*)

который расположен ближе всего к источнику света. Номер этого объекта записывается в массив. При отрисовке других объектов, расположенных вдоль данного луча дальше от источника, этот луч не учитывается.

Для каждого источника света создается своя карта теней, поэтому метод требует весьма значительных объемов памяти и затрат процессорного времени на создание карт (особенно для сложных сцен). Тени на рис. 7.25 *в* созданы с помощью карты размером 32×32 — при этом отчетливо видна дискретная структура карты — каждая ячейка дает тень прямоугольной формы. На рис. 7.25 *д* использована карта размером 1024×1024 ячейки.

Второй метод — определение теней ***расчетным методом*** непосредственно в процессе визуализации. Для каждого полигона поверхности выполняется анализ освещенности путем перебора всех остальных объектов сцены. Если некий другой полигон находится на пути светового луча от источника к данному полигону, на последнем создается тень — проекция экранирующего полигона на поверхность изображаемого. На рис. 7.25 *б* изображены два полигона. Полигон *DEF* расположен ближе к источнику света, чем полигон *ABC* и проецируется на плоскость последнего лучами, исходящими из источника света, в виде фигуры *D'E'F'*. Поскольку не вся фигура *D'E'F'* расположена внутри полигона *ABC*, необходимо найти координаты точек пересечения ребер фигуры *D'E'F'* с ребрами полигона *ABC* (точки *G* и *H*). Получаем фигуру *D'E'HG*, которая оконтуривает тень на грани *ABC*. Метод позволяет получить более качественный, «гладкий» рисунок тени, рис. 7.25 *в*, но требует выполнения сложных расчетов и в некоторых случаях привести к ошибкам в вычислении — неправильной отрисовке тени..

Третий метод — построение тени ***трассировкой световых лучей*** — обычно применяется при построении всего изображения методом *обратной трассировки* (см. ниже). Для каждой точки поверхности выполняется расчет трассы луча, идущего от источника. Если луч на своем пути пересекает другие поверхности, он считается экранированным. Метод не требует больших объемов памяти, позволяет получить наиболее качественный рисунок тени, рис. 7.25 *е*, дает возможность получать тени от полупрозрачных объектов, но требует выполнения большого объема вычислений.

7.4 Метод трассировки лучей

Изображения трехмерных сцен с относительно низким уровнем реалистичности строятся методом *прямого расчета* — последовательно отрисовываются все объекты и их элементы. При этом могут учитываться освещенность объектов, их взаимное перекрытие и тени. Однако такой метод не позволяет моделировать сложные оптические эффекты, такие как *зеркальное отражение* одних объектов в других, *преломление* световых лучей при прохождении через прозрачные объекты и т.п. Поэтому для создания качественных реалистичных изображений используются методы *трассировки лучей*.

Трассировка лучей — это метод получения изображения, заключающийся в *моделировании распространения лучей света*, их отражения от объектов и преломления в них.

Основным положением трассировки лучей есть то, что луч света в свободном пространстве распространяется *по прямой* до тех пор, пока не встретит препятствие — поверхность некоторого объекта. Каждый источник света испускает бесчисленное множество лучей. Такие лучи называют *первичными*. Часть из них уходит в пространство, часть — попадает на поверхности объектов. В зависимости от своих оптических свойств эти поверхности отражают и преломляют свет, порождая бесчисленное множество *вторичных лучей*. Вторичные лучи также либо уходят в пространство, либо попадают на поверхности других объектов, вновь отражаясь и преломляясь — порождая новые вторичные лучи и т.д. Какая-то (обычно очень малая) часть лучей попадает в зрачок глаза или объектив камеры и проецируется на поверхность экрана (сетчатку глаза, фотопленку или чувствительную матрицу). Так создаются реальные изображения.

Известны два метода моделирования распространения света — *прямая* и *обратная трассировки*.

Прямая трассировка лучей предполагает отслеживание движения луча от источника света к поверхностям объектов и далее — на плоскость экрана. Такой метод позволил бы наиболее точно воспроизвести реальные оптические явления, но его реализация на ЭВМ представляет серьезные затруднения — необходимо отслеживать все первичные и вторичные лучи (но и тех, и других *бесконечно много*). Кроме того, в создании изображения участвуют лишь лучи, пересекающие плоскость экрана и проходящие через точку фокуса — найти такие лучи в общей их массе весьма сложно. Все же остальные лучи будут рассчитаны зря. Значительно проще реализовать алгоритм *обратной трассировки лучей*.

Обратная трассировка лучей заключается в отслеживании световых лучей в обратном направлении — от экрана к объектам, и далее — к другим объектам и источникам света. В таком случае изначально трассируются только «нужные» лучи — проходящие через точку фокуса и пересекающие плоскость экрана. Обычно трассируют лучи, проходящие через центр пикселов растра генерируемого изображения, рис. 7.26, — результатом трассирования будет *цвет* данного пиксела.

Метод трассирования предполагает поиск *точек пересечения* луча с поверхностями объектов с использованием переборных алгоритмов. Ближайшая из найденных точек пересечения используется для дальнейших расчетов.

Точка объекта, с которой пересекается луч трассирования, в свою очередь освещается *бесконечным* числом световых лучей, идущих от источников света и других объектов. Поэтому для продолжения трассирования вводятся следующие ограничения:

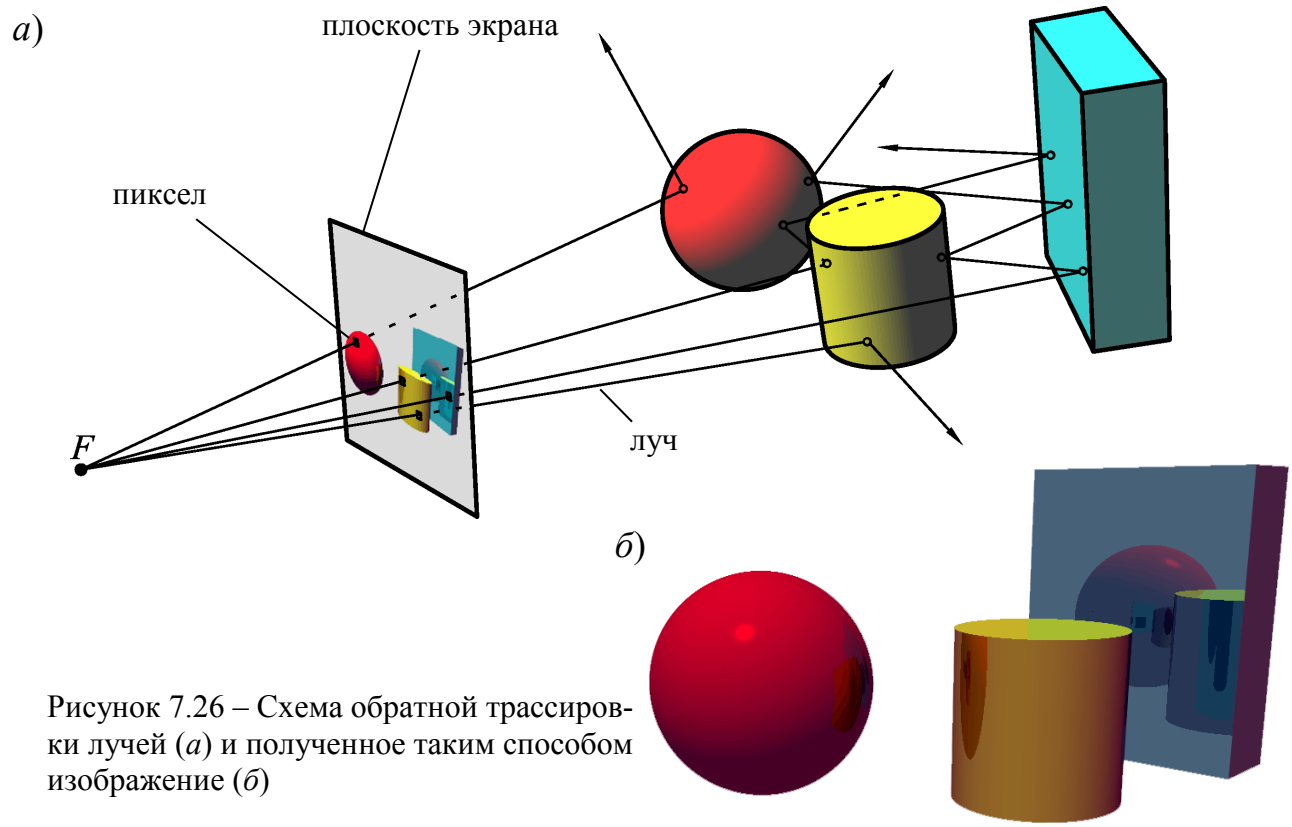


Рисунок 7.26 – Схема обратной трассировки лучей (*а*) и полученное таким способом изображение (*б*)

1. Источники света могут быть либо *точечными*, либо *бесконечно удаленными* — они не могут иметь размер и форму. Для имитации распределенных источников света (световых панелей, абажуров ламп, открытого огня) используются специальные методы.

2. Свойства отражающих поверхностей описываются сумой пяти компонентов — *диффузного* отражения, *идеального зеркального* отражения, *идеального преломления*, *зеркального* отражения по модели *Фонга* и освещения рассеянным светом, причем трассировка лучей выполняется только для моделей идеального отражения и преломления.

Из точки пересечения исходного луча с поверхностью трассируется только один *идеально отраженный* луч, или два — отраженный и преломленный — для полупрозрачных объектов, рис. 7.27 (в действительности этот луч — падающий, а отражается исходный для расчета, но в соответствии с законами отражения и преломления траектория светового луча при изменении направления его движения не меняется, и его условно можно назвать «отраженным»). Трассирование этих лучей позволяет получить результат идеального отражения или преломления света от других *отражающих объектов* (но не источников света).

Кроме того, для рассматриваемой точки определяется направление на источники света и в соответствии с диффузной моделью и моделью Фонга вычисляются интенсивности отражения. Диффузная модель позволяет получить равномерную освещенность поверхности, а модель Фонга — *блики* от источников света. Полученные по всем перечисленным моделям яркости и

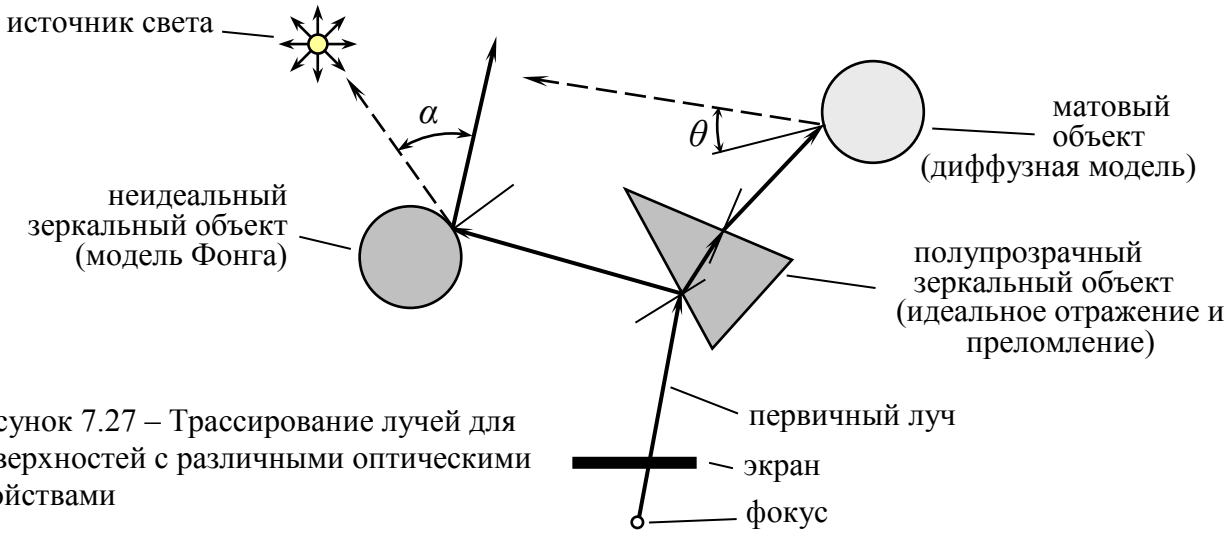


Рисунок 7.27 – Трассирование лучей для поверхностей с различными оптическими свойствами

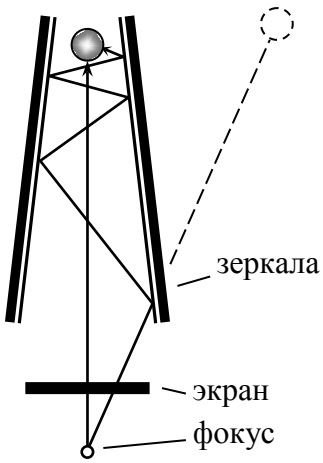
оттенки складываются и дают результат — цвет и яркость данной точки поверхности.

При трассировании отраженного луча он может снова попасть на зеркальную поверхность, что вызовет необходимость снова его трассировать. Возможны случаи, когда луч будет отражаться большое (и даже бесконечно большое) число раз, рис. 7.28. Чтобы не увеличивать чрезмерно время выполнения трассирования, обычно вводят ограничение на глубину трассировки — один и тот же луч может отражаться не более n раз.

Рассмотрим преимущества и недостатки метода обратной трассировки световых лучей. Безусловным преимуществом метода является создание *наиболее реалистичных изображений*. Ни один из известных ныне методов визуализации 3D-сцен не позволяет столь же точно учесть законы оптики, воспроизвести сложные оптические эффекты, рис. 7.29 (см. также рис. 7.13 *д* и 7.29 *б*). Другое преимущество метода — *автоматическое удаление невидимых элементов* — не требуется использовать Z-буфер или какой-либо другой метод.

К недостаткам метода следует отнести следующее:

1. Метод не позволяет учесть освещение матовых объектов и объектов – неидеальных зеркал светом, отраженным от других объектов, поскольку при расчете освещенности по диффузной модели и модели Фонга учитываются



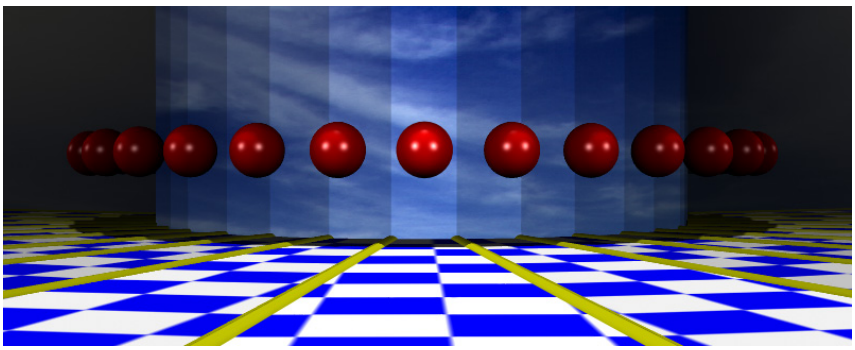


Рисунок 7.28 – Эффект многократного отражения

только источники света. Создание такого простого эффекта, как «солнечный зайчик» (освещение участка объекта светом, отраженным от зеркальной поверхности другого объекта) при помощи стандартных алгоритмов трассировки лучей невозможно. Такие эффекты имитируют, вводя дополнительные источники света.

2. Метод трассировки лучей требует выполнения сложных вычислений для каждого пиксела создаваемого изображения, поэтому это — наиболее медленный из всех методов визуализации.

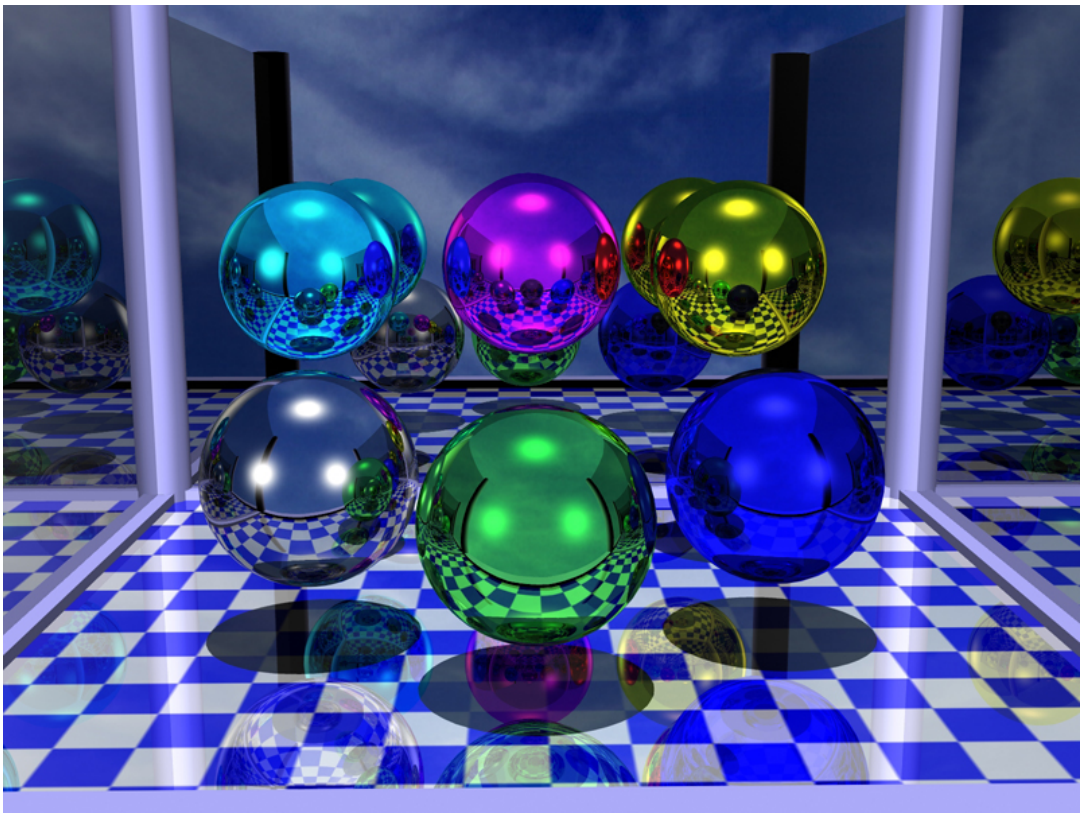


Рисунок 7.29 – Пример изображения, полученного методом обратной трассировки лучей

СПИСОК ЛИТЕРАТУРЫ

1. Ашкенази Г. И. Цвет в природе и технике. – М.: Энергоатомиздат, 9185.
2. Блинова Т. А., Порев В. Н. Компьютерная графика. – К. : Юниор, 2006.
3. Иванов В. П., Батраков А. С. Трехмерная компьютерная графика. – М: Радио и связь, 1995.
4. Павлидис Т. Алгоритмы машинной графики и обработки изображений. – М.: Радио и связь, 1986.
5. Порев В. Н. Компьютерная графика. – СПб.: БХВ-Петербург, 2002.
6. Роджерс Д. Алгоритмические основы машинной графики: Пер. с англ. – М: Мир, 1989.
7. Шикин Е. В., Боресков А. В. Компьютерная графика. – М.: «Диалог-МИФИ», 1995.
8. Шикин Е. В., Боресков А. В. Компьютерная графика. Полигональные модели – М.: «Диалог-МИФИ», 2000.

Компьютерное обеспечение инженерной деятельности
в энергомеханической сфере. Курс лекций
(для студентов направления подготовки 15.04.02 и специальности 21.05.04)

Составитель: Олег Васильевич Федоров, к.т.н., доц.