ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ИНДУСТРИАЛЬНЫЙ УНИВЕРСИТЕТ

ИНСТИТУТ ДИСТАНЦИОННОГО ОБРАЗОВАНИЯ

Б.В.Кириличев

ГРАФИЧЕСКИЙ ИНТЕРФЕЙС ОПЕРАТОРА

Учебное пособие

для студентов специальности 220301

Москва 2011

Содержание

Введение

Графический интерфейс пользователя (ГИП, англ. *graphical user interface*, *GUI)* в вычислительной технике – система средств для взаимодействия пользователя с компьютером, основанная на представлении всех доступных пользователю системных объектов и функций в виде графических компонентов экрана (окон, значков, меню, кнопок, списков и т. п.). При этом в отличие от интерфейса командной строки, пользователь имеет произвольный доступ (с помощью клавиатуры или устройства координатного ввода типа «мышь») ко всем видимым экранным объектам.

Впервые концепция ГИП была предложена учеными из исследовательской лаборатории Xerox PARC в 1970-х, но получила коммерческое воплощение лишь в продуктах корпорации Apple Computer. В операционной системе AmigaOS ГИП с многозадачностью был использован в 1985 г. В настоящее время ГИП является стандартной составляющей большинства доступных на рынке операционных систем и приложений. Примеры систем, использующих ГИП: Mac OS, Solaris, GNU/Linux, Microsoft Windows, NEXTSTEP, OS/2, BeOS.

Если рассматривать вопрос шире, то следует определить, что в современном мире вкладывается в понятие интерфейса пользователя (не только графического).

Интерфейс пользователя (*UI* – англ. *user interface*) – совокупность средств, при помощи которых пользователь общается с различными устройствами, чаще всего – с компьютером или бытовой техникой, либо иным сложным инструментарием (системой). Это определение, как видим, несколько более широкое.

Интерфейс пользователя компьютерного приложения включает:

1) средства отображения информации, саму информацию, форматы и коды;
2) командные режимы, язык «пользователь – интерфейс»;
3) устройства и технологии ввода данных;
4) диалоги, взаимодействие и транзакции между пользователем и компьютером, обратную связь с пользователем;
5) поддержку принятия решений в конкретной предметной области;
6) порядок использования программы и документацию на неё.

Пользовательский интерфейс часто понимают только как внешний вид программы. Однако на деле пользователь воспринимает через него всю программу в целом, а значит, такое понимание является слишком узким.

В действительности пользовательский интерфейс объединяет в себе все элементы и компоненты программы, которые способны оказывать влияние на взаимодействие пользователя с программным обеспечением (ПО). Это не только экран, который видит пользователь. К этим элементам относятся:

1) набор задач пользователя, которые он решает при помощи системы;
2) используемая системой метафора (например, рабочий стол в MS Windows);
3) элементы управления системой;
4) навигация между блоками системы;

5) визуальный (и не только) дизайн экранов программы;
6) средства отображения информации, отображаемая информация и форматы;
7) устройства и технологии ввода данных;
8) диалоги, взаимодействие и транзакции между пользователем и компьютером;
9) обратная связь с пользователем;
10) поддержка принятия решений в конкретной предметной области;
11) порядок использования программы и документация на нее.

Для упрощения восприятия функции программы пользователем при разработке пользовательского интерфейса желательно использовать метафоры.

Графический интерфейс оператора (ГИО) – это совокупность средств, при помощи которых оператор (человек) общается с технической частью человеко-машинной системы в процессе управления ею.

Главное отличие данного определения от приведенных выше состоит, прежде всего, в том, что человек выступает в роли не просто абстрактного пользователя, решающего какие-то свои задачи. Он выступает в роли оператора, осуществляющего управление сложной человеко-машинной системой, в состав которой он и сам входит. Еще одно отличие заключается в возможном расширении арсенала устройств отображения графической информации: это может быть не только экран компьютерного монитора или плазменная панель, копирующая изображение с экрана, но и специальные табло, коналоги[1], мнемосхемы[2], и т.п. Другое дело, что по причине универсальных возможностей компьютерного представления любых графических изображений на экране можно воспроизвести вид как всех перечисленных, так и любых других, даже гипотетических устройств отображения. Причем уже сейчас великолепное качество компьютерных изображений продолжает совершенствоваться, развиваются средства виртуальной реальности, трехмерные изображения обретают реальность.

---

[1] Коналог (контактный аналог) – устройство отображения информации, дающее целостное изображение состояния системы, объекта управления или внешней среды. Различают горизонтальные и вертикальные коналоги. В первом случае на индикаторе отображается как бы вид сверху на объект (например, с самолета). Горизонтальные коналоги применяются преимущественно в авиации для решения штурманских задач. На вертикальных коналогах обстановка изображается во фронтальной плоскости (например, вид с наблюдательного пункта подводной лодки или из кабины самолета при посадке и взлете). Вертикальные коналоги применяются для управления скоростью, направлением, пространственным положением и другими параметрами движущегося объекта. Отображение информации на коналоге может производиться по принципам внешнего наблюдения, когда оператор смотрит на управляемый объект как бы со стороны, и внутреннего – оператор как бы движется вместе с объектом относительно фиксированного ориентира. Различие между этими принципами незначительно и проявляется обычно лишь в начале тренировки. В авиации находят применение индикаторы, объединяющие оба принципа отображения. Отличительное свойство коналога – координатная система, в которой одновременно представлена информация о нескольких параметрах объекта. Преимущества коналогов по сравнению с обычными (стрелочными) приборами: 1) один коналог заменяет несколько отдельных индикаторов по каждому параметру; 2) используются те же особенности восприятия человека, которые сформировались у него в естественных условиях.

[2] Мнемосхема (мнемоническая схема) – графическая модель, отображающая динамически изменяющуюся функциональную (техническую) схему управляемого оператором объекта. Может быть реализована в виде разных типов дисплеев, приборов. Это совокупность сигнальных устройств и сигнальных изображений оборудования и внутренних связей контролируемого объекта, размещаемых на диспетчерских пультах, операторских панелях или выполненных на персональном компьютере. Информация, которая выводится на мнемосхему, может быть представлена в виде аналогового, дискретного и релейного сигнала, а также графически. На мнемосхемах отражается основное оборудование, сигналы, состояние регулирующих органов. Вспомогательный и справочный материал должен быть расположен в дополнительных формах отображения, с возможностями максимально быстрого извлечения этих вспомогательных форм на экран.

Глава 1. Системы управления с человеком

В дальнейшем мы будем рассматривать такие системы, которые наряду с техническими устройствами, в том числе и с вычислительной техникой и программным обеспечением, включают также человека или группу людей, участвующих в процессе управления этой системой. Таким образом, мы подчеркиваем активную роль человека в управлении: именно он принимает управленческие решения и дает команду на их выполнение или собственноручно их выполняет.

Существует очень много систем, в состав которых входят люди, но нас интересуют только те, в которых люди осуществляют управление. Например, движущиеся объекты: автомобили, корабли, подводные лодки, самолеты, пилотируемые космические аппараты, и даже мотоциклы, велосипеды и, что на первый взгляд звучит довольно странно, всадники и более того – пешеходы! В двух последних примерах системы управления с человеком в качестве управляющей части имеют естественное, т.е. природное происхождение. Они приведены для полноты классификации. Нас конечно в первую очередь будут интересовать системы искусственного происхождения, спроектированные и созданные людьми.

Другой пример систем с человеком – организационные системы, т.е. системы, состоящие главным образом из коллективов людей, т.е. персонала, а также возможно включающие помещения, оборудование и т.д. Таковыми являются различные предприятия (производственные, обслуживающие, торговые и др.) или учреждения (образовательные, проектные, медицинские, детские, банковские и др.), а также различные подразделения (воинские, милицейские, специального назначения и т.п). Управление такими системами имеет свою специфику: оно складывалось исторически в процессе общения людей между собой и, как правило, регламентируется устными указаниями и письменными документами, например, приказами, должностными инструкциями, штатным расписанием. В отличие от термина *управление* (*control*), принятого в технических системах, для обозначения управления в организационных системах часто используют термин *менеджмент* (*management*). Но даже в этом случае, несмотря на чрезвычайную сложность объекта управления, автоматизация затронула и организационные системы: разработаны и успешно функционируют автоматизированные системы документооборота, программные бухгалтерские системы, системы автоматизированного проектирования и т.д.

Автоматизированные системы управления (АСУ) – широкий класс систем с человеком (с людьми), в которых объект управления представляет собой сложное техническое устройство или процесс, управлять которым приходится одновременно по многим переменным. Это влечет необходимость использования большого количества разнообразных датчиков и, соответственно, контуров обратной связи с локальными регуляторами, а также появление иерархических уровней управления и потребность в координации одновременной и согласованной работы системы на разных

уровнях. Все это приводит к усложнению алгоритмов управления и управляющей части системы в целом. Сложность объекта управления кроме многомерности может также проявляться в нестационарности, нелинейности и распределенности параметров его модели. Информацию о состоянии многочисленных переменных отображают операторам (диспетчерам) на специальных экранах, мнемосхемах, табло. Примером таких систем являются АСУ противовоздушной обороной, АСУ атомной электростанцией, АСУ технологическими процессами (АСУТП). Следует заметить, что системы управления многими из современных движущихся объектов по причине их сложности по существу также являются АСУ, например, АСУ пассажирским лайнером, истребителем 4-5 поколений, космическим аппаратом, подводным крейсером.

Все эти системы, безусловно, нуждаются в качественном и эффективном графическом интерфейсе, посредством которого человек может взаимодействовать с технической частью системы.

1.1. Человек-оператор в системах управления

Человек-оператор (Ч-О) – это человек, который работает в составе технической системы (например, системы ручного управления или АСУ). По существу человек-оператор – это искусственно выделенная часть свойств, возможностей и способностей человека, предназначенная для выполнения поставленных перед ним задач в составе технической системы. Человек рассматривается как звено системы наряду с другими звеньями. При этом другие многочисленные и разнообразные индивидуальные свойства человека физиологического, интеллектуального или эмоционального характера могут совершенно не учитываться, не приниматься во внимание. Таким образом, под человеком-оператором подразумевают некоторую упрощенную модель деятельности человека в составе системы.

Тем не менее, как будет видно из дальнейшего, правильный, т.е. системный подход к созданию и использованию такой упрощенной модели требует предварительного тщательного изучения всех особенностей, ограничений и возможностей человека в процессе управления, включая самые разнообразные аспекты его жизнедеятельности и их прямое или косвенное влияние на исследуемый процесс. Только после такого всестороннего изучения можно создать адекватную упрощенную и истинную модель человека, действующего в составе системы управления, т.е. человека-оператора. Этим вопросам посвящены три первые главы учебника.

1.2. Системы ручного управления

Исторически системы ручного управления появились намного раньше автоматических систем. Лодка с рулевым (кормчим) – хрестоматийный пример системы ручного управления, давший, кстати говоря, название всей

науке об управлении – кибернетике (по-гречески *ξιβερνωσ* – кормчий). А лодки, как известно, человек научился делать еще в древнейшие времена.

В системах ручного управления человек-оператор выполняет функции регулятора, поэтому на типовой структурной схеме САР он вместе с индицирующим устройством, сообщающим ему информацию об ошибке, замещает именно регулятор, как показано на рис. 1.

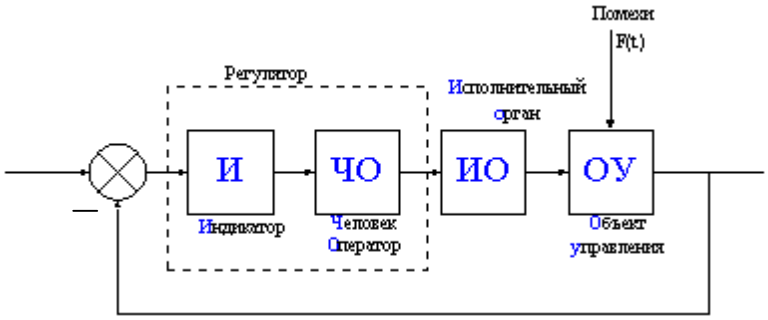


Рис. 1. Блок-схема системы ручного управления

Всадник на коне – пример другой системы ручного управления, в данном случае естественного происхождения, в которой исполнительный орган мощностью в одну лошадиную силу совмещен с объектом управления. Предельный случай естественной системы ручного управления, в которой вообще все принципиальные устройства совмещены в одном объекте, – это, например, пешеход.

Как уже было сказано, нас, прежде всего, интересуют системы искусственного происхождения, в которых объект управления представляет собой более или менее сложное техническое устройство. Примеры таких систем ручного управления: велосипедист на велосипеде (несмотря на то, что он тормозит или разгоняется с помощью ног), косильщик газонов с машинкой, рабочий, обрабатывающий на станке деталь, автомобиль с водителем, самолет с летчиком, пилотируемый космический аппарат в режиме ручного управления и т.д.

1.2.1. Общие черты систем ручного управления

Наиболее важный вывод, который можно сделать из анализа блок-схемы системы ручного управления (рис.1) состоит в том, что человек оказывается в прямой цепи управления. Наличие же человека-оператора в прямой цепи управления приводит к следующим результатам.

1. Человек в процессе управления испытывает значительные информационные перегрузки, так как он обладает ограниченной пропускной способностью (примерно 50 *бит/сек)*, а в реальных ситуациях управления скорость поступления информации к человеку значительно превышает эту величину.
2. Человек в процессе управления утомляется, и это приводит к тому, что ухудшаются его характеристики, связанные со способностью

воспринимать, перерабатывать информацию и принимать решения, а также выполнять управляющие действия; таким образом, при длительной работе модель Ч-О становится нестационарной.

3. Человек допускает ошибки в принципе (*humanum errare est*), а кроме того в силу причин, указанных в пунктах 1 и 2 вероятность ошибок значительно возрастает; следовательно понижается надежность работы всей системы, в составе которой человек действует (в штатных ситуациях).
4. Надёжность системы с человеком в нештатных ситуациях увеличивается за счет умения человека-оператора принимать быстрые и качественные управленческие решения в условиях неожиданно изменившейся обстановки, не предусмотренных заранее (инструкцией).

Нештатной называется непредвиденная ситуация, создавшаяся либо в результате случайных воздействий со стороны внешней среды, либо вызванная непредусмотренными изменениями в работе аппаратуры по иным причинам.

1.2.2. Следящие системы ручного управления

В ручных следящих системах различают два вида слежения: 1) слежение с компенсацией; 2) слежение с преследованием.

Слежение с компенсацией

Вспомним из курса теории управления: 1) если входное воздействие *x(t)* изменяется по какой-то известной программе *x(t)=f(t)=var*, то это система *программного управления*; 2) если же *x(t) = const*, то такая система называется системой *стабилизации*; 3) наконец, в случае, когда входной сигнал заранее неизвестен, а изменяется некоторым случайным образом, т.е. *x(t) = random*, система называется *следящей*.

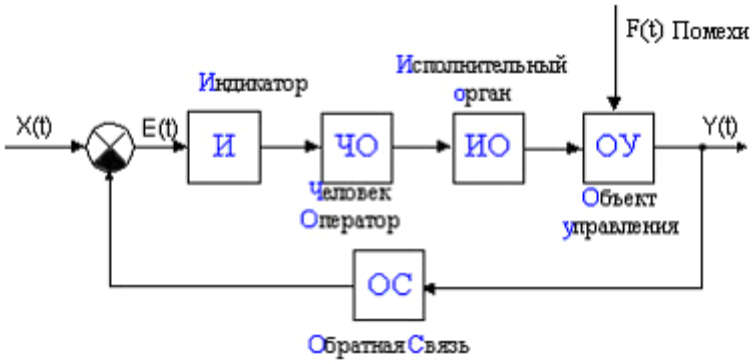


Рис. 2. Блок-схема системы слежения с компенсацией

В данном случае мы имеем дело именно со следящей системой. При компенсаторном слежении (рис. 2) на индикатор поступает и предъявляется

Ч-О только сигнал ошибки *ε(t)*, представляющий собой разность между входным и выходным сигналами.

Примером реализации такого режима может служить движение на автомобиле по дороге с нанесенной разметкой в условиях нулевой видимости. При этом водитель видит только отклонение автомобиля от линии разметки непосредственно перед ним, но не видит вперед: куда повернет дорога?

Слежение с преследованием

Характерной особенностью второго случая (рис. 3) слежения с преследованием является то, что на индикаторе одновременно отображаются как сигнал ошибки *ε(t)*, так и входной сигнал *x(t)*. И в случае слежения с компенсацией, и в случае слежения с преследованием оператор должен свести ошибку к нулю. Характеристики Ч-О при этом являются определяющими.

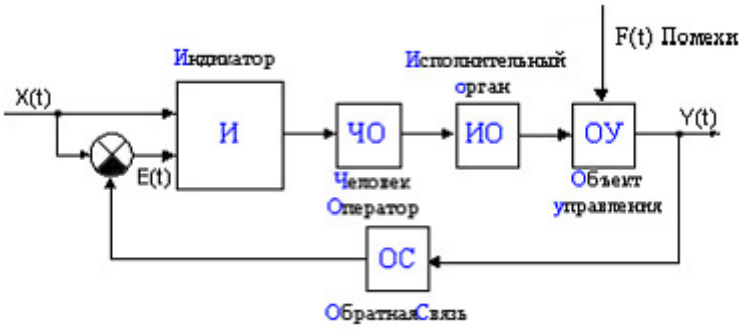


Рис. 3. Блок-схема системы слежения в режиме преследования

Однако при прочих равных условиях слежение с преследованием оказывается для Ч-О более удобным, т.к., наблюдая изменения входного сигнала, он может экстраполировать (предсказывать) будущие значения входного сигнала, что значительно облегчает слежение за траекторией. Примером реализации такого режима может служить движение на автомобиле по дороге с нанесенной разметкой в условиях хорошей видимости. Входным сигналом в данном случае является набегающая лента разделительной полосы дорожного полотна, а сигналом ошибки – отклонение автомобиля от нее. Водитель видит не только отклонение, но и характер изгибов дороги впереди по ходу движения и может заранее подготовить свои управляющие действия по управлению вектором скорости автомобиля.

1.3. Системы автоматизированного управления

Системы ручного управления обладают существенным недостатком: наличие Ч-О в прямой цепи управления влечет большую информационную и моторную нагрузку на него, что приводит к ошибкам, цена которых очень

велика. Для того чтобы избежать этого, Ч-О выводят во внешний контур управления, перепоручив автоматике его функции во внутреннем контуре. При этом человеку-оператору остаются функции контроля над ходом процесса управления с возможностью вмешаться в него при необходимости. Такое распределение функций между Ч-О и технической частью системы приводит к полуавтоматическим системам управления (рис. 4). Если мысленно повернуть рис. 4 на 90^0 по часовой стрелке, то Ч-О окажется сверху, т.е. на более высоком уровне иерархии, где, как известно из теории многоуровневых иерархических систем, ответственность за принятие решений больше, а информационная нагрузка – меньше.

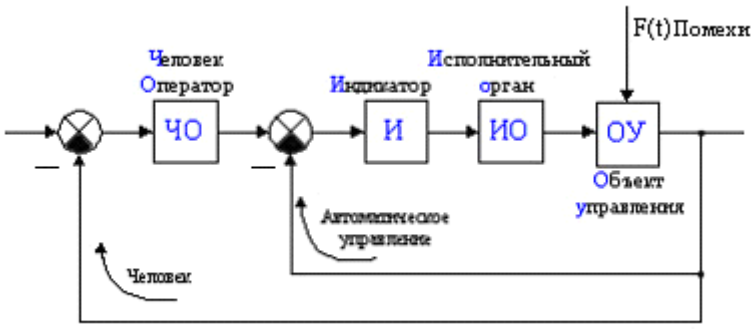


Рис. 4. Блок схема полуавтоматической системы управления

В принципе те же соображения используются и в автоматизированных системах управления (АСУ), однако эти системы являются значительно более сложными по своей структуре и подробно здесь не рассматриваются. Укрупненная схема АСУ приведена на рис. 5.

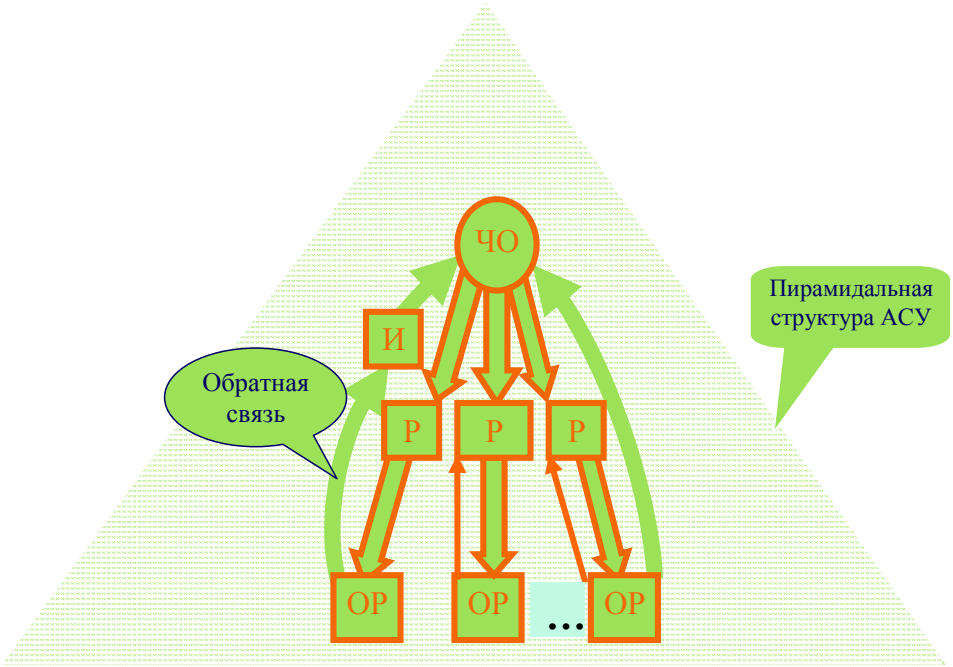


Рис. 5. Упрощенная схема автоматизированной системы управления (АСУ). ЧО – человек-оператор; Р – регулятор; ОР – объект регулирования; И – индикатор

1.4. Проблемы эргономического подхода к проектированию систем с человеком-оператором

1. Анализ задач человека в СУ и способов его связи с другими компонентами системы. При этом наибольшее внимание уделяется проблемам точности, надежности, быстродействия звена «человек-оператор».
2. Исследование групповой деятельности операторов, обслуживающих систему управления.
3. Анализ структуры деятельности Ч-О: определяются требования, которым должен удовлетворять оператор с точки зрения инженерной психологии (эргономики). В отдельных случаях требования к оператору могут быть удовлетворены после соответствующей подготовки (тренировки).
4. Исследование факторов, влияющих на эффективность работы операторов.

1.5. Упрощенная передаточная функция человека-оператора в режиме компенсаторного слежения

Американские исследователи Макруэ и Крэндэл (McRuer, Crandall) получили для режима компенсаторного слежения (рис. 6) передаточную функцию деятельности человека-оператора следующего вида:

$$W(S) = \frac{k \cdot e^{-snT}(T_1 S + 1)}{(T_2 S + 1)\ (T_3 S + 1)}.$$

Здесь отдельные составляющие описывают различные характерные особенности восприятия и переработки информации человеком и его реакции на нее.

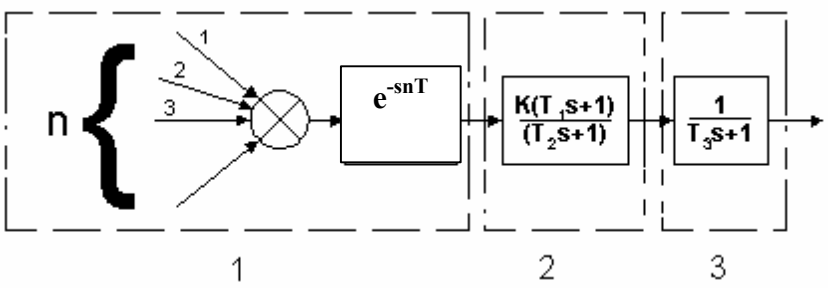


Рис. 6. Структура передаточной функции человека-оператора по Макруэ и Крэндэлу

Первый блок (1) описывает сенсорный вход человека, который связан с процессом получения и восприятия им информации. Во-первых, этот блок отражает наличие у человека фактически не одного, а нескольких входов (зрение, слух, осязание, обоняние, вкус, – соответственно различным органам его чувств). Так для зрительного анализатора, например, чувствительным элементом является сетчатка глаза. Требуется некоторое время на отображение картинки на сетчатке, преобразование графической информации в электрические сигналы, их передачу по глазному нерву в мозг и, наконец, распознавание поступившей информации. Именно этот интервал

времени и описывает звено чистого запаздывания. Итак, сенсорный вход характеризует процесс восприятия информации. Причем данная модель учитывает пропорциональное замедление процесса восприятия релевантной (важной) информации по некоторому входу, если по другим входам одновременно поступает также релевантная, но другая информация. Это замедление отражает коэффициент *n* в показателе экспоненты звена чистого запаздывания. Иначе говоря, если различная полезная информация поступает одновременно на органы зрения и слуха, то восприятие пойдет в 2 раза медленнее, чем если бы это была только видеоинформация или только звук.

Второй блок (2) характеризует переработку информации и принятие решения. Соотношение постоянных времени переработки информации $T_1 / T_2 = 0{,}1$ соответствует неподготовленному оператору, а для подготовленного оператора это соотношение приближается к единице: $T_1 / T_2 = 1$.

Третий блок (3) – описывает моторный выход человека, характеризующий отработку его мышцами электрического управляющего сигнала, переданного мозгом по нервной системе. Поскольку моторный выход описывает биомеханику человека, соответствующая постоянная времени значительно превышает постоянную времени переработки информации: $T_3 >> T_2$.

1.6. Реакции человека-оператора

В лабораторных условиях изучают реакции человека в различных условиях.
Различают 3 вида реакции:

1) сенсорно-моторная;
2) реакция на движущийся объект (РДО), или реакция слежения;
3) дизъюнктивная реакция (реакция выбора).

Простая сенсорно-моторная реакция определяется так: известен входной сигнал (стимул) и известна реакция оператора, замеряется интервал между ними. Время сенсорно-моторной реакции для различных анализаторов человека приведено в табл.1.

Время сенсорно-моторной реакции (мс) **Таблица 1**

№	Название анализатора	Нижняя граница	Верхняя граница
1	Зрительный анализатор	150	220
2	Слуховой анализатор	120	180
3	Тактильный анализатор	90	220
4	Болевой порог	130	890
5	Вестибулярный аппарат	400	400

Реакция слежения (РДО) включает во время, затрачиваемое на совмещение курсора, которым управляет человек-оператор, с некоторой отметкой (маркером), движущейся по неизвестной заранее оператору

траектории. Этот вид реакции зависит от времени предшествования, т.е. от того времени, в течение которого оператор наблюдал движение изображаемого объекта до начала своих действий. Это облегчает задачу оператору, поскольку он может экстраполировать (предугадывать) движение маркера. Если время предшествования превышает 300 миллисекунд, то время РДО составляет для зрительного анализатора 100-150 мс, если же время предшествования меньше 300 мс, то время РДО равно времени простой сенсорно-моторной реакции.

Дизъюнктивная реакция соответствует времени, затрачиваемому оператором на осуществление выбора наиболее подходящего решения из нескольких возможных, известных заранее, в зависимости от некоторых условий, накладываемых на стимулы. Латентный период для этого вида реакции значительно больше, чем в случае простой сенсорно-моторной реакции. В принципе время латентного (скрытого от наблюдателя) периода зависит от многих факторов: от интенсивности сигнала, его модальности и т.д.

1.7. Структура модели деятельности человека-оператора

Для упорядочения представлений о деятельности человека-оператора в целом в составе системы ручного управления полезно рассмотреть структуру модели, которую используют при исследованиях. Структурная схема модели деятельности человека-оператора, учитывающая работу его сенсоров, периферической и центральной нервной системы (ЦНС), биомеханический привод руки (ДР), а также работу органа управления (ОУ) и динамику объекта управления (ДО), показана на рис. 6.

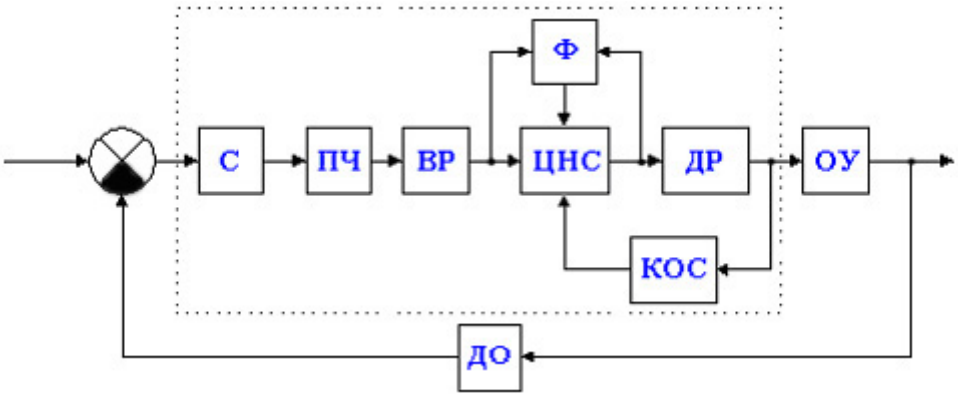


Рис. 6. Структурная схема модели деятельности человека-оператора

С – стимул; Ф – фильтр ;
ПЧ – порог чувствительности; ДР – динамика руки;
ВР – время реакции; ОУ – орган управления;
ЦНС – центральная нервная система; ДО – динамика объекта управления;
КОС – кинематическая обратная связь, используется для улучшения кинематики системы.

Каждый из блоков схемы может быть более или менее подробно описан соответствующими передаточными функциями и типовыми нелинейными

элементами, так что для конкретного объекта управления можно провести эксперименты моделирования динамики системы ручного управления. В результате могут быть получены переходные процессы в любой интересующей исследователя точке схемы. Обработка результатов моделирования позволит сделать объективные выводы о влиянии характеристик человека-оператора на работу системы, а также, возможно, дать полезные рекомендации по улучшению качества ее работы.

1.8. Безошибочность работы человека-оператора и его влияние на точность системы управления

Безошибочность работы системы, называемая иногда точностью, оценивается как произведение вероятностей безошибочной работы всех её элементов. По сути дела речь идет о надежности системы.

$$P_{cy} = \prod_{i-1}^{n} P_i$$

Если установлен менее точный элемент, то необходимо либо повысить его точность непосредственно, либо найти другие способы её улучшения. Наименее точным в системе является человек-оператор, поскольку вероятность совершения ошибок со стороны человека значительно превышает аналогичные значения вероятностей со стороны технических элементов системы.

1.8.1. Структура ошибок человека-оператора

Различают постоянную и переменную составляющие ошибки. Постоянная составляющая оценивается средним арифметическим от всех допускаемых ошибок, а переменная составляющая – среднеквадратической ошибкой.

Постоянная составляющая ошибки:

$$M_e = \frac{\sum_{i=1}^{n} e}{N}$$, где N - число испытаний.

$$M_{сист} = \sum_{i=1}^{n} M_e$$

Пример: определение дальности.

Ошибка, вносимая человеком-оператором: M_{e1}=50м.

Система дает составляющую ошибки: M_{e2} =+10м, M_{e3}=–20м, $M_{сист}$ = 50+10–20=40м.

Переменная составляющая ошибки оценивается среднеквадратичной ошибкой:

$$\sigma_e = \sqrt{\sum_{i=1}^{n} \sigma_{ei}^2} .$$

Если $\frac{M_{сист}}{\sigma_{сист}} \le \frac{1}{3}$, то постоянной ошибкой человека-оператора можно пренебречь.

Если $\frac{M_{сист}}{\sigma_{сист}} \geq \frac{1}{3}$, то необходимо принять меры по компенсации ошибки. Этого можно достичь, в том числе и за счет улучшения графического интерфейса человека-оператора. Человек может неправильно или неточно считывать информацию, отображаемую для него на экране дисплея, особенно если она представлена в неудобном виде. Иначе говоря, правильно спроектированный и удобный для оператора интерфейс позволяет уменьшить, а иногда и полностью компенсировать постоянную составляющую ошибки оператора.

1.9. Надежность человека-оператора

Под надежностью человека-оператора понимается его способность выполнять поставленную задачу. Эта способность (надежность) оценивается такой количественной характеристикой как вероятность безошибочного выполнения задачи в заданных условиях.

Известно, что в стрессовых ситуациях человек-оператор работает так, что жертвует точностью ради повышения устойчивости и живучести системы, которой он управляет.

1.9.1. Способы повышения надёжности

Надежность человека-оператора может быть увеличена посредством его обучения. Кроме того существуют структурные способы повышения надежности систем, которые применимы и для человека как элемента системы:

1) дублирование человека-оператора;
2) использование дополнительных обратных связей;
3) использование контролирующих устройств.

Дублирование подразумевает включение в работу параллельно дополнительного оператора, который выполняет ту же самую работу.

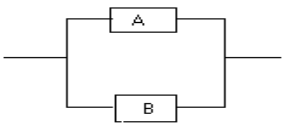


Рис. 8. Параллельное включение операторов

Рассмотрим пример параллельной работы двух операторов: *A* и *B* с одинаковой вероятностью безошибочной работы – 80%. Это означает, что

$$P_A=P_B=0.8$$

По формуле условной вероятности получим:

$$P=P_A*P_B+P_A*(1-P_B)+P_B*(1-P_A)=0.8*0.8+2*0.2*0.8=0.96$$

Таким образом, получается, что при параллельной работе двух одинаковых операторов (дублировании) вероятность безошибочной работы значительно увеличилась – до 96%. Это доказывает эффективность

дублирования. При этом, однако, следует иметь в виду, что затраты на реализацию системы управления также возрастают.

Использование дополнительных обратных связей подразумевает введение дополнительных датчиков, которые позволили бы измерять различные физические сигналы, соответствующие другим переменным системы помимо уже измеряемых ранее, и передавать эту информацию либо на соответствующие индикаторы человеку, либо на автоматические регуляторы. Таким образом, замыкаются контуры дополнительных локальных обратных связей, охватывающие человека, либо действующие параллельно ему. Это увеличивает устойчивость системы в целом и, в конечном счете, повышает надежность выполнения конкретной задачи, например, слежения.

1.10. Работоспособность человека-оператора

Под работоспособностью понимается способность человека выполнять предписанные ему операции. Лучше всего, если он выполняет свою работу с одинаковой интенсивностью и качеством, то есть стабильно. Стабильность означает постоянство таких его характеристик как время реакций, вероятность ошибок, пропускная способность и др. К сожалению, природа человека такова, что он не может сохранять свои характеристики стабильными в течение длительного времени. После некоторого интервала интенсивной работы наступает утомление, которое проявляется субъективно в ощущении усталости, ухудшении самочувствия, а объективно – в снижении значений характеристик работы. Поэтому человеку требуются своевременные перерывы в работе для отдыха и восстановления характеристик.

Таким образом, работоспособность является неким интегральным показателем, включающим в себя целый ряд характеристик, всесторонне описывающих работу человека-оператора. Экспериментально получены данные по работоспособности операторов, представленные на рис. 9.

Работоспособность R отображена в форме графика в зависимости от времени суток в некоторых относительных единицах. На графике явно можно выделить три участка: A, B и C. Участок A соответствует так называемой *врабатываемости*, когда человек настраивается на работу, подготавливается к ней, отвлекаясь от помех и концентрируя внимание на выполнении требуемой задачи. При этом рабочие характеристики человека достигают постепенно своих оптимальных значений. Участок B соответствует периоду *стабильной работы* человека с наилучшими значениями характеристик и, соответственно, с наилучшими показателями качества работы. Участок C отражает наступление *утомления*, выражающегося в постепенном ухудшении значений характеристик деятельности оператора и показателей качества его работы.

При этом следует понимать, что, как и любая другая модель, график на рис. 9 описывает характер указанных изменений лишь приближенно и нарочно подчеркивает резкое отличие участков.

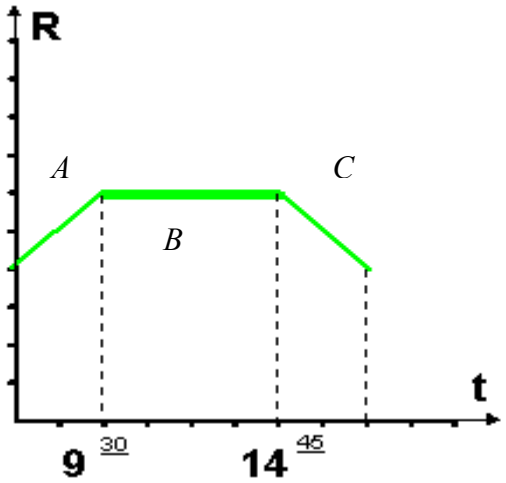


Рис. 9. Зависимость работоспособности человека-оператора от времени
A - врабатываемость; *B* - период стабильной работы; *C* – утомляемость

Соединение между этими участками на реальном экспериментальном графике выглядит значительно более плавным. Кроме того здесь показано качественное изменение усредненного показателя работоспособности по группе операторов.

1.11. Помехоустойчивость человека-оператора

На прием и переработку информации влияют как *внешние,* так и *внутренние* факторы.

1) К внешним факторам относят: шум, вибрацию, уровень электромагнитных излучений, уровень радиации, запыленность и загазованность атмосферы, давление, температуру и влажность, окружающей среды и т.д.
2) К внутренним факторам относятся: физиологические параметры организма и состояние психики человека.

При приеме информации по нескольким каналам одновременно один вид информации может служить помехой для приема информации другого вида.

1.12. Оптимальная зона условий работоспособности человека-оператора

Существует понятие оптимальной зоны условий, при которых человек-оператор работает наилучшим образом. Признаки оптимальной зоны:

1) наиболее высокие показатели сенсорного входа и моторного выхода оператора;
2) длительное сохранение работоспособности;
3) наибольшая стабильность характеристик оператора;
4) адекватность оператора (одинаковая реакция на один и тот же стимул).

Глава 2. Анализаторы человека

Анализаторами называют чувствительные элементы, или сенсоры человека, соответствующие его органам чувств. Как известно, человек объективно обладает пятью органами чувств: зрением, слухом, обонянием, осязанием и вкусом.

2.1. Характеристики анализаторов

2.1.1. Чувствительность анализаторов. Различают две характеристики чувствительности: абсолютную чувствительность и дифференциальный порог.

Абсолютная чувствительность оценивается значениями верхнего и нижнего порога, т.е. диапазоном чувствительности. Для зрительного анализатора, например, нижний порог составляет около 300 ммк, а верхний – до 1000 ммк (оптический диапазон).

Количественная оценка абсолютной чувствительности выражается:

$$E = \frac{1}{I}$$

I – пороговая величина раздражителя (интенсивность стимула).

Дифференциальный порог характеризует разрешающую способность, т.е. наименьшую разность между двумя значениями стимула, которые различает человек.

Дифференциальный порог характеризуется коэффициентом K:

$$K = \frac{\Delta I}{I_0}$$

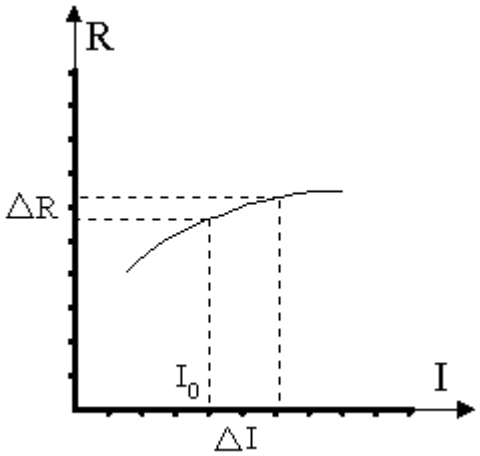


Рис. 10. Зависимость реакции от стимула (статическая характеристика)

ΔI – минимальная величина приращения, соответствующая различению стимулов;

I_0 – величина исходного стимула;

R - величина реакции на соответствующий стимул.

2.2. Зрительный анализатор человека и его свойства

Зрительным анализатором человека служат, как известно, глаза.

2.2.1. Строение глаза человека

Глаз человека состоит из хрусталика, радужной оболочки, стекловидного тела, глазных мышц, сетчатки, в свою очередь состоящей из миллионов т.н. «колбочек» и «палочек», склеры, конъюктивы, роговицы, а также из нервных волокон, соединяющих глаз с мозгом.

Рис.11. Внешний вид глаза человека

Глаз можно назвать сложным оптическим прибором. Его основная задача – «передать» правильное изображение зрительному нерву.

Глазное яблоко имеет шарообразную форму, его диаметр составляет около 24 мм. Внутри него находится внутриглазная жидкость, хрусталик и стекловидное тело (рис. 12).

Стекловидное тело – прозрачная желеобразная структура глаза, занимающая по объему и весу большую его часть. Оно обеспечивает необходимую форму глазного яблока и поддерживает нужное для работы глаза взаимное расположение между собой внутренних частей глаза. Стекловидное тело ограничено тремя глазными оболочками: фиброзной, сосудистой и сетчаткой. Плотная непрозрачная внешняя (фиброзная) оболочка, образующая форму глаза, называется склерой. К склере крепятся 6 глазодвигательных мышц. В ней находится небольшое количество нервных окончаний и сосудов.

На переднем, видимом участке склера частично покрыта конъюнктивой и переходит в прозрачную роговицу. В роговице отсутствуют кровеносные сосуды. За счет своей куполообразной формы она имеет большую преломляющую силу (примерно в 40 диоптрий – у каждого человека различна). Роговица входит в оптическую систему глаза.

Средний слой – сосудистая оболочка, содержит кровеносные сосуды, которые обеспечивают глаз кислородом. В сосудистую оболочку входит цилиарное (ресничное) тело с его ресничными поясками и радужка.

Ресничное тело – часть сосудистой оболочки глаза с круговой кольцеобразной мышцей в своей толще. Эта мышца может регулировать силу преломления лучей хрусталиком, что обеспечивает четкое видение предметов, находящихся на разных расстояниях от глаза. Отростки ресничного тела вырабатывают водянистую влагу, необходимую для питания прозрачных частей глаза.

Радужка – по форме похожа на круг с отверстием внутри (зрачком). Радужка состоит из мышц, при сокращении и расслаблении которых размеры зрачка меняются. Именно радужка придает глазу его цвет, в зависимости от количества пигментных клеток. Между роговицей и радужкой находится пространство – так называемая передняя камера глаза, заполненная прозрачной внутриглазной жидкостью (водянистой влагой).

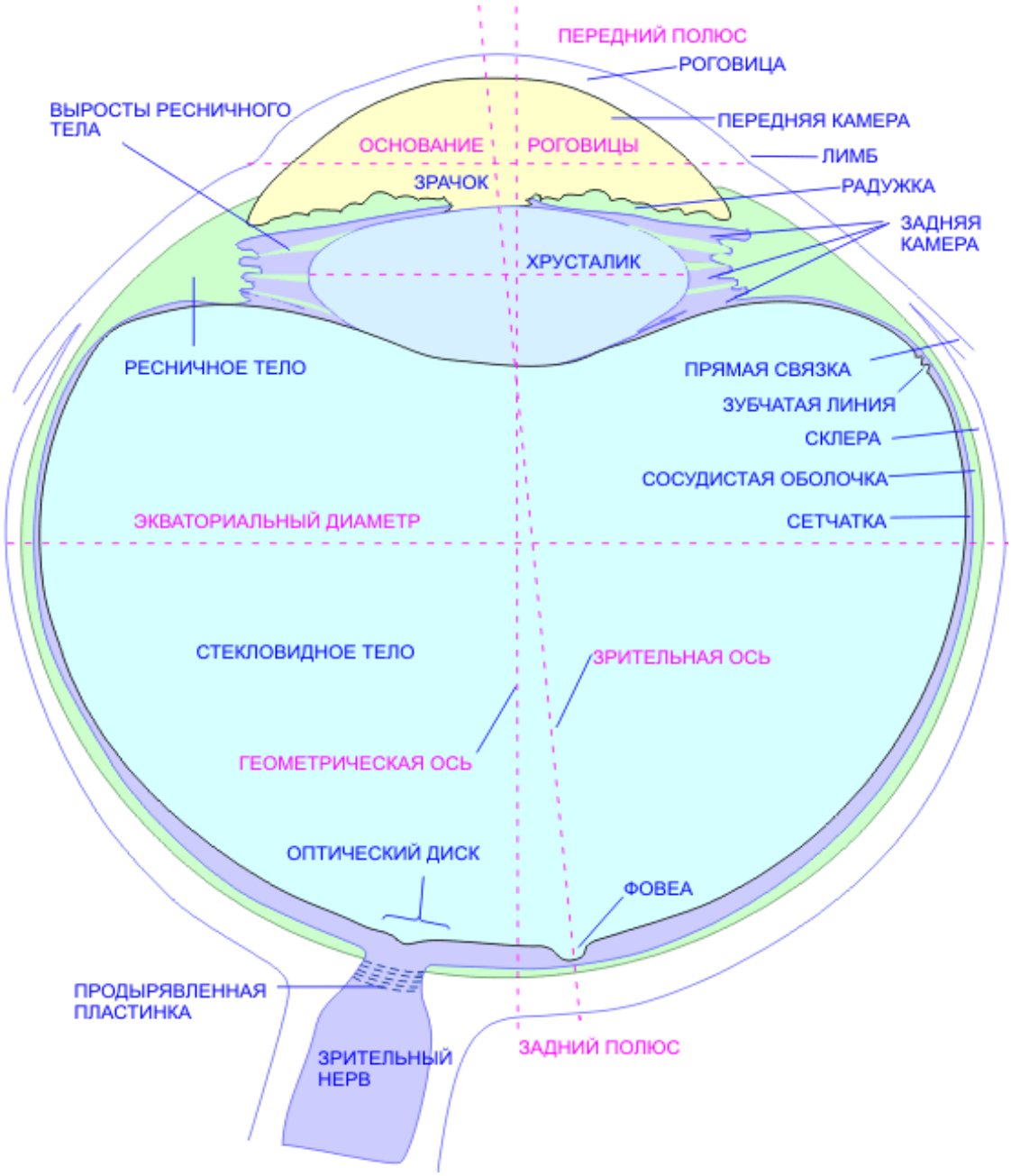


Рис. 12. Строение глаза

Роговица и внутриглазная жидкость пропускают световые лучи, которые попадают внутрь глазного яблока через зрачок. Попадание внутрь глаза лучей яркого света вызывает рефлекторное сужение отверстия зрачка. При слабом освещении зрачок расширяется. Непосредственно за зрачком находится прозрачный хрусталик, имеющий форму двояковыпуклой линзы. Хрусталик прозрачен, эластичен – может менять свою форму, почти мгновенно «наводя фокус», за счет чего человек видит хорошо и вблизи, и вдали. Вокруг него располагается кольцевая мышца. Можно сказать, что хрусталик – выпуклая живая линза с силой преломления лучей света примерно в 20 диоптрий, настраивающая фокусировку пучка лучей света, попадающего внутрь глаза через роговицу и зрачок точно на центральную зону сетчатки. Хрусталик, как и роговица, входит в оптическую систему глаза.

После того, как лучи пройдут сквозь хрусталик, они проникают через стекловидное тело – желеобразную прозрачную субстанцию, которая заполняет собой всю внутреннюю часть глазного яблока. В конечном итоге, лучи света попадают на внутреннюю, очень тонкую оболочку глаза – сетчатку. Сетчатка прилегает к сосудистой оболочке, но на многих

участках неплотно. Именно здесь она и имеет тенденцию отслаиваться при различных заболеваниях сетчатки.

Сетчатка имеет чрезвычайно сложное строение. Она состоит из фоторецепторов (они чувствительны к свету) и нервных клеток. Клетки-рецепторы, расположенные в сетчатке, делятся на два вида: колбочки и палочки. Палочки обладают высокой светочувствительностью и позволяют видеть при плохом освещении, также они отвечают за периферическое зрение. Колбочки, наоборот, требуют для своей работы большего количества света, но именно они позволяют разглядеть мелкие детали (отвечают за центральное зрение), дают возможность различать цвета. Наибольшее скопление колбочек находится в центральной ямке (макуле), или «желтом пятне», обеспечивая самую высокую остроту зрения. Вся остальная часть сетчатки дает человеку возможность иметь периферическое зрение. Сетчатка является частью мозга, выросшей вперед из черепа в процессе развития каждого человека еще в утробе матери.

В палочках и колбочках, вырабатывающих фермент родопсин, происходит преобразование энергии света (фотонов) в электрическую энергию нервной ткани, т.е. фотохимическая реакция. Возбуждение проводится по отросткам нейронов, образующих зрительный нерв. Можно сказать, что он состоит из суммы нервных нитей, идущих от световоспринимающих клеток сетчатки в мозг. По этому пучку нервных нитей передаются биологические электрические токи. Эти токи возникают в нервных клетках сетчатки как реакция на действие лучей света. Таким образом, световая информация, поступающая в глаз, превращается в электрические сигналы, которые по зрительному нерву передаются в мозг.

Подробнее строение глаза предлагается изучить (повторить) самостоятельно, воспользовавшись учебником или справочником по офтальмологии.

Так, например, можно уточнить, что в задней части глаза, где зрительная ось пересекает сетчатку, имеется углубление – фовеа (*fovea*), обильно населенная колбочками, отвечающими за зрение при дневном освещении. Фовеа окружена более широкой круговой областью сетчатки – *area centralis*, называемой в тех случаях, когда она пигментирована, как у человека и других приматов, желтым пятном (*macula lutea*). Оно также, хотя и в меньшей степени, приспособлено к зрению высокого разрешения. Со стороны носа, а следовательно вне оптической оси, к *area centralis* примыкает зрительный диск, где собираются зрительные нервные волокна, покидающие глаз в составе зрительного нерва. Эта область лишена фоторецепторов, нечувствительна к свету и именуется слепым пятном – это «слабое место» и в склере, которая в остальном жесткая. Эта слабость компенсируется тем, что зрительный нерв проходит здесь через перфорированную коллагенозную пластинку, так называемую продырявленную пластинку, непосредственно под оптическим диском.

2.2.2. Свойства глаз

Аккомодация – свойство изменения кривизны хрусталика в зависимости от расстояния до рассматриваемого предмета. Чем больше это расстояние, тем меньше кривизна хрусталика и наоборот. Аккомодация связана с необходимостью помещать рассматриваемый объект в фокус выпуклой линзы, которую представляет собой хрусталик.

Параметры, характеризующие ***скорость движения глаз***, определяются по специальным таблицам.

Адаптация – изменение чувствительности глаза по яркости при воздействии раздражителя. Изменение чувствительности может достигать 200000 раз.

Длительность сохранения остаточного образа (подобно явлению послесвечения осциллографа) составляет от 0,05 до 1 с.

Острота зрения – способность глаза различать детали предмета.

$$O = \frac{1}{\gamma},$$

γ – угол, на котором два предмета различаются между собой.

Поле зрения без поворота головы и напряжения глазных мышц составляет в горизонтальной плоскости примерно 180° и еще $\pm 45^\circ$ за счет поворота головы. Причем для правого глаза 94° вправо и 60° влево, а для левого наоборот: влево 94° и вправо 60° (из-за того, что нос заслоняет видимость). В вертикальной плоскости: вверх 60°, а вниз 75°; с поворотом (поднятием или опусканием) головы – еще $\pm 30^\circ$.

На рисунке 13 приведены зоны видимости, достижимые с учетом поворота головы в сторону взгляда. Оптимальная (нормальная) линия взгляда соответствует минимальной активности мышц затылка и, следовательно, наименьшей утомляемости человека при данной рабочей позе.

Зоны видимости, построены с учетом уменьшения чувствительности глаза от центра поля зрения к периферии.

Таблица 2

Часть тела	Характер движения	Угол поворота, °	
		Среднее значение M	Разброс $\pm\sigma$
Голова	Наклон головы назад	60	34…85
	Наклон головы вперед	44	25…70
	Наклон головы вправо	40	24…60
	Наклон головы влево	42	26…62
	Поворот головы вправо	72	53…86
	Поворот головы влево		

Центром поля зрения называется точка, на которую направлен сосредоточенный взгляд. Если световой сигнал находится на периферии поля зрения, то латентный период двигательной реакции увеличивается. Однако периферическое зрение более чувствительно к слабым и движущимся световым сигналам. При поступлении такого сигнала человек переводит на него взгляд для детального анализа. В пределах поля зрения постоянно совершаются микродвижения глаз, причем эти движения происходят скачками. Время каждого такого скачка — сотые доли секунды. Время перевода взгляда с одной точки пространства к другой зависит от углового расстояния между этими точками и от маршрута движения взгляда. За счет микродвижений глаз производится поиск предмета, считывание показаний прибора, опознание предмета. Для выполнения этих функций оптимальна зона, ограниченная углом примерно 15° вверх-вниз и вправо-влево от нормальной линии взгляда.

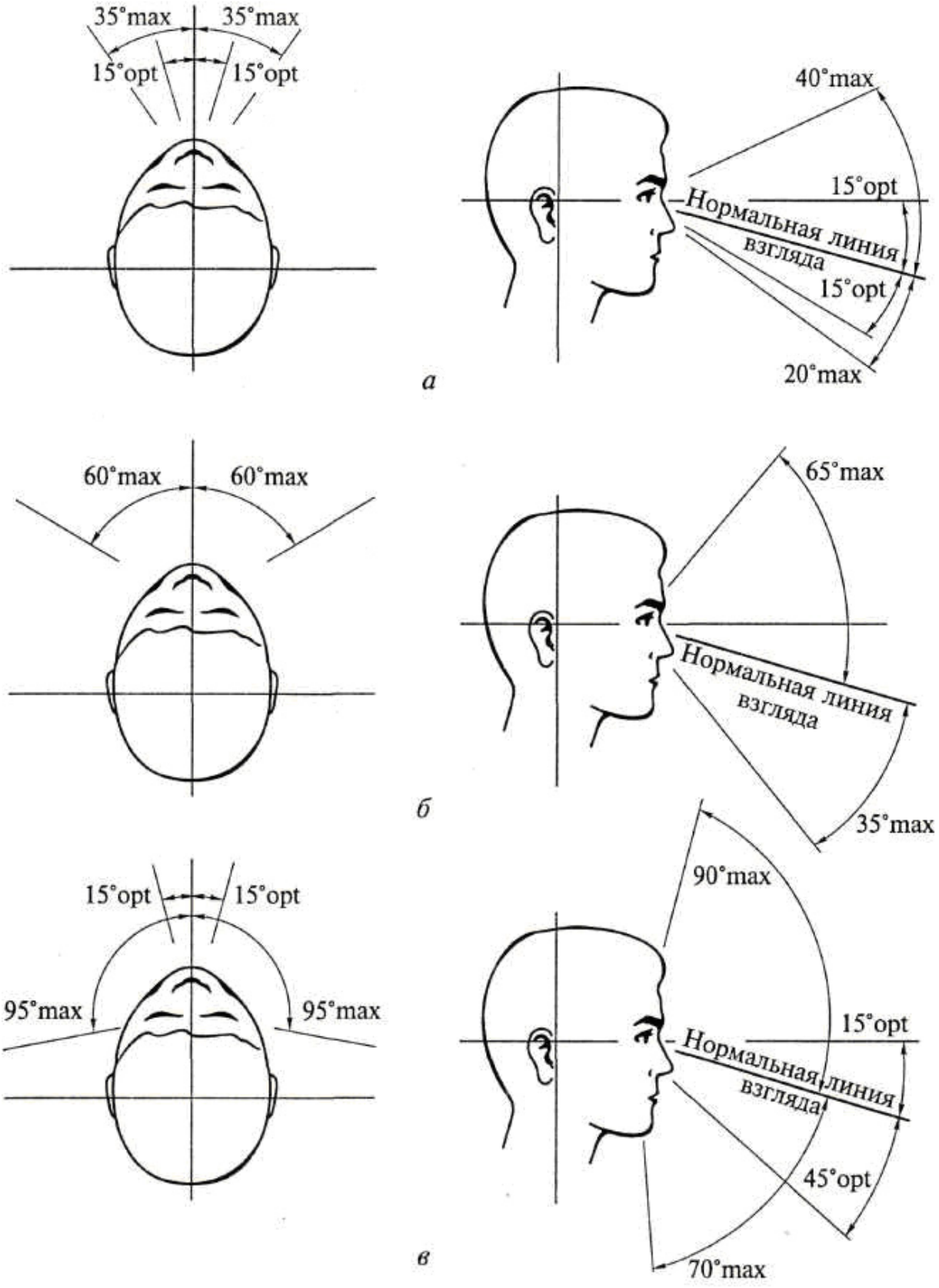


Рис. 13. К понятию поля зрения человека

Конвергенция – свойство, заключающееся в том, что оптические оси глаз сходятся (пересекаются) на рассматриваемом предмете (рис. 14).

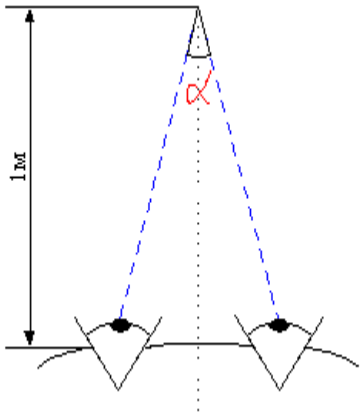


Рис. 14. К понятию конвергенции

Стереоскопичность – свойство, которое позволяет измерять расстояние до предмета (за счет наличия двух глаз). *Параллакс* – расстояние между глазами. Чем больше параллакс, тем точнее измеряется расстояние до рассматриваемого предмета.

Цветовое восприятие – свойство, позволяющее воспринимать электромагнитные волны длиной в диапазоне от 380 ммк (фиолетовый) до 760 ммк (красный).

2.2.3. Характеристики зрительного анализатора человека

Восприятие зрительного анализатора человека описывается тремя основными характеристиками: тоном, насыщенностью и яркостью.

1. ***Тон*** – атрибут визуального восприятия, согласно которому рассматриваемая область кажется обладающей одним из воспринимаемых цветов (красного, желтого, зеленого и синего) или комбинацией любых двух из них. Тону соответствует определенная длина волны $\lambda_в$ из диапазона абсолютной чувствительности (оптического диапазона), составляющего приблизительно 380 – 760 ммк.

2. ***Насыщенность*** – это пропорция чистого (красного, желтого, зеленого и т.д.) и белого цветов, необходимая для того, чтобы определить цвет. Насыщенность показывает, насколько чистым является цвет (насколько мало в нем белого цвета). Насыщенность указывается в процентах. Красный цвет имеет насыщенность, равную 100%, а серые цвета имеют насыщенность, равную нулю.

3. ***Яркость*** (светлота); характеризуется коэффициентом яркости

$$r = \frac{I}{B_0},$$

где B_0 – яркость абсолютно белого тела (окись магния по гипсу).

Иногда используется ахроматический, т.е. нецветной ряд (градации серого цвета, или *grey scale*).

Условно можно основные семь цветов радуги: красный, оранжевый, желтый, зеленый, голубой, синий, фиолетовый, – изобразить в виде секторов кольца, которое окажется незамкнутым (рис. 15). Чтобы замкнуть цветовое кольцо потребовался искусственный пурпурный цвет, получаемый смешением красного и синего (фиолетового).

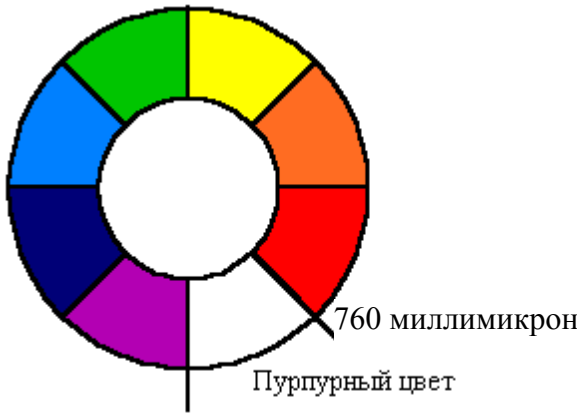


Рис. 15. Цветовое кольцо

Абсолютный диапазон цветовой чувствительности, или оптический диапазон, составляет от 380 ммк (фиолетовый цвет) до 760 ммк (красный).

Дифференциальный порог цветовой чувствительности, т.е. различаемая разница длины волны, составляет:

$$\Delta\lambda = 1 \text{ ммк}.$$

Относительная величина дифференциального порога составляет:

$$\frac{\Delta I}{I} = k = 0{,}01\,.$$

Временной порог цветовой чувствительности составляет от 0,001 с до 0,01 с.

2.2.4. Особенности цветового восприятия

У отдельных людей встречаются следующие отличительные особенности цветового восприятия.

1. Явление ахроматического интервала.
2. Явление световой и цветовой контрастности.
3. Явление последовательного цветового контраста.
4. Воздействие цвета на психологию человека. Сине-зеленая часть спектра действует успокаивающе, вызывает ощущение прохлады, а красно-оранжевая раздражает и вызывает ощущение тепла. Зеленый расширяет пространство, темные цвета сужают пространство.
5. Ненормальности цветового восприятия:

а) различаются все семь цветов, но восприятие – пороговое;

б) частичная цветовая слепота;

в) полное отсутствие цветового восприятия (ахроматизм).

Частичная цветовая слепота, являющаяся одним из видов нарушения цветового зрения, носит также название: дальтонизм. Дальтонизм впервые описан в 1794 г. английским ученым Дж. Дальтоном, который сам страдал этим недостатком. Дальтонизм встречается у 8% мужчин и у 0,5% женщин. Как уже упоминалось, в центральной части сетчатки человека расположены цветочувствительные нервные клетки, которые называются «колбочки». Они содержат три типа цветочувствительных пигментов белкового происхождения. Один тип пигмента чувствителен к красному цвету, другой – к зеленому, а третий – к синему. Точнее, они чувствительны к длине волны, соответствующей красному, зеленому и синему цвету в нашем понимании. Видение всех красок мира обеспечивается «складыванием» именно этих трех цветов в нашем мозге. У человека пик чувствительности этих пигментов приходится на длину волны 552-557 нм для красного, 530 нм для зеленого и 426 нм – для синего цвета. Это – нормальное, трихроматическое цветоощущение. При выпадении одного из этих элементов наступает частичная цветовая слепота – дихромазия. Лица, страдающие дихромазией, различают цвета главным образом по их яркости; качественно они способны отличать в спектре лишь «тёплые» тона (красный, оранжевый, жёлтый) от «холодных» тонов (зелёный, синий, фиолетовый). Цветослепые на один цвет и люди с пониженным цветовым зрением воспринимают краски окружающего их мира иначе, чем мы, но часто не замечают своего отличия от других. Происходит это потому, что цветослепые с детства учатся называть цвета обыденных предметов общепринятыми обозначениями. Они слышат и запоминают, что трава – зеленая, небо – синее, кровь – красная. Кроме того, они сохраняют способность различать цвета по степени светлости.

Среди дихроматов различают слепых на красный цвет (протанопия), у которых воспринимаемый спектр укорочен с красного конца, и слепых на зелёный цвет (дейтеранопия). При протанопии красный цвет воспринимается более тёмным, смешивается с тёмно-зелёным, тёмно-коричневым, а зелёный – со светло-серым, светло-жёлтым, светло-коричневым. При дейтеранопии зелёный цвет смешивается со светло-оранжевым, светло-розовым, а красный – со светло-зеленым, светло-коричневым. Слепота на фиолетовый цвет – тританопия, встречается крайне редко и практического значения не имеет. При тританопии все цвета спектра представляются оттенками красного или зелёного. В некоторых случаях наблюдается лишь ослабление цветоощущения – протаномалия (ослабление восприятия красного цвета) и дейтераномалия (ослабление восприятия зелёного цвета).

Все формы врождённой цветовой слепоты являются наследственными. Женщины являются проводником этой патологической наследственности; сами сохраняют нормальное зрение и оказываются цветослепыми лишь тогда, когда имеют цветослепого отца наряду с хотя бы гетерозиготной по этому гену матерью.

Приобретённые расстройства цветового зрения могут возникать при различных заболеваниях органа зрения и центральной нервной системы; поражается один или оба глаза и нередко – на все основные цвета. Расстройства цветового зрения выявляют при помощи специальных таблиц или спектральных приборов. Лечению дальтонизм не подлежит.

2.3.Мнимые эффекты зрения

Это группа свойств, к которым относят такие известные явления как кажущееся движение, оптический обман и некоторые другие.

Пример. Маргарет Тэтчер – это выдающаяся женщина-политик конца XX века. Но мы не будем вдаваться в политику, а хотим предложить вам взглянуть на парочку портретов этой железной леди. Обратите внимание, на то, что

портреты на рис. 16, перевёрнутые вверх ногами, воспринимаются практически одинаковыми.

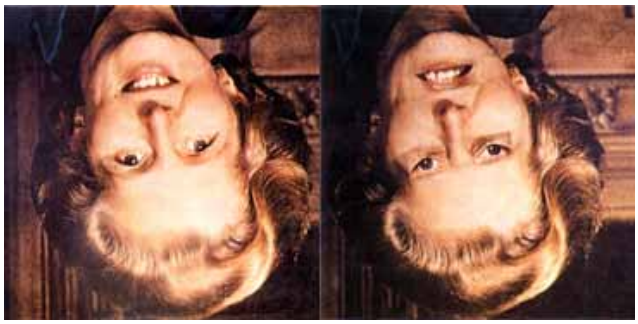


Рис. 16. К «иллюзии переворота», подобной эффекту персептивной готовности

После их переворачивания с Маргарет происходит удивительная метаморфоза.

Рис. 17. Перевернутый рис. 16

Портрет Маргарет Тэтчер с правой стороны на рис. 17 уже выглядит как то не по-человечески… ☺ Эта иллюзия чем-то напоминает эффект персептивной готовности, когда наш мозг видит что-то более привычное для себя, а не то, что есть на самом деле.

Еще один пример. Посмотрите, какой интересный французский солдат с роскошными усами изображен на рис. 18.

Рис. 18. Французский солдат

А вот так, легко и просто, всего лишь методом перевёртывания он превращается в лошадь (рис. 19).

Рис. 19. Лошадь, в которую превратился солдат ☺

Другой пример. Танцующие человечки (рис. 20) – это замечательная оптическая иллюзия, в которой используется *эффект движения*. Начните рассматривать этих человечков, и они пустятся в пляс, исполняя весьма необычный танец. Пластика их движений не оставит вас равнодушными. И помните, человечки станут неподвижными сразу после того, как только вы перестанете их рассматривать.

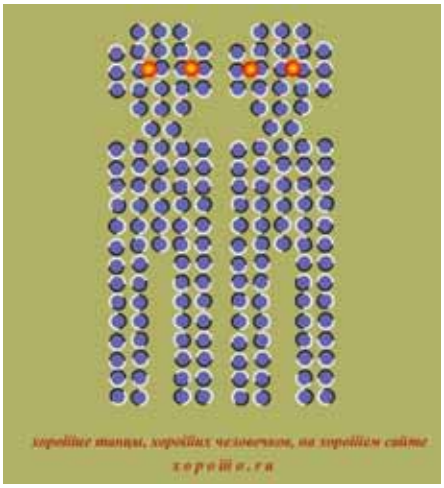


Рис. 20. Танцующие человечки

Иллюзия Селфриджа. Если вы хоть немного знакомы с английским языком, то для вас не составит особого труда прочитать название домашнего животного на рис 21.

Рис. 21. К иллюзии Селфриджа

Как видно из названия, первым этот обман зрения описал Селфридж (*Selfridge*, 1955). Суть его заключается в том, что в зависимости от контекста один и тот же символ воспринимается как “H” или как “A”. Посмотрите внимательно, ведь на картинке написана абракадабра THE CHT, а не THE CAT.

Эффект Струпа. Внимательно посмотрите на рис. 22. Ваша задача, быстро называть цвет слова, а не само слово!

Рис. 22. К эффекту Струпа

Надписи на картинке обозначают цвета, но цвет слов не соответствует названиям. Наблюдаем конфликт цвета шрифта и значения слова. В результате, возникает лёгкое замешательство.

Кстати, этот тест доказывает, что наш мозг использует каждое полушарие для различных действий. Ваше правое полушарие заставляет вас назвать цвет слова, в то время как левое упорно читает само слово. Впервые этот эффект был исследован в работах Джона Струпа (*John Ridley Stroop*, 1935) и в психологической литературе встречается как тест Струпа или эффект Струпа.

Вот еще один тест, позволяющий понять, каким именно образом человек читает текст (рис. 23).

По рзелульаттам илссеовадний одонго анлигйсокго унвиертисета, не иеемт занчнеия, в кокам пряокде рсапожолены бкувы в солве. Галвоне, чотбы преавя и пслоендяя бквуы блыи на мсете. Осатьлыне бкувы мгоут селдовтаь в плоонм бсепордяке, всё рвано ткест чтаитсея без побрелм. Пичрионй эгото ялвятеся то, что мы не чиатем кдаужю бкуву по отдльенотси, а всё солво цликеом.

Рис. 23. Тест на чтение текста

Теперь рассмотрим уже упомянутый *эффект персептивной готовности*, который выражает свойство человека адаптироваться к процессу распознавания графических образов, приобретая некоторую инерционность. Посмотрите на рис. 24 и дайте быстрый ответ, на вопрос – “какой символ в центре?”.

Рис. 24. К эффекту персептивной готовности

Этот тест наглядно демонстрирует эффект персептивной готовности (*персепция* – восприятие). Суть его заключается в том, что наш мозг распознает что-то более привычное для себя, а не то, что есть на самом деле. В данном тесте в зависимости от того, откуда вы начали читать, вы готовы увидеть разные символы. Если сверху вниз, то число 13. Если слева направо, то букву “B”.

Пример распознавания изображений. Вы когда-нибудь замечали, что если приблизиться к телевизору слишком близко, то можно увидеть, что картинка на самом деле состоит из тысяч маленьких точек. По сути дела это и есть так называемые пиксели. Так же формируется и картинка на мониторе вашего компьютера.

Посмотрите на весьма бессмысленное изображение на рис. 25. Это наглядная демонстрация принципа работы телевидения. Теперь отодвиньтесь от монитора. Продолжайте отдаляться до тех пор, пока не увидите… Что вы видите? Прокомментируйте свою находку.

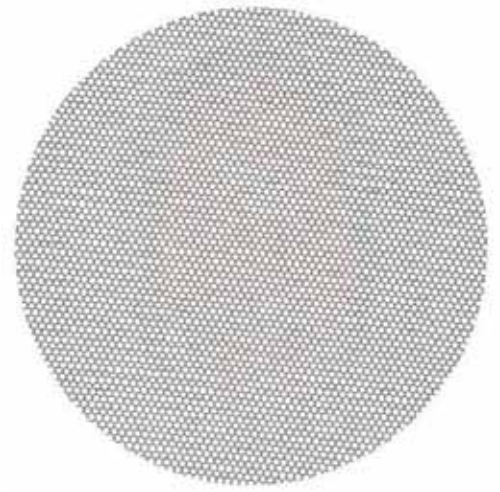


Рис. 25. Эффект пикселей

Иллюзия покосившихся синих квадратов. Очень интересный оптический фокус. Глядя на рис. 26, мозг уверяет вас в том, что синие квадраты в центре этой картинки, немного перекосило, и их то и дело клонит на бок. Но, расфокусировав взгляд или просто немного отойдя от монитора компьютера, вы поймете, что это правильные четырёхугольники, и что это всего лишь иллюзия.

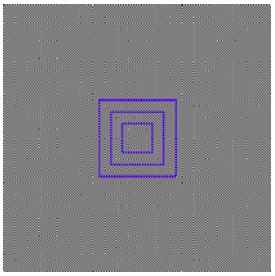


Рис. 26. Иллюзия покосившихся квадратов

Иллюзия Геринга, или как её ещё называют в психологической литературе – *иллюзия веера*, наглядно демонстрирует нам эффект искажения (рис. 27). Две горизонтальные линии кажутся выпуклыми.

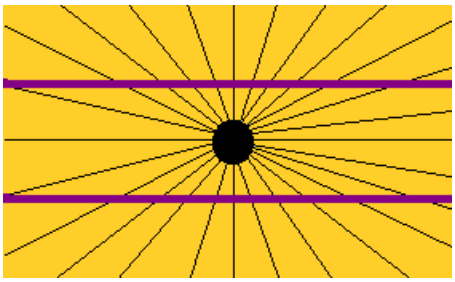


Рис. 27. Иллюзия веера

На самом деле эти линии прямые и параллельны друг другу.

На рис. 28 показана *иллюзия Цолльнера* (*Zolliner*, 1860). Иллюзии такого типа относятся к классике оптических иллюзий.

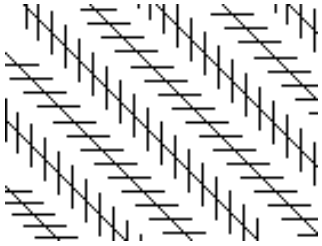


Рис. 28. Иллюзия Цолльнера

Суть этого обмана зрения заключается в том, что прямые на самом деле параллельны, хотя таковыми совсем не кажутся. Как обычно, эту иллюзию на предмет искажения, вы можете проверить самостоятельно, при помощи линейки.

Эффект последействия. На рис. 29 чёрно-белая картинка с изображением логотипа Бэтмэна. Ваша задача смотреть на неё в течении 30 секунд, и затем перевести взгляд на чистую белую поверхность (потолок, чистый лист бумаги, обои на стене).

Рис. 29. К эффекту последействия

Эффект мерцания. На рис. 30 в узлах пересечения серых линий на черном фоне изображены круглые точки, цвет которых весьма затруднительно определить: то ли они черные, то ли белые…

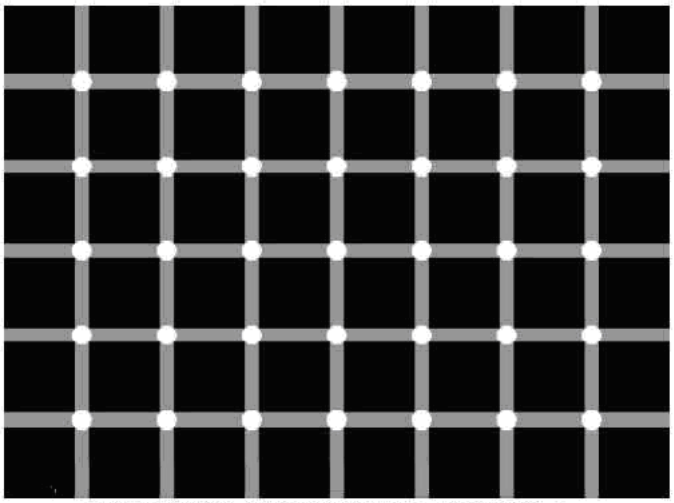


Рис. 30. Эффект мерцания. Попробуйте подсчитать количество черных точек ☺

Рисунки этих и других оптических эффектов Вы можете найти, например, на сайте *xopoiiio.ru*.

Рассмотренные мнимые эффекты зрения в контексте нашего предмета представляют собой не просто курьезные случаи, но явления, требующие особого внимания и изучения, поскольку они связаны со спецификой восприятия графической информации человеком. Любая ошибка человека-оператора, связанная с неадекватным восприятием графического интерфейса, чревата негативными последствиями для всей системы в целом, включая человека (людей).

2.4. Звуковой анализатор человека

Для сравнения с характеристиками зрительного анализатора человека приведем некоторые характеристики звукового анализатора, который реализуется с помощью слухового аппарата, включающего ушные раковины, барабанные перепонки и слуховые нервы.

2.3.1. Характеристики звукового анализатора

Звуковой анализатор человека реагирует на следующие три характеристики звукового сигнала:

1. Громкость A (пропорциональна амплитуде звукового сигнала);
2. Частота f (пропорциональна высоте тона звукового сигнала);
3. Фаза φ (тембр[3]).

3 Те́мбр (фр. *timbre*) — окраска звука; один из признаков музыкального звука (наряду с высотой, громкостью и длительностью). По тембрам отличают звуки одинаковой высоты и громкости, но исполненные или на разных инструментах, разными голосами, или на одном инструменте разными способами, штрихами.

Диапазон абсолютной чувствительности по частоте составляет от 16 Гц до 20 кГц.

Дифференциальный порог чувствительности (разрешающая способность) по частоте при интенсивности звука I_0=30 *дБ* составляет:

$$\Delta f = 2 \div 3 \text{ Гц}.$$

Относительная величина дифференциального порога на средних частотах (200÷1600Гц) составляет:

$$\frac{\Delta f}{f} = k = 0{,}002 .$$

Дифференциальный порог чувствительности по амплитуде (по громкости) составляет на средних же частотах (200÷1600Гц):

$$\frac{\Delta I}{I} = \frac{\Delta A}{A} = k = 0{,}1 .$$

Временной порог звуковой чувствительности на частоте *f=1000 Гц* и при амплитуде *A=30 дБ* составляет от 1 мс и более.

При уменьшении громкости временной порог звуковой чувствительности увеличивается.

2.5. Закон Вебера-Фехнера

В ряде экспериментов, начиная с 1834 года, Э. Вебер показал, что новый раздражитель, чтобы отличаться по ощущениям от предыдущего, должен отличаться от исходного раздражителя на величину, пропорциональную исходному раздражителю. Так, чтобы два предмета воспринимались как различные по весу, их вес должен различаться на 1/30, для различения яркости двух источников света необходимо, чтобы их яркость отличалась на 1/100 и т. д.

На основе этих наблюдений Г. Фехнер в 1860 году сформулировал «основной психофизический закон», по которому сила ощущения R пропорциональна логарифму интенсивности раздражителя I:

$$R = k \log \frac{I}{I_0},$$

где I_0 — граничное значение интенсивности раздражителя: если $I < I_0$, раздражитель совсем не ощущается (рис. 31).

Тембр определяется материалом, формой вибратора, условиями его колебаний, резонатором, акустикой помещения. В характеристике тембра большое значение имеют обертоны и их соотношение по высоте и громкости, шумовые призвуки, атака (начальный момент звука), форманты, вибрато и др. факторы.

При восприятии тембров обычно возникают различные ассоциации: тембровое качество звука сравнивают со зрительными, осязательными, вкусовыми и др. ощущениями от тех или иных предметов, явлений (звуки яркие, блестящие, матовые, тёплые, холодные, глубокие, полные, резкие, мягкие, насыщенные, сочные, металлические, стеклянные и т. п.); реже применяются собственно слуховые определения (звонкие, глухие). Научно-обоснованная типология тембра ещё не сложилась. Установлено, что тембровый слух имеет зонную природу.

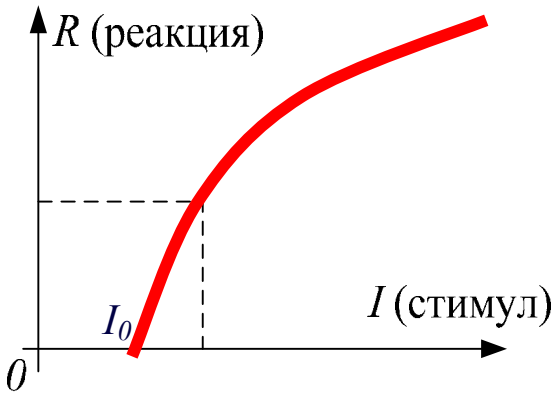


Рис. 31. К закону Вебера-Фехнера. Кривая изменения величины реакции в зависимости от величины стимула

Итак, закон Вебера–Фехнера – это эмпирический психофизиологический закон, который можно сформулировать следующим образом: *интенсивность ощущения пропорциональна логарифму интенсивности стимула*.

Так, люстра, в которой 8 лампочек, кажется нам настолько же ярче люстры из 4-х лампочек, насколько люстра из 4-х лампочек ярче люстры из 2-х лампочек. То есть, количество лампочек должно увеличиваться в разы, чтобы нам казалось, что прирост яркости постоянен. И наоборот, если прирост яркости постоянен, нам будет казаться, что он уменьшается. Например, если добавить одну лампочку к люстре из 12 лампочек, то мы практически не заметим прироста яркости. В то же время, одна лампочка, добавленная к люстре из двух лампочек, даёт значительный кажущийся прирост яркости.

Иначе закон Вебера–Фехнера можно сформулировать следующим образом. *Отношение приращения стимула к его абсолютной величине сохраняется постоянным*, т. е. для бо́льших значений абсолютного стимула нужен бо́льший порог различения (дифференциальный порог).

$$\frac{\Delta I}{I} = const$$

Закон Вебера–Фехнера можно объяснить тем, что константы скорости химических реакций проходящих при рецептировании нелинейно зависят от концентрации химических посредников физических раздражителей или собственно химических раздражителей.

Глава 3. Сведения из теории информации и инженерной психологии

3.1. Количество информации

Количество информации можно определить как меру неопределенности, снятую во время акта передачи информации.

Пусть $x_1, x_2, \ldots x_n$ – возможные события.

Р. Хартли в 1928 году предложил измерять количество информации с помощью энтропии H, которую следует вычислять по формуле:

$$H=log(m) ,$$

где m – количество возможных исходов.

Энтропия как раз и является мерой неопределенности: чем больше неопределенность (неизвестность точного результата), тем больше энтропия.

Клод Шеннон в 1948 г. предложил эту формулу модифицировать с учетом различных вероятностей $p_1, p_2, \ldots p_n$ возможных исходов:

$$H = -\sum_{i=1}^{m} p_i \log p_i \quad ,$$

где вероятности исходов подчиняются нормирующему правилу: $\sum_{i=1}^{m} p_i = 1$.

Количество информации – это разность между энтропией, которая была до акта передачи, и той энтропией, которая осталась после него:

$$J = H_1 - H_2$$

В том случае, когда после акта передачи информации неопределенность полностью снята, т.е. при $H_2=0$, количество информации численно равно исходной энтропии:

$$J=H_1.$$

3.1.1. Свойства количества информации

1. Энтропия является неотрицательной величиной: $J \geq 0$.
2. Энтропия заранее известного события равна нулю.
3. Количество информации одного объекта относительно другого не превышает энтропию любого из этих объектов.
4. Никакое преобразование объекта не увеличивает в нем количество информации о другом объекте.

3.1.2. Избыточность информации

Избыточность информации R:

$$R = \frac{J_{\max} - J}{J_{\max}} ,$$

J_{max} – максимальное количество информации, возможное для данного алфавита сообщений.

J – количество информации в данном сообщении.

Избыточность увеличивает помехозащищенность информации. В подтверждение этого утверждения рассмотрим пример.

Пример. Попытаемся прочитать фразу:

Озботочность онформоцоо оволочоворт вромо оо породочо.

Большинство сумеет восстановить правильное написание и прочитать ее, несмотря на то, что в этой фразе количество ошибок весьма велико, так как вместо любой гласной используется только одна – «о»:

Избыточность информации увеличивает время ее передачи.

Для естественных языков избыточность составляет приблизительно 60%. Для специальных языков значение избыточности может доходить до 95%.

3.1.3. Скорость создания информации

Источник сообщений характеризуется скоростью создания информации

$$G=nH_0$$

n – число, созданных источником сообщений

H_0 – энтропия сообщений.

3.1.4. Пропускная способность

Характеристикой канала связи является пропускная способность – максимальная скорость передачи сообщения по данному каналу.

$$G=max\{V\}$$

$$V=\frac{dJ}{dt}.$$

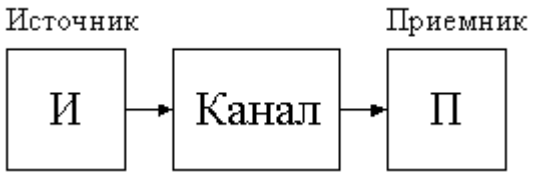


Рис. 32. Структура канала передачи информации

На рис. 32 показана структура канала передачи информации.

3.2. Факторы, влияющие на переработку информации человеком

1. Объективные характеристики сообщений и сигналов.
2. Условия работы человека-оператора.
3. Индивидуальные характеристики человека.

3.2.1. Объективные характеристики сообщений и сигналов

Сигналы как физические носители, содержащие информацию, оцениваются следующими количественными характеристиками.

1) Отношение «сигнал – шум» $h=P_{сигнала}/P_{шума}$ представляет собой отношение мощности полезного сигнала к мощности помехи.
2) Ширина частотного спектра сигнала F учитывает все частоты, формирующие сигнал, от наименьшей частоты до наибольшей.
3) Время существования сигнала T выражает длительность действия (экспозиции) сигнала.

Существует понятие объема сигнала как произведения всех трех вышеуказанных характеристик (рис. 33). Возможны преобразования сигналов, инвариантные к их объему. Изменение же объема приводит, как правило, к потере информации.

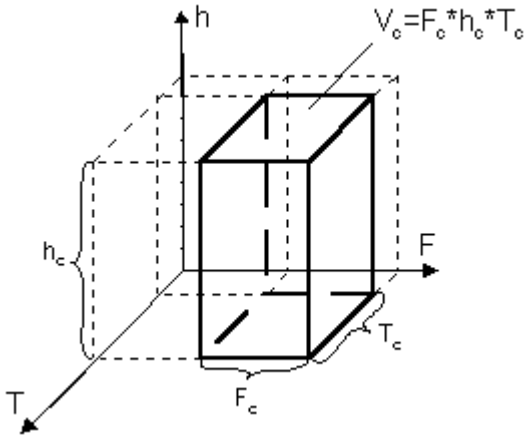


Рис. 33. К понятию объема сигнала

К *информационным* характеристикам сигналов относят алфавиты сообщений, распределение вероятностей их появления, статистические связи, избыточность сигналов, значимость сигналов, соотношение релевантной (полезной, важной) и иррелевантной (не важной, не полезной) информации.

Очевидно, что объективные характеристики сигналов и сообщений в значительной степени оказывают влияние на результат деятельности человека-оператора: так, например, при низком значении отношения «сигнал – шум» h_c человек хуже распознает такой зашумленный сигнал, нежели при большом значении этого отношения.

3.2.2. Условия работы человека-оператора

Сюда входят статические и динамические характеристики полей информационной модальности, соотношение величины информационного потока и пропускной способности человека. Кроме того влияние на условия работы оказывают следующие факторы:

1) совместимость стимула и реакции;
2) особенности деятельности оператора;

3) особенности информационного процесса с точки зрения психики (запоминание, распознавание, перекодирование);
4) мотивация деятельности;
5) воздействие на оператора физических и иных возмущений.

К иным возмущениям можно отнести, например, психологическое воздействие на человека, которое по силе воздействия иногда может даже превышать физические возмущения.

3.2.3. Индивидуальные характеристики человека-оператора

К индивидуальным характеристикам человека-оператора относят: пол, возраст, характеристики высшей нервной деятельности человека, обученность, тренированность, способности опыт, состояние, соотношение психической нормы и патологии, особенности физического, биологического и психологического характера. Индивидуальные различия играют тем большую роль, чем сложнее характер деятельности. Важную роль играет состояние и тренированность.

3.3. Применение теории информации в инженерной психологии

Некоторые количественные оценки используются для определения пропускной способности оператора, а также для определения характеристик восприятия человеком информации, переработки ее и хранении.

При информационной оценке сообщений в виде символов необходимо знать алфавиты сообщений и статические характеристики связей между символами, если таковые имеются.

3.3.1. Информационные оценки восприятия и памяти

Кратковременная память

Человек может безошибочно различать не более:
- 4-х концентраций растворов
- 6-и тонов звуков (по частоте)
- 5-и уровней интенсивности звука
- от 10 до 15 позиций между двумя отметками на линейной шкале.

Число различных состояний объекта возрастает при увеличении числа измерений.

Например, при 6-и различных акустических переменных число различных категорий звука примерно равно 150 элементов. Экспериментальные исследования кратковременной памяти человека, показали, что человек способен удерживать не более 9-и двоичных цифр (9 бит), 9-и десятичных (25 бит) или 7-и букв алфавита (33 бита) или 5-и односложных слов(50 бит).

Емкость кратковременной памяти измеряется не количеством информации, а числом информационных единиц. Следовательно, при кодировании информации, предназначенной для использования человеком, следует использовать наиболее информационно-емкие символы.

Информационная емкость символов увеличивается с возрастанием длины алфавита. Алфавит должен быть твердо усвоен оператором. Следовательно, надо применять либо знаковые алфавиты (буквы и цифры), либо нужно обучать оператора новому алфавиту.

Наличие связей между единицами информации уменьшает количество информации в последовательности символов и при этом лучше запоминается.

Долговременная память

Объем зависит в основном от информационного содержания, нежели от числа единиц информации. Этот объем измеряется количеством информации, которая усваивается в среднем при одном предъявлении запоминаемого материала и составляет от 5 до 20 бит. Иррелевантная информация является для долговременной памяти помехой.

Избыточность релевантной информации способствует долговременному запоминанию.

Максимальная скорость переработки информации человеком оценивается по-разному при однократном и непрерывном предъявлении информации.

При однократном предъявлении оценивается время реакции человека, в зависимости от стимула, по одной из следующих схем:

1) стимул – ответная моторная реакция;
2) стимул – отсроченный словесный отчет.

Схема №1.

В экспериментах, соответствующих первой схеме установлена линейная зависимость между скоростью реакции и количеством информации в стимуле. Иррелевантная информация увеличивает время обнаружения полезного сигнала. Тренировка позволяет уменьшить это время. При запоминании временных интервалов человеком существует неопределенность, которая связана с несовершенством памяти человека. Эта неопределенность тем сильнее, чем больше интервал времени между сигналами.

Другой фактор – вариативность времени появления сигнала (изменение интервала).

Чем больше вариативность интервала между стимулами, тем больше время простой сенсорно-моторной реакции.

Схема №2.

Она характеризует не столько скорость обработки информации, сколько объем поля восприятия. Объем поля зрительного восприятия в экспериментах

наиболее постоянен и примерно равен 4-5 объектам независимо от их информационного содержания.

При непрерывном поступлении информации человек может рассматриваться как канал связи. При этом нужно стремиться, чтобы этот канал удовлетворял свойству стационарности (не менял своих характеристик во времени). Для этого нужно иметь отчетливый источник информации, хорошо обученный оператор, нормальный условия работы и т.д. При этом под пропускной способностью человека-оператора понимают максимальную скорость безошибочной переработки информации.

3.3.2. Модели работы человека-оператора как канала связи

В теории информации рассматриваются каналы с памятью и без памяти. Работу человека-оператора можно рассматривать, используя одну из следующих 3-х моделей:

1) человек как канал связи без памяти;
2) человек как канал с кратковременной памятью;
3) человек как канал с долговременной памятью.

В первом случае человек работает как канал передачи информации, последовательно передающий сигналы, которые считаются независимыми друг от друга. При этом пропускная способность лежит в пределах от 10 до 70 бит/с. Конкретное значение сильно зависит от вида деятельности.

В табл. 3 представлены экспериментально полученные данные по пропускной способности человека (бит/с) при различных видах деятельности.

Таблица 3.

1	Чтение текста про себя	45
2	Чтение вслух	30
3	Игра на фортепьяно	23
4	Работа корректора	18
5	Печатание на машинке	15
6	Умножение двух цифр	12
7	Сложение двух цифр	8
8	Счет предметов	3

Для сравнения: скорость обмена информацией у муравьев составляет около 0,1 бит/сек.

Если в процессе деятельности человеку нужно запомнить отрезок входной последовательности сигналов по объему, не превышающего объема его кратковременной памяти, то его деятельность можно рассматривать как работу канала с кратковременной памятью. Если же отрезок информации, который необходимо запомнить превосходит объем кратковременной памяти, то для его запоминания необходимо многократное повторение.

Следовательно, пропускная способность падает до десятых долей бита в секунду и ниже. Для увеличения пропускной способности такого канала можно уменьшать длину входной последовательности, или уменьшать нагрузку на память.

3.4. Способы борьбы с избытком и недостатком информации

Многие технические устройства плохо согласованы с органами чувств человека. Так, например, телевизионные каналы имеют пропускную способность в десятки миллионов бит/сек. Информационные перегрузки перевозбуждают нервную систему человека, вызывают стрессы.

Существуют как объективные, так и субъективные способы решения этих проблем. Для устранения *информационных перегрузок* используют следующие способы.

Объективные способы.

1) Оптимальный для человека код;
2) Уменьшение избыточности информации (не в ущерб надежности);
3) Организуют параллельные потоки информации;
4) Организуют резервные системы.

Субъективные способы.

Человек самостоятельно борется с информационными перегрузками (чаще всего автоматически, спонтанно), прибегая к следующим действиям.

1) Пропуск информации
2) Ошибки
3) Фильтрация
4) Загрубление
5) Избегание
6) Отказ

С другой стороны недостаток информации также оказывает негативное воздействие на человека. Психологи установили, что всего 15 минут пребывания в сенсорной депривации (то есть при полном отсутствии внешних раздражителей) достаточно для того, чтобы у людей начались галлюцинации. Свои результаты ученые опубликовали в журнале *Journal of Nervous and Mental Disease*, а краткое изложение работы приводит издание *Wired*.

В рамках исследования ученые помещали по очереди 19 добровольцев в темную звукоизолированную комнату. Все участники эксперимента прошли предварительный отбор, целью которого было проанализировать склонность добровольцев к галлюцинациям. В результате были отобраны люди наименее и наиболее предрасположенные к подобным эпизодам.

Оказалось, что 15 минут достаточно для того, чтобы совершенно здоровые и не склонные к галлюцинациям люди начинали видеть вещи, которых нет в действительности. Так, из всей группы пятеро участников видели лица, а шестеро – непонятные формы. Некоторые участники отмечали повышенную обостренность обоняния, а двое почувствовали присутствие в комнате «чего-то ужасного».

Ученые полагают, что причиной возникновения подобных временных расстройств является отсутствие поступления привычного уровня информации в человеческий мозг. На это, например, указывает обостренное чувство обоняния.

По словам ученых, на опыт их вдохновили результаты американского психолога Джона Лилли (*John Lilly*), который в 50-х годах прошлого века изобрел так называемую сенсорно-депривационную камеру, представляющую собой темный звукоизолированный бак.

Для борьбы с «*информационным голодом*» используют следующие меры:

1) устанавливают дополнительные источники информации;
2) применяют меры к устранению монотонности поступления информации.

3.5. Оценка полезности информации

О полезности судят по тому, насколько информация служит достижению цели. Общих методов оценки полезности информации не существует. Большинство способов оценки основано на вероятностном подходе. Например, можно использовать сравнение вероятностей достижения цели до получения информации и после него. Если вероятность достижения цели после получения информации окажется выше, то информацию следует признать полезной.

Глава 4. Компьютерная графика как инструмент проектирования интерфейса

4.1. Общая характеристика компьютерной графики

4.1.1. От наскальных рисунков – к компьютерной анимации

Рис. 34. Сальвадор Дали ... в 3D Studio MAX

С доисторических времен и до сего дня человек в силу своего естественного строения значительно лучше (надежнее, быстрее) воспринимает графические образы, нежели таблицы, формулы, текст. В будущем, при нынешних темпах увеличения количества информации, которую вынужден перерабатывать человек, значение графики будет только возрастать, а возможно станет для него единственной альтернативой в условиях информационного взрыва.

Сегодня человеку предстоит развить свои естественные способности, заложенные природой, вспомнить и усовершенствовать международный язык графических изображений, широко используемый в искусстве, архитектуре, а также и в таких прагматических областях как дорожное движение, реклама и т.д. Наука и техника, экономика и экология, производство и социальная сфера –

практически ни одна область человеческой деятельности не обходится сегодня без графических объектов, созданных преимущественно компьютерными средствами.

Современные компьютерные технологии позволяют получать изображения чрезвычайно высокого качества и, используя разнообразные удивительные эффекты, перемещать и трансформировать их по своему желанию любым, самым фантастическим образом. Существуют уникальные компьютерные программы, которые используются в фильмах наряду с «живой» видеосъемкой.

4.1.2. Классификация проблем, связанных с графическими изображениями

Обработка информации, связанная с графическими изображениями на экране монитора или на бумаге, подразделяется на 3 основные вида: распознавание образов (РО), обработка изображений (ОИ) и компьютерная графика (КГ) (рис. 35).

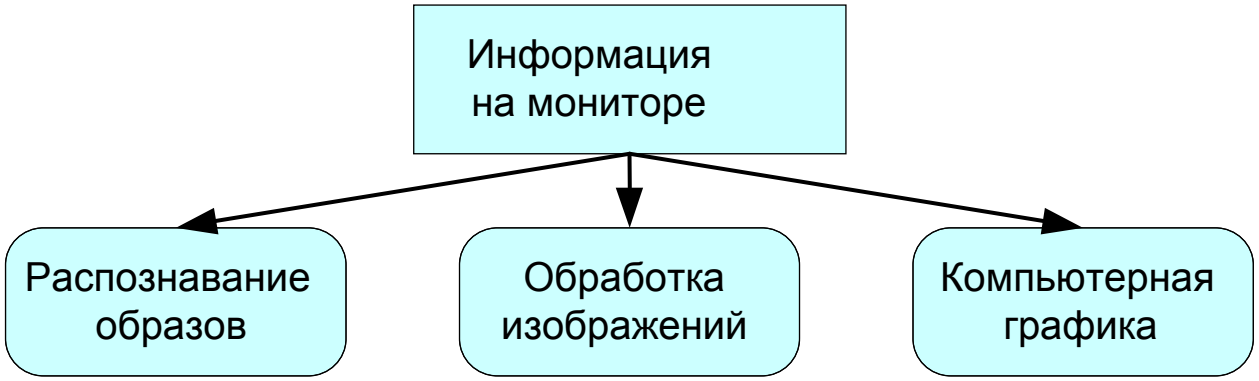


Рис 35. Схема представления работы с изображением

Распознавание образов (изображений) есть совокупность методов, позволяющих получить описание изображения, поданного на вход, либо отнести заданное изображение к некоторому классу (так поступают, например, при сортировке почты или в медицинской диагностике, где путем анализа томограмм оценивают наличие отклонений от нормы). При этом рассматриваемое изображение часто преобразуется в более абстрактное описание набор чисел, набор символов или граф. Одной из интересных задач распознавания образов является так называемая скелетизация объектов, при которой восстанавливается некая основа объекта, его «скелет».

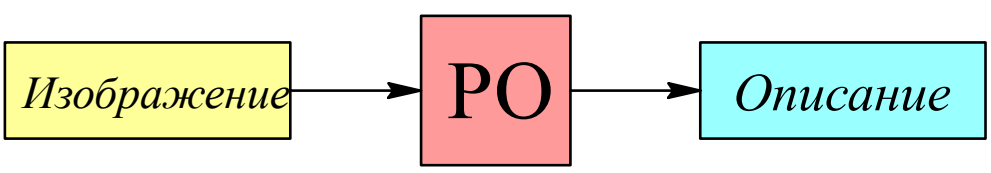


Рис. 36. Схема представления распознавания образов

Символически, распознавание изображений, или система технического зрения (*Computer Vision)*, может быть описана так (рис. 36):
• на входе – изображение;
• на выходе – описание этого изображения.

Обработка изображений рассматривает задачи, в которых и входные и выходные данные являются изображениями (рис. 37). Примерами обработки изображений могут служить: передача изображений вместе с устранением шумов и сжатием данных, переход от одного вида изображения (полутонового) к другому (каркасному), контрастирование различных снимков, а также синтезирование имеющихся изображений в новые, например, по набору поперечных сечений объекта построить продольные сечения или восстановить сам объект.

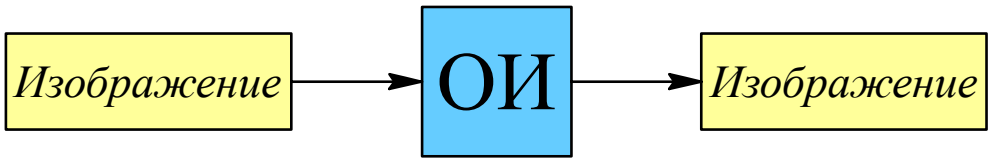
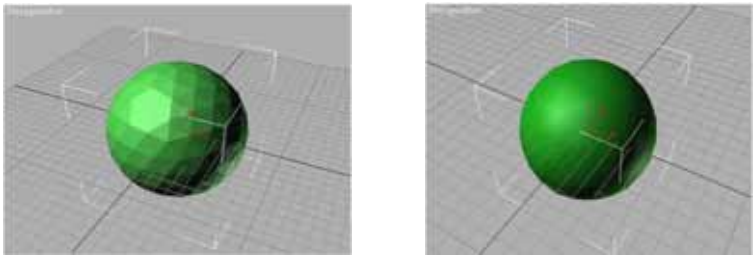


Рис 37. Схема представления обработки изображения

Пример (рис. 38). На входе изображение сферы (а), а на выходе – изображение той же сферы, но преобразованное с помощью другой модели (б).



а) б)

Рис. 38. Различные изображения сферы

Таким образом, обработка изображений (*Image Processing)* имеет следующую структуру:
• на входе – изображение;
• на выходе – изображение (преобразованное).

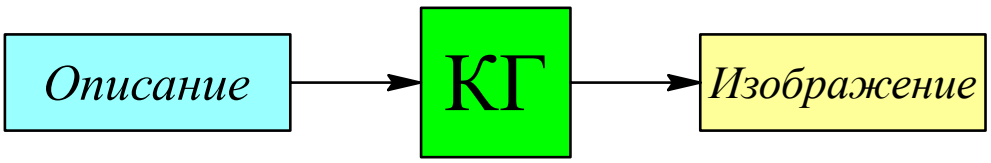


Рис 39. Схема представления компьютерной графики

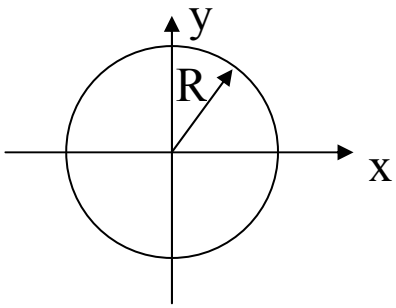
Компьютерная (машинная) графика воспроизводит изображение в случае, когда исходной является информация неизобразительной природы (рис. 39). Например, визуализация экспериментальных данных в виде графиков,

гистограмм или диаграмм, вывод информации на экран в компьютерных играх, синтез сцен для тренажеров. А еще есть компьютерная живопись, компьютерная анимация и так далее, вплоть до виртуальной реальности.

Можно сказать, что компьютерная графика рисует, опираясь на формальные правила и имеющийся набор средств.

Пример (рис. 40). На входе уравнение окружности с указанием ее радиуса и координат центра (а), а на выходе – изображение этой окружности (б).

$$X^2 + Y^2 = R^2$$



а) б)

Рис. 40. Входной и выходной сигналы системы компьютерной графики

Символически компьютерную графику (*Computer Graphics*) можно представить следующим образом:
на входе – символьное описание;
на выходе – изображение (синтезированное изображение).

Удобной для запоминания представляется общая схема (рис. 41), вмещающая в себя описание и функции всех трех задач: РО, ОИ и КГ.

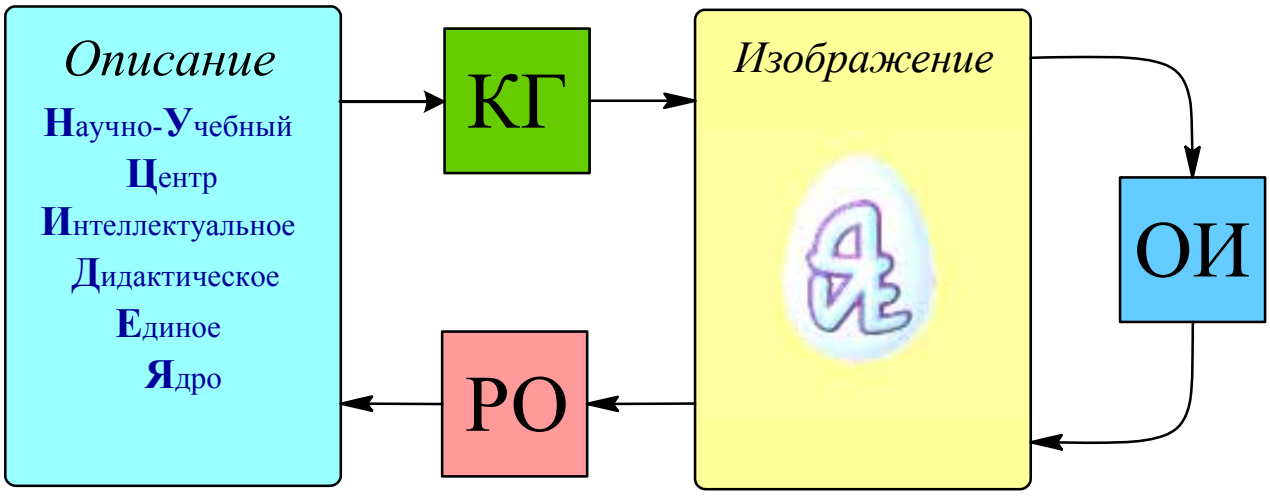


Рис 41. Схема представления взаимодействия описания изображения и самого изображения по средствам КГ, РО, и ОИ

4.1.3. Направления развития и улучшения компьютерной графики

Существуют следующие основные направления развития компьютерной графики (рис. 42):

1) иллюстративное,
2) саморазвивающее,
3) исследовательское.

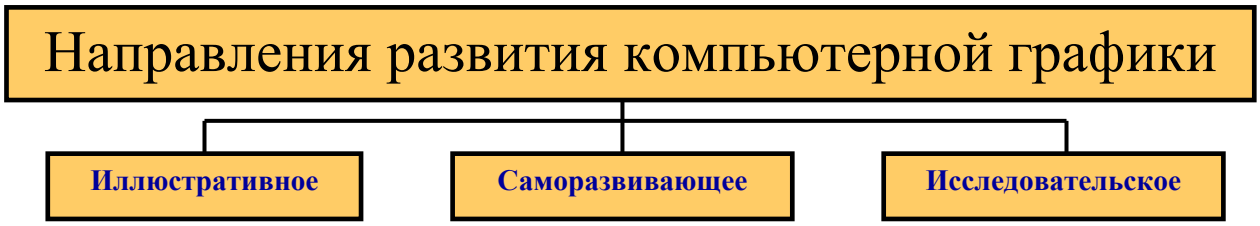


Рис. 42. Направления развития компьютерной графики

Первое направление включает очень широкий спектр средств, от простейших изображений, понятий, результатов действий, до сложных, таких как рекламные ролики, живая видеосъёмка, спецэффекты, комбинации и т. д.

Второе направление связано с обслуживанием собственных потребностей компьютерной графики и служит для расширения её возможностей. Это связано с компьютерным оборудованием, программным обеспечением.

Третье направление касается того, в чём компьютерная графика может служить инструментом для исследований, позволяя выявлять некоторые особенности, которые без неё не были доступны.

Компьютерная графика совершенствуется с каждым годом в различных отношениях, однако, основными остаются следующие направления (рис. 43):

1) улучшение динамики изображений;
2) улучшение реалистичности изображений.

Рис. 43. Направления совершенствования компьютерной графики

Улучшение, как динамики, так и реалистичности требует значительных компьютерных ресурсов: быстродействия процессоров и памяти (в первую очередь оперативной и кэш-памяти).

Развитие компьютерной графики создало новый изобразительный инструмент, который используется художниками, дизайнерами, архитекторами, скульпторами и т. д. Кроме того, существует целый класс профессиональных программных пакетов, которые объединены общей аббревиатурой *CAD* – *Computer Aidded Design* (проектирование с помощью компьютера). Компьютерная графика позволяет облегчить работу человека, освобождая его от рутинных операций ради творческих задач.

4.2. Разновидности компьютерной графики

Графические изображения по принципу действия и функциональному назначению можно разделить на три группы:
1. Растровая графика (*bitmap*, или *raster*);
2. Векторная графика (*vector*, или *draw*);
3. Фрактальная графика (*fractal*).

Наиболее широко в компьютерной графике представлены первые два типа графики: растровая и векторная. Важно понимать принципиальные различия между двумя этими типами графики, так как каждый из них имеет свои сильные и слабые стороны и свою сферу применения.

4.2.1. Растровая графика

Растровая графика представляет изображение в виде совокупности точек, различающихся по цвету. При большом увеличении все точечные изображения выглядят как мозаика (сетка), состоящая из мельчайших ячеек. Сама сетка получила название растровой карты (*bitmap*), а ее единичный элемент называется пикселем (от англ. *picture element* – элемент изображения). Каждый из пикселей окрашивается в определенный цвет из имеющейся палитры. Простейший вид окрашивания – бинарный (всего 2 цвета – чёрный или белый) требует всего одного бита памяти.

По способу отображения информации растровое изображение напоминает мозаичное панно. Если вы рассматриваете мозаичное изображение вблизи (при сильном увеличении), то стыки между отдельными деталями хорошо видны; если же вы отойдете подальше (что равносильно уменьшению масштаба), то мелкие элементы сливаются, и вы видите картину слитно, в целостном восприятии. Так же и с растровой графикой: если пиксели достаточно малы, то границы между ними незаметны и глаз воспринимает «пиксельную мозаику» (растр) как одно целое изображение (рис. 44).

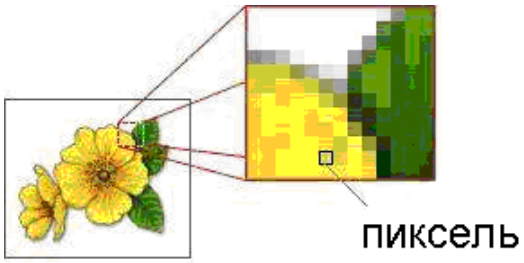


Рис. 44. Пример растрового изображения при увеличении его масштаба

При масштабировании растровых изображений возникают характерные искажения – «ступеньки» (*aliasing,* или *jaggies*). В большинстве растровых редакторов «ступеньки» удается частично убрать за счет специальных приемов (например, *anti-aliasing*), но качество картинки от этого заметно снижается.

Рис. 45. Пример растрового изображения

Недостатками растровой графики являются:

1) неудобство трансформации растровых объектов (при поворотах пиксели становятся явно видимыми, а при увеличении масштаба изображений пиксели также пропорционально увеличиваются и тоже «вылезают» – становятся отчетливо видимыми);
2) большие затраты памяти, к тому же увеличивающиеся с улучшением качества изображений;
3) сложность редактирования изображений;

Достоинства растровой графики вполне очевидны:

1) высокая степень фотореалистичности изображений;
2) возможность автоматического ввода изображений в компьютер (например, с помощью сканера);
3) совместимость с текстом, другими объектами;
4) высокое быстродействие (на экран выводится все пиксели изображения одновременно, т.е. вся картинка целиком).

4.2.2. Векторная графика

Изображение, созданное в векторных программах, основывается на математических формулах, а не на координатах пикселей и информации об их цвете. Поэтому векторные файлы содержат наборы инструкций для построения геометрических объектов – линий, эллипсов, прямоугольников,

многоугольников и дуг. В соответствии с этим основу векторных изображений составляют разнообразные линии и кривые, называемые векторными, или, по-другому, контурами. Каждый контур представляет собой независимый объект, который можно редактировать: перемещать, изменять, масштабировать. При этом качество изображения не изменяется (рис. 46), так как после внесения каких-либо изменений в файл происходит пересчет размера и места положения объекта изображения (т.е. при каждом выводе изображения происходит новый расчет изображения по математическим формулам, следовательно, образование «ступенек», как в растровой графике, невозможно).

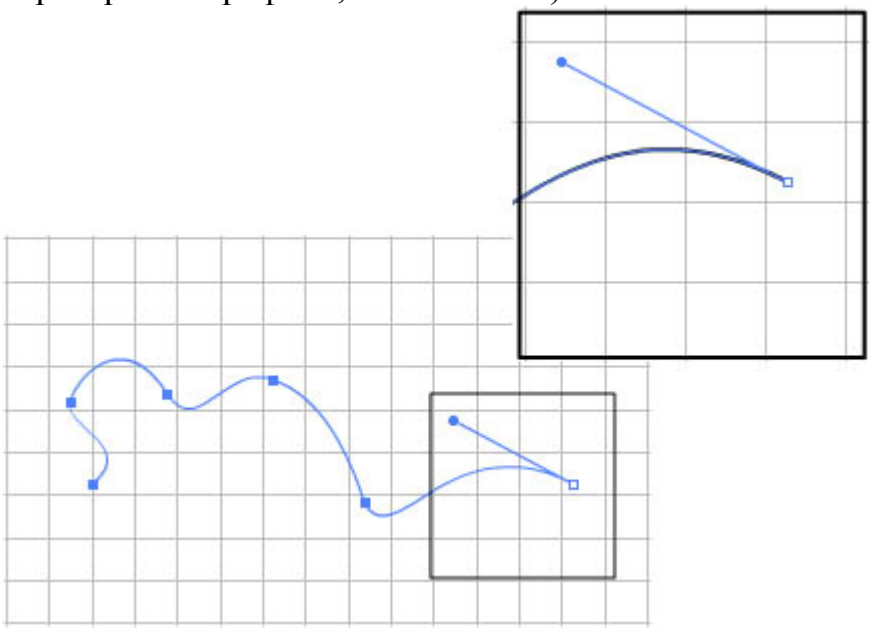


Рис. 46. Пример изменения масштаба векторного изображения

К недостаткам векторной графики относятся:

1) невозможность непосредственного автоматического ввода векторных графических изображений в память компьютера;
2) высокие требования к быстродействию видеопроцессора (прежде чем выводить изображение, на каждом шаге нужно просчитывать по формулам координаты выводимых точек);
3) последовательный вывод на экран (отрисовывание) элементов изображения;

Достоинствами векторной графики являются значительная экономия памяти и высокая четкость изображения, сохраняющаяся при любых преобразованиях.

Векторные изображения могут быть созданы с помощью нескольких видов программ. Это, в частности:

1) редакторы векторной графики, например, *Visio*;
2) программы САПР, например, *AutoCAD, ArchiCAD, OrgCAD*;
3) специальные программы конвертирования растровых изображений в векторные, например, *CorelTrace9*, входящая в состав графического редактора *CorelDraw*.

4.2.3. Фрактальная графика

Понятие фрактал, фрактальная геометрия и фрактальная графика, появившиеся в конце 70-х, сегодня прочно вошли в обиход математиков и компьютерных художников. Слово фрактал образовано от латинского *fractus* и в переводе означает «состоящий из фрагментов». Оно было предложено математиком Бенуа Мандельбротом в 1975 году для обозначения нерегулярных, но самоподобных структур. Объект называется самоподобным, когда увеличенные части объекта походят на сам объект и друг на друга. Определение фрактала, данное Мандельбротом: фракталом называется структура, состоящая из частей, которые в каком-то смысле подобны целому.

К фрактальным множествам относят множество Кантора и ковер Серпинского. Они обладают геометрической инвариантностью и называются «множествами средних третей». Простейший вариант множества Кантора строится следующим образом.

Рассмотрим отрезок единичной длины на вещественной оси. Этот отрезок делится на три равные части, причем средняя часть вырезается (рис. 47).

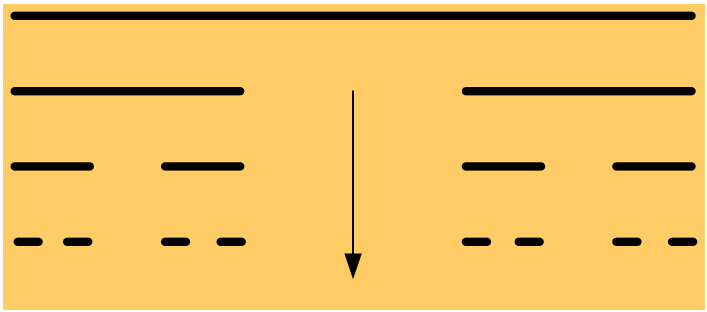


Рис. 47. Построение множества Кантора

Далее с каждым из оставшихся отрезков происходит то же самое: они делятся на три равные части, и средние части удаляются. В результате получается последовательность отрезков убывающей длины. На первом этапе это один отрезок, на втором – два, на третьем – четыре и т.д., на k–м – $2k$. При $k \to \infty$ получим множество точек, называемое множеством Кантора. Суммарная длина всех вырезанных отрезков равна 1.

Множество Кантора является фракталом. Для фрактала введено понятие фрактальной размерности. Для множества Кантора, которое состоит из $N = 2n$, разделенных интервалов длиной $\varepsilon = (1/3)n$, фрактальная размерность равна:

$$ФР = n\ ln2/n\ ln3 \approx 0,63$$

Обобщение множества Кантора средних третей на случай плоских фигур приводит к ковру Серпинского. Например, возьмем квадратную матрицу и разделим ее на девять равных квадратов. При первой итерации удалим центральный квадрат, аналогично поступим с каждым из оставшихся восьми квадратов и т.д. Пересечение полученных при $k \to \infty$ множеств – это ковер Серпинского (рис. 48). Для ковра Серпинского фрактальная размерность равна $ФР \approx 1,893$.

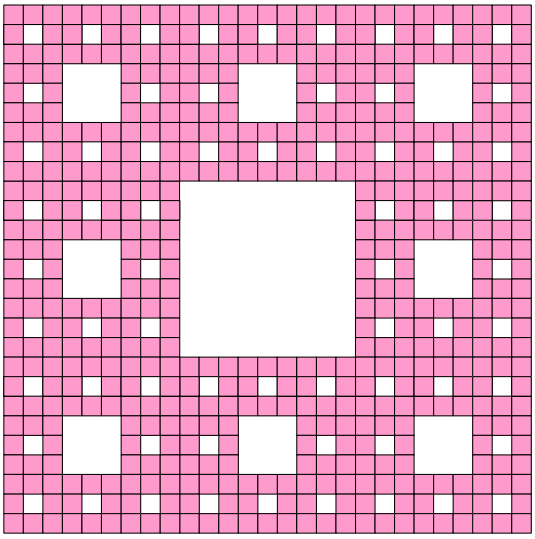
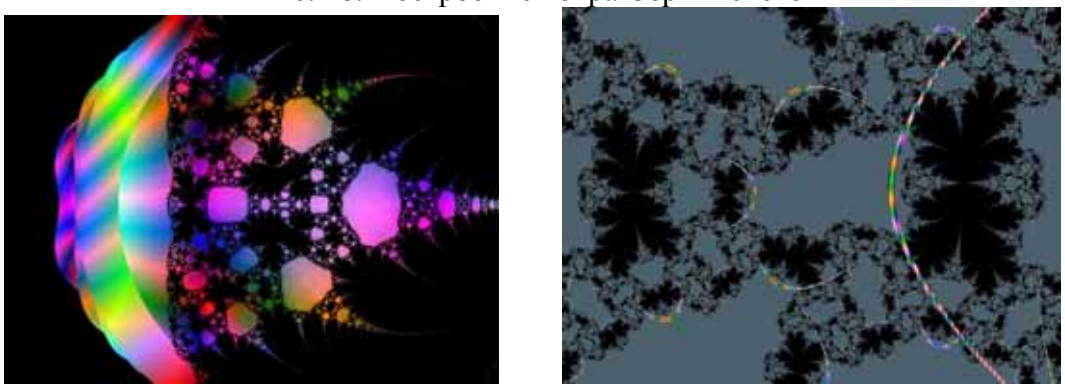


Рис. 48. Построение ковра Серпинского



а) б)

Рис. 49. Примеры 2-х мерных фрактальных изображений: а) «Клоп», б) «Лиана»

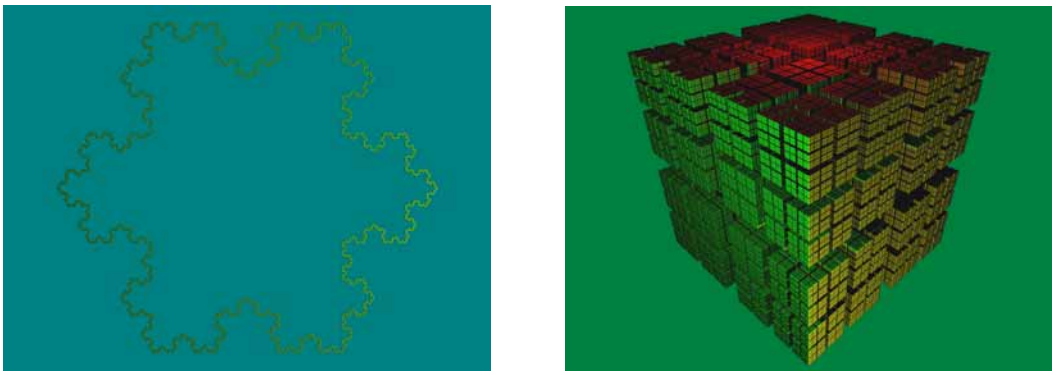


Рис. 50. Примеры 3-х мерных фракталов: а) Коха, б) «Куб»

Другие примеры фрактальных изображений показаны на рис. 49 и 50.

К достоинствам фрактальной графики следует отнести то, что фракталы являются одной из лучших моделей живой природы, а кроме того, исследование фракталов открывает невиданные перспективы для сжатия информации.

Недостатком фрактальной графики является достаточно узкая область ее применения.

4.2.4. Цветовые модели и режимы

Графическое изображение может быть представлено в разных видах, в частности – в различных цветовых моделях. Человеческий глаз фиксирует электромагнитные волны длиной от 380 до 760 нм как различные цвета – от красного до фиолетового. В видимом спектре семь цветов, но глаз не физический прибор, его можно «обмануть». Смесь синего и красного цветов он воспринимает как фиолетовый, а смесь синего и желтого – как зеленый. При представлении почти любого цвета или оттенка можно обойтись всего тремя цветами, что технически значительно проще. Способ отображения цветового пространства с помощью набора базовых цветов называют цветовой моделью. Обычно в качестве основных (базовых) цветов используют красный (*Red*), зеленый (*Green*) и синий (*Blue*) цвета. Этот способ представления называют цветовой моделью *RGB*. Но это не единственно возможная цветовая модель, существуют и другие.

Цветовая модель ***RGB*** используется во многих видеоустройствах: мониторах, сканерах, цифровых фотокамерах и др. Нужный цвет получается как сумма (*addition*) базовых цветов (***Red***, ***Green***, ***Blue***) (рис. 51).

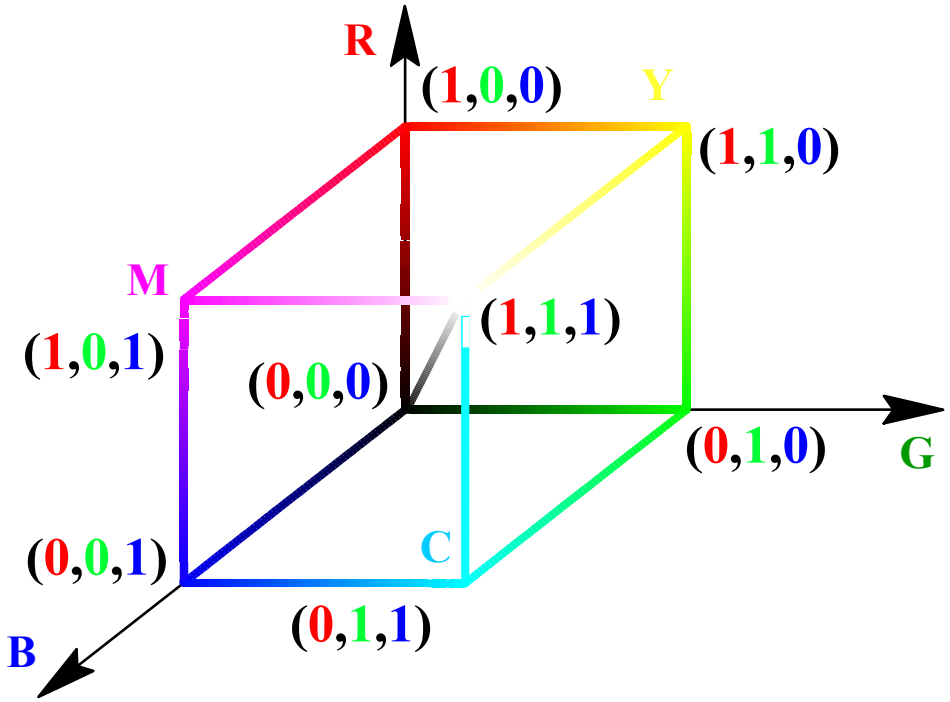


Рис. 51. Графическое представление *аддитивной* цветовой модели *RGB*

Цветовая модель ***CMY*** (система основных цветов: голубой, малиновый, желтый) используется в цветной печати. Нужный цвет получается как разность (*subtraction*) белого и базовых цветов (***Cyan***, ***Magenta***, ***Yellow***) (рис. 52):

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

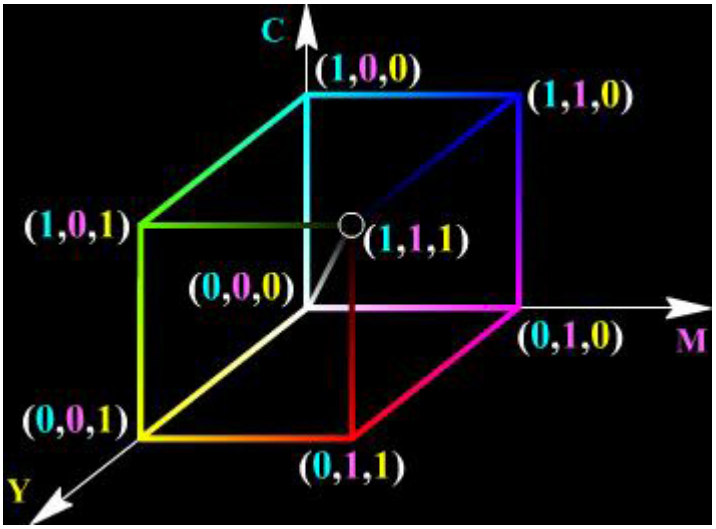


Рис. 52. Графическое представление *субтрактивной* цветовой модели *CMYK*

От модели *CMY* можно перейти к модели *RGB*, вычитая из белого:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} C \\ M \\ Y \end{bmatrix}.$$

Модель ***CMYK*** используется в цветной печати из практических соображений: чтобы избежать расточительного расхода цветных красителей для получения черного цвета этот черный цвет вводится отдельно и добавляется к трем базовым цветам, и получается ***CMYK*** (***Cyan***, ***Magenta***, ***Yellow***, ***blacK***).

Модели *RGB*, *CMY*, *CMYK* ориентированы на работу с цветопередающей аппаратурой, а для задания цвета человеком неудобны. С другой стороны, модель *HSV* (*Hue, Saturation, Value – Оттенок, Насыщенность, Значение*), иногда называемая *HSB* (*Hue, Saturation, Brightness – Оттенок, Насыщенность, Яркость*), больше ориентирована на работу с человеком и позволяет задавать цвета, опираясь на интуитивные понятия тона, насыщенности и яркости. Данная модель представлена на рис. 53 в каркасном виде, на ней отражены основные направления цветопереходов.

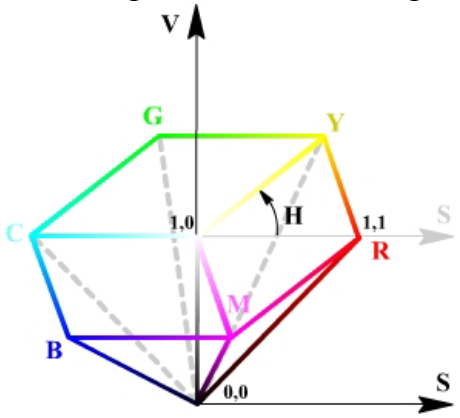


Рис. 53. Схематическое представление графической модели *HSV* (*HSB*)

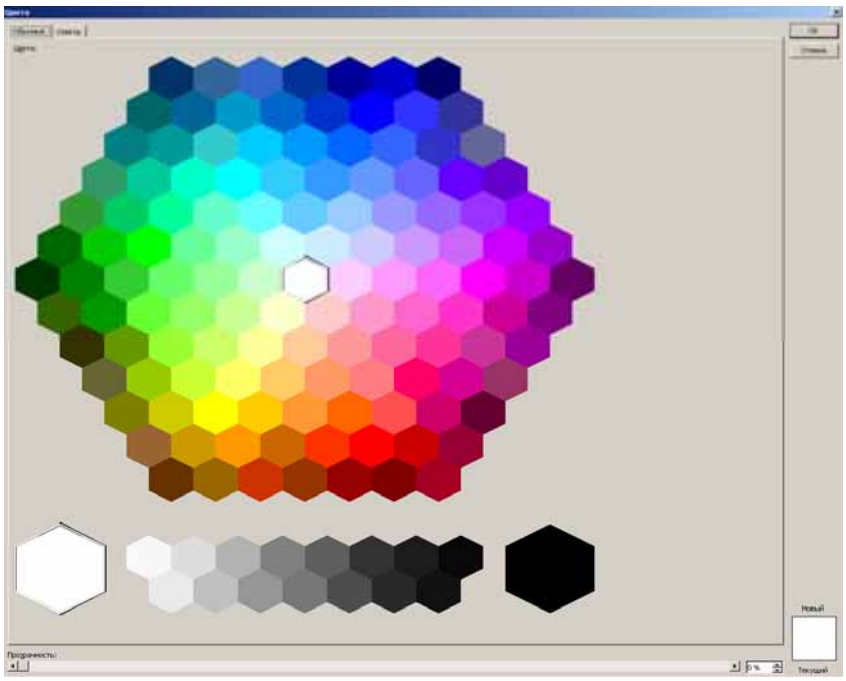


Рис. 54. Представление цветовой палитры в графическом редакторе с помощью модели *HSV*

4.2.5. Видеокарты

Для реализации вывода изображения на монитор в состав компьютера входит видеокарта, или видеоадаптер. Она определяет следующие показатели:
1) скорость обработки информации;
2) четкость изображения;
3) цветность рабочего поля экрана (поддерживаемые цветовые режимы).

Основные функции видеокарты:
а) хранение изображения, для этого используется видеопамять; на каждый пиксель отводится определённое количество бит для хранения основных цветов;
б) регулярный вывод изображения на экран монитора с определённой частотой.
Каждый из видеоадаптеров поддерживает несколько режимов работы, отличающихся размерами матрицы пикселей (разрешением) и цветовой палитры.

Растровые изображения требуют очень много памяти, это еще один недостаток данного типа компьютерной графики. Рассчитать количество необходимой памяти под размещение изображения можно следующим образом:

$$M = h \cdot w \cdot c,$$

где M (*memory*) – количество памяти под изображение;
h (*height*) – количество пикселей по вертикали;
w (*width*) – количество пикселей по горизонтали;
c (*colour*) – цветовой режим.

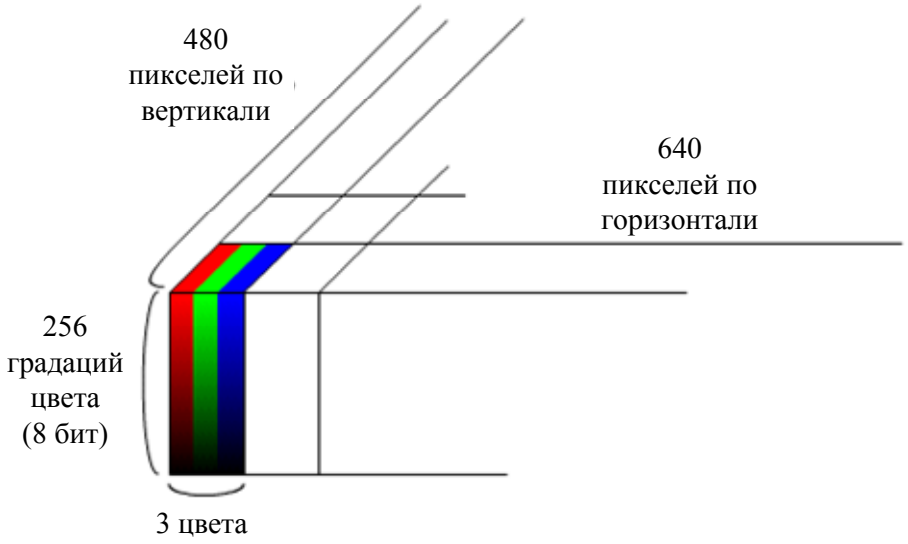


Рис. 55. Графическое представление видеопамяти компьютера

Пример. Если рассматривать растровое изображение размером 640x480 пикселей в цветовом режиме *RGB Color* 24 *bit*, то количество необходимой памяти для хранения этого изображения рассчитывается следующим образом (рис. 55):

h = 480 (количество пикселей по вертикали);

w = 640 (количество пикселей по горизонтали);

$c = 256 * 256 * 256 = 2^8 * 2^8 * 2^8 = 2^{8*3} = 2^{24}$ (по 8 бит на каждый из основных цветов в палитре);

M = 480 * 640 * 24 бит = 7372800 бит = 900 Кбайт.

Достоинствами растровой графики являются высокая степень фотореалистичности и возможность автоматического ввода в компьютер (например, с помощью сканера или цифрового фотоаппарата).

Примеры цветовых режимов, применяемых при использовании растровой графики:

· Черно-белый (*Black and White*), 1-*bit*;
· Градации серого (*Grayscale*), 8-*bit* или 16-*bit*;
· Дуплекс (*Duotone*), 8-*bit*;
· Палитра (*Paletted*), 8-*bit*;
· *RGB True Color*, 24-*bit*;
· *CMYK Color, 32-bit.*

4.2.5. Форматы графических файлов

Изображения сохраняются в памяти в файлах определенного конкретного вида, называемого форматом графического файла. Существует довольно много различных графических файлов, отличающихся теми или иными особенностями. Здесь приведем краткие характеристики лишь нескольких наиболее используемых форматов.

Формат BMP *(BitMaP)* создан фирмой *Microsoft* и является одним из первых графических растровых форматов. Его распознает любая программа, работающая с графикой, поддержка формата интегрирована в операционные системы *Windows* и *OS/2*.

BMP используется для представления растровых изображений в модели *RGB* с глубиной цвета до 48 бит и максимальным размером 65535x65535 пикселей. В принципе, формат предполагает использование простейшего алгоритма сжатия *RLE* (*Run Length Encoding*) без потерь информации.

На данный момент формат BMP практически не используется ни в интернете (*JPG* весит в разы меньше), ни в полиграфии (*TIFF* справляется с этой задачей лучше).

Формат TIFF *(Tagged Image File Format)* создан фирмой *Aldus* специально для хранения сканированных изображений. Это формат, максимально точно передающий изначальное изображение. Он не компрессирует файл, поэтому файл занимает много места на жестком диске. Хотя с момента его создания прошло уже много времени, до сих пор является основным форматом, используемым для хранения сканированных изображений и размещения их в издательских системах и программах иллюстрирования. Версии формата существуют на всех компьютерных платформах, что делает его исключительно удобным для переноса растровых изображений между ними. *TIFF* поддерживает монохромные, индексированные, полутоновые и полноцветные изображения *в моделях RGB и CMYK с 8- и 16-битными каналами*. Файлы этого типа — стандарт в полиграфии, где требуется четкая картинка с максимальным разрешением. При сохранении файла в формате *TIFF* с использованием схемы сжатия *LZW (Lempel-Ziv-Welch)* его размеры уменьшаются настолько, насколько это возможно. Данная схема сжимает файл без потери данных, то есть в результате ее применения качество изображения не ухудшается.

Формат JPEG *(Joint Photographic Experts Group). JPEG* (*JPG)* – это формат, позволяющий создать минимальный по размерам файл с наименьшей потерей качества изображения. Поэтому этот формат получил большую популярность — он является стандартом для хранения изображений в Интернете и многих цифровых фотоаппаратах, снимки в формате *JPEG* принимаются на печать в лабораториях. Наиболее широко *JPEG* используется при создании изображений для электронного распространения на CD-ROM или в Интернете. Недостаток этого формата заключается в том, что при каждом сохранении изображение несколько искажается. Он основан на удалении из изображения той информации, которая все равно не воспринимается (или слабо воспринимается) человеческим глазом. Лишенное избыточной информации изображение занимает гораздо меньше места, чем исходное. Степень сжатия, а следовательно, и количество удаляемой информации плавно регулируется. Низкие степени сжатия дают лучшее качество изображения, а высокие могут существенно его ухудшить. Компактность файлов *JPEG* делает этот формат незаменимым в тех случаях, когда размер файлов критичен, например, при их передаче по каналам связи. В полиграфии использовать его не рекомендуется.

Формат GIF *(Graphics Interchange Format)* создан крупнейшей онлайновой службой *CompuServe* (ныне подразделение AOL, *America Online*) специально для передачи растровых изображений в глобальных сетях. Ориентирован на компактность и использует алгоритм сжатия *LZW*, не приводящий к потере качества. Используется только по своему первоначальному предназначению – в *Internet*, поскольку поддерживает только индексированные изображения. Версия позволяет сохранять в одном файле несколько индексированных изображений (почти как слои в *Photoshop*). Браузеры способны демонстрировать все эти изображения по очереди, получая в результате несложную анимацию. В файле анимации хранятся не только кадры анимации, но и параметры ее демонстрации. *GIF*-анимация в силу своей простоты наиболее распространена в *Internet*. Кроме того, один из цветов в палитре индексированного изображения можно объявить прозрачным. В браузере участки этого цвета будут прозрачными, сквозь них будет виден фон страницы.

Формат PNG *(Portable Network Graphics)*. Как явствует из названия, формат PNG предназначен для передачи изображений по сети. Это достаточно «свежий» формат для *Web*-графики, конкурирующий с *GIF*. Все последние версии браузеров поддерживают его без специальных подключаемых модулей. Формат поддерживает полутоновые и полноцветные RGB-изображения с единственным альфа-каналом, а также индексированные и монохромные изображения без альфа-каналов. Альфа-канал служит маской прозрачности. Таким образом, формат *PNG* – единственный из распространенных форматов, позволяющий получать полноцветные изображения с прозрачным фоном. В формате *PNG* использован мощный алгоритм сжатия без потерь информации, основанный на популярном *LZW*-сжатии.

Формат PSD *(PhotoShop Document)* – это собственный, «родной» формат программы *Adobe Photoshop*. Единственный формат, поддерживающий все возможности программы. В нем сохраняется не только изображение, но и служебная информация программы (слои, из которых состоит документ, попавшие за границы фрагменты рисунка и т.д.). Предпочтителен для хранения промежуточных результатов редактирования изображений, так как сохраняет их послойную структуру, но для вставки изображения в документ *Word* рекомендуется сохранять файлы в других графических форматах. Все последние версии продуктов фирмы *Adobe Systems* поддерживают этот формат и позволяют импортировать файлы *Photoshop* непосредственно. К недостаткам формата можно отнести недостаточную совместимость с другими распространенными приложениями и отсутствие возможности сжатия. Поскольку формат *PSD* способен хранить любой документ, который можно создать в *Photoshop*, при сохранении не требуется указание каких-либо параметров.

Формат PCX – один из первых растровых форматов, созданных фирмой *ZSoft* для программы *PC Paintbrush*. Поддерживает монохромные, индексированные и полноцветные RGB-изображения. Формат предполагает

использование простейшего алгоритма сжатия *RLE* без потерь информации. Ныне имеет преимущественно историческое значение.

4.3. Аффинные преобразования на плоскости

В компьютерной графике все, что относится к двумерному случаю, принято обозначать символом (*2D*) (*2-Dimension*, т.е. 2-Размерный). Допустим, что на плоскости введена прямолинейная координатная система. Тогда каждой точке *M* ставится в соответствие упорядоченная пара чисел *(x, y)* – ее координат (рис. 56).

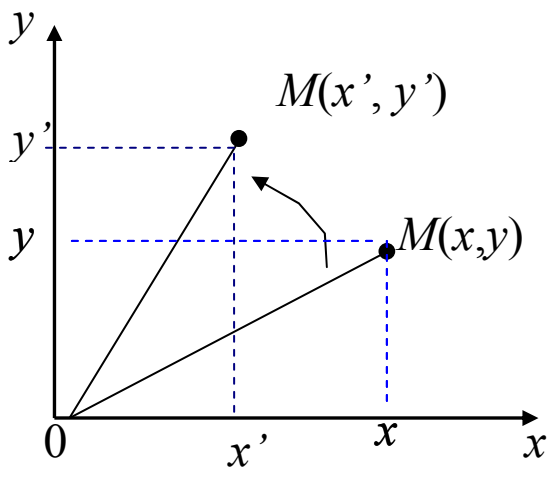


Рис. 56. Пример представления точки на плоскости

Вводя на плоскости еще одну прямолинейную систему координат, мы ставим в соответствие той же точке *M* другую пару чисел – *(x',y')*.

Переход от одной прямолинейной координатной системы на плоскости к другой описывается следующими соотношениями:

$$x' = \alpha x + \beta y + \lambda$$
$$y' = \gamma x + \delta y + \mu,$$

где $\alpha, \beta, \lambda, \gamma, \delta, \mu$ – произвольные числа.

Числа $\alpha, \beta, \gamma, \delta$ - связаны неравенством:

$$\begin{vmatrix} \alpha & \beta \\ \gamma & \delta \end{vmatrix} \neq 0.$$

4.3.1. Вращение (поворот)

Поворот вокруг начальной точки на угол φ (рис. 57) описывается формулами:

$$x' = x\cos\varphi - y\sin\varphi$$
$$y' = x\sin\varphi + y\cos\varphi.$$

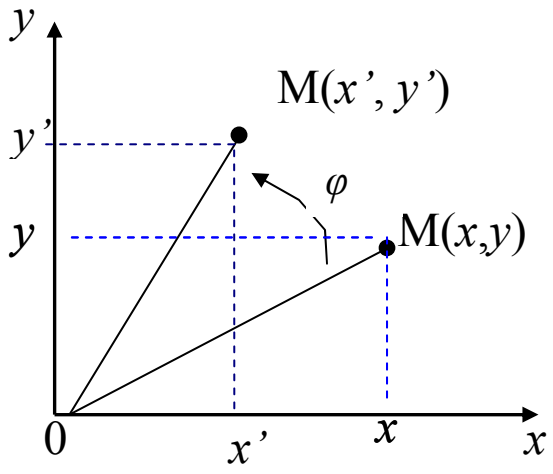


Рис. 57. Поворот *OM* на угол φ относительно центра координат

В дальнейшем мы будем рассматривать формулы как правила, согласно которым в заданной системе прямолинейных координат преобразуются точки плоскости.

В аффинных преобразованиях на плоскости особую роль играют несколько важных частных случаев, имеющих хорошо прослеживаемые геометрические характеристики. При исследовании геометрического смысла числовых коэффициентов в формулах для этих случаев нам удобно считать, что заданная система координат является прямоугольной декартовой.

4.3.2. Растяжение (сжатие)

Растяжение (сжатие) вдоль координатных осей можно задать так:

$$x' = \alpha x$$
$$y' = \delta y,$$
$$\alpha > 0, \delta > 0.$$

Растяжение вдоль оси абсцисс обеспечивается при условии, что $\alpha > 1$, а сжатие вдоль той же оси – при $\alpha < 1$.

На рис. 58 $\alpha=\delta>1$.

В данном примере мы наблюдаем изменение положения красного квадрата. Это происходит потому, что объект не находится в центре координат, и расстояние $\boldsymbol{x}$ до объекта изменяется в том же масштабе, что и все точки объекта. На практике этого можно избегать, привязывая координаты либо к центру самого объекта, либо к одному из его углов.

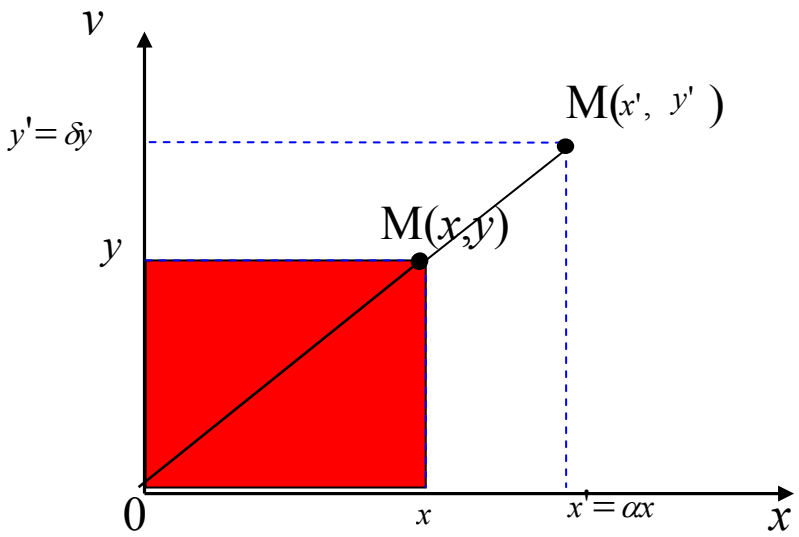


Рис. 58. Изменение положения точки посредством растяжения вектора

4.3.3. Отражение

Отражение (относительно оси абсцисс, рис. 59) задается при помощи формул:

$$x' = x$$
$$y' = -y.$$

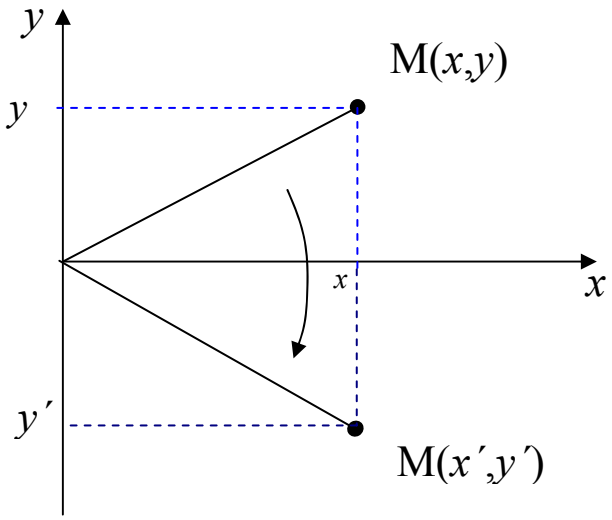


Рис. 59. Отражение точки относительно оси *OX*

4.3.4. Перенос (сдвиг)

Перенос, или сдвиг показан на рис. 60: вектор переноса MM' имеет координаты λ, и μ. Перенос описывают соотношения:

$$x' = x + \lambda$$
$$y' = y + \mu.$$

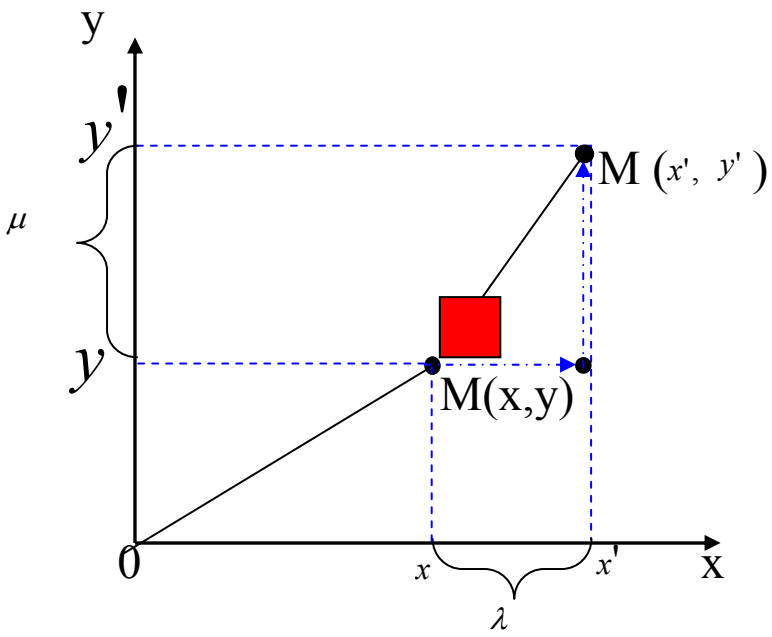


Рис. 60. Перенос точки посредством изменения её координат

Выбор этих четырех частных случаев определяется двумя обстоятельствами:

1) каждое из приведенных выше преобразований имеет простой и наглядный геометрический смысл (геометрическим смыслом наделены и постоянные числа, входящие в приведенные формулы);

2) как доказывается в курсе аналитической геометрии, любое преобразование вида (x',y') п.3 всегда можно представить как последовательное выполнение (суперпозицию) простейших преобразований вида 4.3.1 – 4.3.4 в нужной последовательности. Таким образом, справедливо следующее важное свойство аффинных преобразований плоскости: любое отображение вида (x',y') п.3 можно описать при помощи отображений, задаваемых формулами 4.3.1 – 4.3.4.

Для эффективного использования этих известных формул в задачах компьютерной графики более удобной является их матричная запись. Матрицы, соответствующие случаям 4.3.1, 4.3.2 и 4.3.3, строятся легко и имеют соответственно следующий вид:

$$\begin{bmatrix} \cos\varphi & \sin\varphi \\ -\sin\varphi & \cos\varphi \end{bmatrix}, \quad \begin{bmatrix} \alpha & 0 \\ 0 & \delta \end{bmatrix}, \quad \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}.$$

К сожалению, преобразование 4.3.4 (перенос, или сдвиг) невозможно описать с помощью матрицы размерности 2x2.

4.3.5. Однородные координаты точки

Для решения рассматриваемых далее задач весьма желательно охватить матричным подходом все 4 простейших преобразования (в том числе и перенос), а, значит, и общее аффинное преобразование. Этого можно достичь,

например, так: перейти к описанию произвольной точки плоскости, не упорядоченной парой чисел, как это было сделано выше, а упорядоченной тройкой чисел.

Пусть M – произвольная точка плоскости с координатами x и y, вычисленными относительно заданной прямолинейной координатной системы. Однородными координатами этой точки называется любая тройка одновременно неравных нулю чисел x_1, x_2, x_3, связанных с заданными числами x и y следующими соотношениями:

$$\frac{x_1}{x_3} = x, \quad \frac{x_2}{x_3} = y.$$

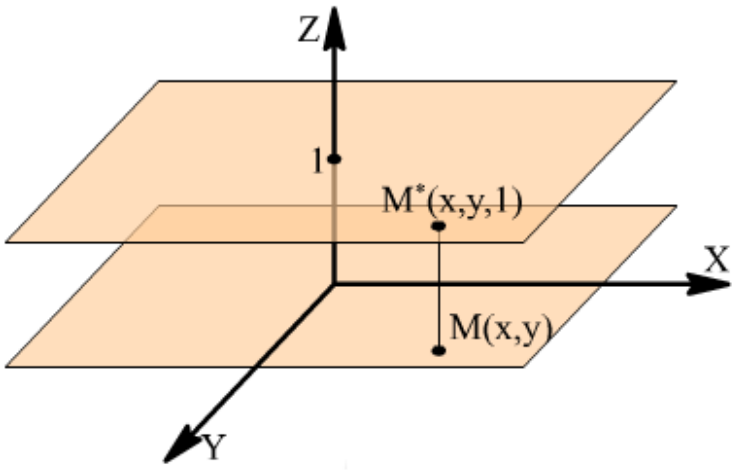


Рис. 61. Переход к однородным координатам

При решении задач компьютерной графики однородные координаты обычно вводятся так: произвольной точке $M(x,y)$ плоскости ставится в соответствие точка $M(x,y,1)$ в пространстве (рис. 61).

Заметим, что произвольная точка на прямой, соединяющей начало координат, точку $O\ (0,\ 0,\ 0)$, с точкой $M\ (x,\ y,\ 1)$, может быть задана тройкой чисел вида $(h_x,\ h_y,\ h)$. Будем считать, что $h \neq 0$.

Вектор с координатами $(h_x,\ h_y,\ h)$ является направляющим вектором прямой, соединяющей точки O $(0,\ 0,\ 0)$ и M $(x,\ y,\ 1)$. Эта прямая пересекает плоскость $Z=1$ в точке $(x,\ y,\ 1)$, которая однозначно определяет точку $(x,\ y)$ координатной плоскости xy.

Тем самым между произвольной точкой с координатами $(x,\ y)$ и множеством троек чисел вида $(h_x,\ h_y,\ h)$, $h \neq 0$, устанавливается (взаимно однозначное) соответствие, позволяющее считать числа h_x, h_y, h новыми координатами этой точки.

Замечание. Широко используемые в проективной геометрии однородные координаты позволяют эффективно описывать так называемые несобственные элементы (по существу, те, которыми проективная плоскость отличается от привычной нам евклидовой плоскости).

В проективной геометрии для однородных координат принято следующее обозначение: $x;y;1$,

или, более общее, x_1, x_2, x_3 (напомним, что здесь непременно требуется, чтобы числа x_1, x_2, x_3 одновременно в нуль не обращались).

Применение однородных координат оказывается удобным уже при решении простейших задач.

Рассмотрим, например, вопросы, связанные с изменением масштаба. Если устройство отображения работает только с целыми числами (или если необходимо работать только с целыми числами), то для произвольного значения h (например, $h = 1$) точку с однородными координатами (0.5, 0.1, 2.5) представить нельзя. Однако при разумном выборе h можно добиться того, чтобы координаты этой точки были целыми числами. В частности, при $h = 10$ для рассматриваемого примера имеем (5, 1, 25).

Возьмем другой случай. Чтобы результаты преобразования не приводили к арифметическому переполнению, для точки с координатами (80000, 40000, 1000) можно взять, например, $h = 0{,}001$. В результате получим (80, 40, 1).

Приведенные примеры показывают полезность использования однородных координат при проведении расчетов. Однако основной целью введения однородных координат в компьютерной графике является их несомненное удобство в применении к геометрическим преобразованиям.

Замечание. При помощи троек однородных координат и матриц третьего порядка можно описать любое аффинное преобразование на плоскости.

$$[x' \quad y' \quad 1]=[x \quad y \quad 1]\cdot\begin{bmatrix}\alpha & \gamma & 0\\ \beta & \delta & 0\\ \lambda & \mu & 1\end{bmatrix}.$$

Замечание. Иногда в литературе используется другая запись – запись по столбцам:

$$\begin{bmatrix}x'\\ y'\\ 1\end{bmatrix}=\begin{bmatrix}\alpha & \beta & \lambda\\ \gamma & \delta & \mu\\ 0 & 0 & 1\end{bmatrix}\cdot\begin{bmatrix}x\\ x\\ 1\end{bmatrix}.$$

Такая запись эквивалентна приведенной выше записи по строкам (и получается из нее транспонированием).

4.3.6. Представление преобразований на плоскости с помощью матриц 3-го порядка

Элементы произвольной матрицы аффинного преобразования не несут в себе явно выраженного геометрического смысла. Поэтому чтобы реализовать то или иное отображение, т. е. найти элементы соответствующей матрицы по заданному геометрическому описанию, необходимы специальные приемы. Обычно построение этой матрицы в соответствии со сложностью рассматриваемой задачи и с описанными выше частными случаями разбивают на несколько этапов.

На каждом этапе ищется матрица, соответствующая тому или иному из выделенных выше случаев 4.3.1 – 4.3.4, обладающих хорошо выраженными геометрическими свойствами.

Выпишем соответствующие матрицы третьего порядка.
А. Матрица вращения *(rotation)*:

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

Б. Матрица растяжения или сжатия *(dilatation)*:

$$D = \begin{bmatrix} \alpha & 0 & 0 \\ 0 & \delta & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

В. Матрица отражения *(reflection)*:

$$M = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Г. Матрица переноса *(translation)*:

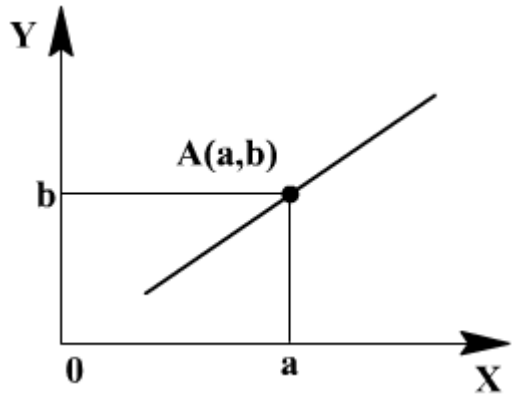
$$T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \lambda & \mu & 1 \end{bmatrix}.$$

Рассмотрим примеры аффинных преобразований на плоскости.

Пример. Построить матрицу поворота вокруг точки A(a, b) на угол φ (рис. 62).

1-й шаг. Перенос на вектор $A(-a,-b)$ для совмещения центра поворота с началом координат. Матрица соответствующего преобразования имеет вид:

$$T_{-A} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -a & -b & 1 \end{bmatrix}.$$



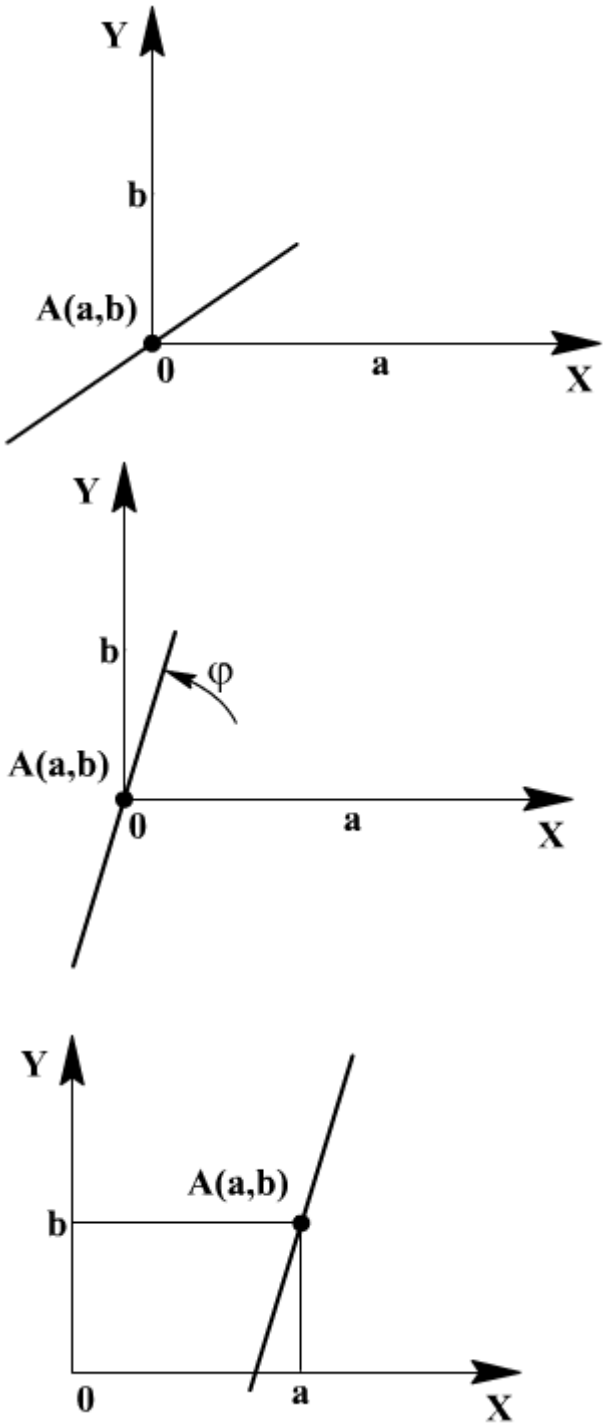


Рис. 62. Пример поворота прямой на плоскости

2-й шаг. Поворот на угол φ. Матрица поворота:

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

3-й шаг. Перенос на вектор *A(a, b)* для возвращения центра поворота в прежнее положение. Матрица переноса:

$$T_A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ a & b & 1 \end{bmatrix}.$$

Перемножим матрицы в том же порядке, как они выписаны:

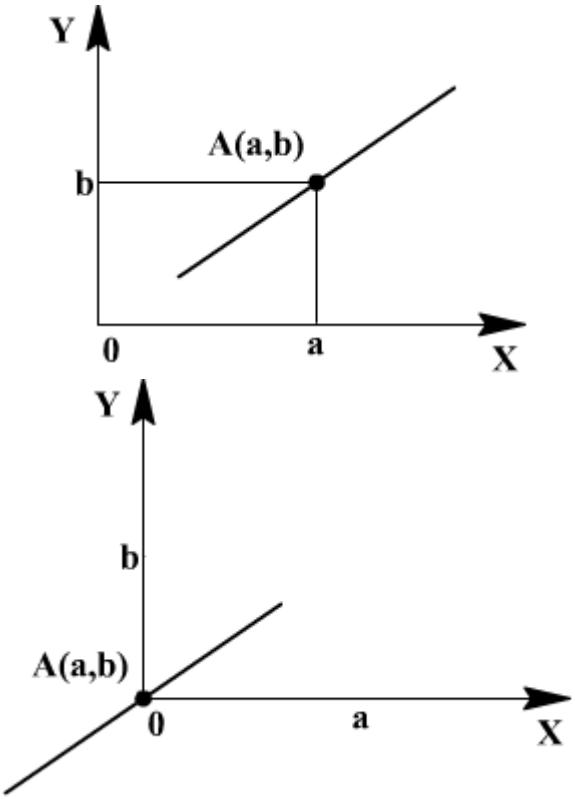
$$T_{-A} \cdot R_{\varphi} \cdot T_A.$$

В результате получим, что искомое преобразование (в матричной записи) будет выглядеть следующим образом:

$$[x' \quad y' \quad 1] = [x \quad y \quad 1] \cdot \begin{bmatrix} \cos\varphi & \sin\varphi & 0 \\ -\sin\varphi & \cos\varphi & 0 \\ -a\cdot\cos\varphi + b\cdot\sin\varphi + a & -a\cdot\sin\varphi - b\cdot\cos\varphi + b & 1 \end{bmatrix}.$$

Элементы полученной матрицы (особенно в последней строке) не так легко запомнить. В то же время каждая из трех перемножаемых матриц по геометрическому описанию соответствующего отображения легко строится.

<u>Пример.</u> Построить матрицу растяжения с коэффициентами растяжения α вдоль оси абсцисс и δ вдоль оси ординат и с центром в точке A(a,b) (рис. 63).



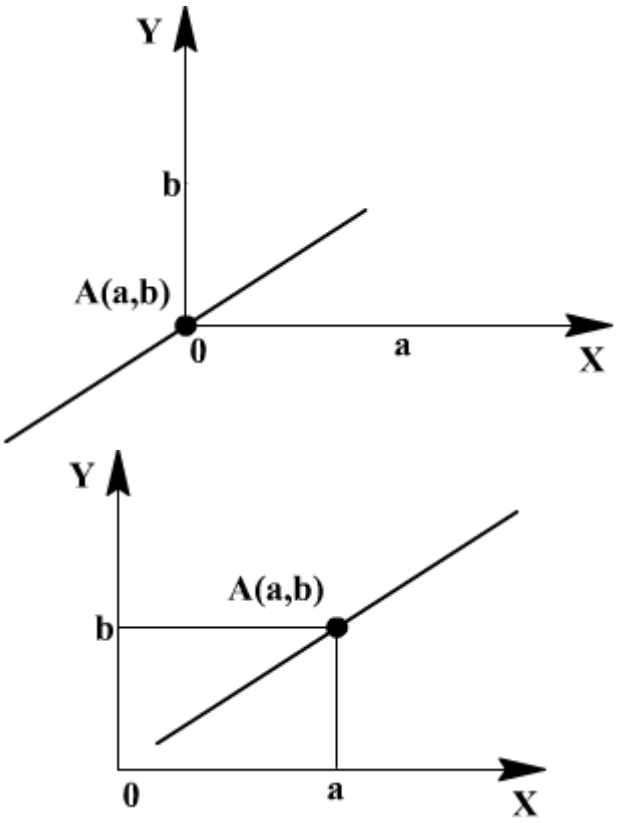


Рис. 63. Пример масштабирования прямой на плоскости

1-й шаг. Перенос на вектор *-A(-a,-b)* для совмещения центра растяжения с началом координат; матрица соответствующего преобразования имеет вид:

$$T_{-A} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -a & -b & 1 \end{bmatrix}.$$

2-й шаг. Растяжение вдоль координатных осей с коэффициентами α и δ соответственно; матрица преобразования имеет вид:

$$D = \begin{bmatrix} \alpha & 0 & 0 \\ 0 & \delta & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

3-й шаг. Перенос на вектор *-A(a,b)* для возвращения центра растяжения в прежнее положение; матрица соответствующего преобразования:

$$T_{A} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ a & b & 1 \end{bmatrix}.$$

Перемножив матрицы в том же порядке

$$T_{-A} \cdot D \cdot T_{A},$$

получим окончательно:

$$\begin{bmatrix} x' & y' & 1 \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} \cdot \begin{bmatrix} \alpha & 0 & 0 \\ 0 & \delta & 0 \\ (1-\alpha)a & (1-\delta)b & 1 \end{bmatrix}.$$

Замечание. Рассуждая подобным образом, т. е. разбивая предложенное преобразование на этапы, поддерживаемые матрицами R, D, M, T можно построить матрицу любого аффинного преобразования по его геометрическому описанию.

4.3.7. Преобразования в 3-мерном пространстве

Обратимся теперь к трехмерному случаю (3-*Dimension*, или 3*D*) и начнем рассмотрение сразу с введения однородных координат. Поступая аналогично тому, как это было сделано для размерности два, заменим координатную тройку *(x, y, z)*, задающую точку в пространстве, на четверку чисел *(x, y, z, 1)* или, в более общем виде, на четверку (h_x, h_y, h_z, h), $h \neq 0$.

Каждая точка пространства (кроме начальной точки *O*) может быть задана четверкой одновременно не равных нулю чисел; эта четверка чисел определена однозначно с точностью до общего множителя.

Предложенный переход к новому способу задания точек дает возможность воспользоваться матричной записью и в более сложных, трехмерных задачах.

Любое аффинное преобразование в трехмерном пространстве может быть представлено в виде суперпозиции вращений, растяжений, отражений и переносов. Поэтому вполне уместно сначала подробно описать матрицы именно этих преобразований. Очевидно, что в данном случае порядок матриц должен быть равен четырем, поскольку помимо трех пространственных координат еще одну единицу размерности добавляет тот факт, что мы используем однородные координаты.

Вращение в пространстве

На рис. 64 показан поворот на угол φ в трехмерном пространстве некоего объекта (куба) вокруг оси абсцисс.

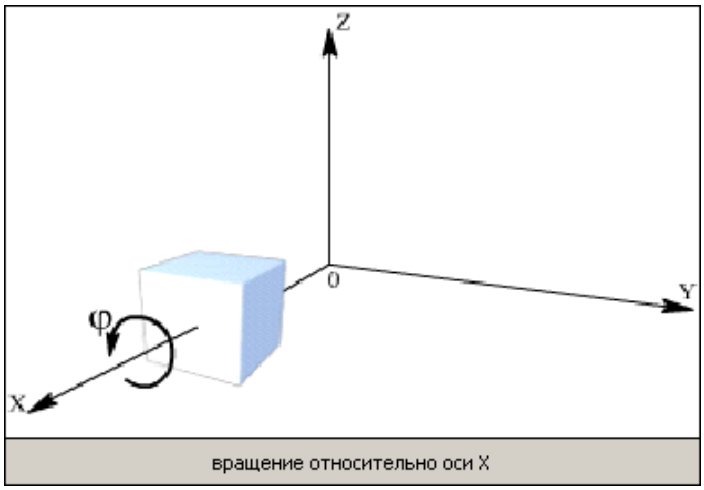


Рис. 64. Поворот куба в трехмерном пространстве вокруг оси абсцисс

Такой поворот описывается в однородных координатах с помощью матрицы 4-го порядка:

$$[R_x] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\varphi & \sin\varphi & 0 \\ 0 & -\sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

На рис. 65 показан поворот на угол ψ в трехмерном пространстве некоего объекта (куба) вокруг оси ординат.

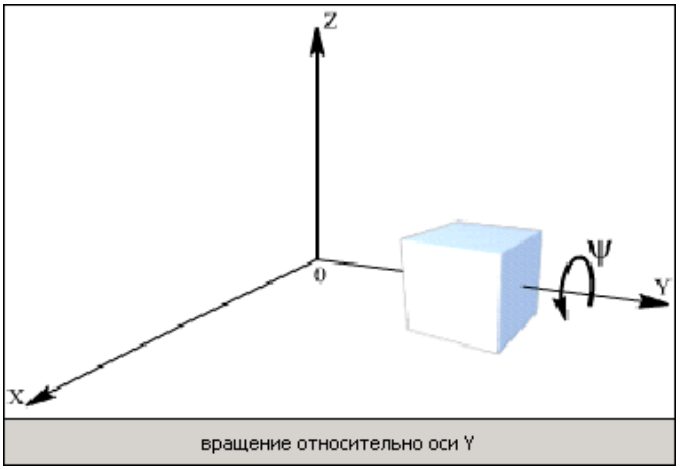


Рис. 65. Поворот куба в трехмерном пространстве вокруг оси ординат

Такой поворот описывается в однородных координатах с помощью матрицы 4-го порядка:

$$[R_y] = \begin{bmatrix} \cos\psi & 0 & -\sin\psi & 0 \\ 0 & 1 & 0 & 0 \\ \sin\psi & 0 & \cos\psi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

На рис. 66 показан поворот на угол χ в трехмерном пространстве некоего объекта (куба) вокруг оси аппликат.

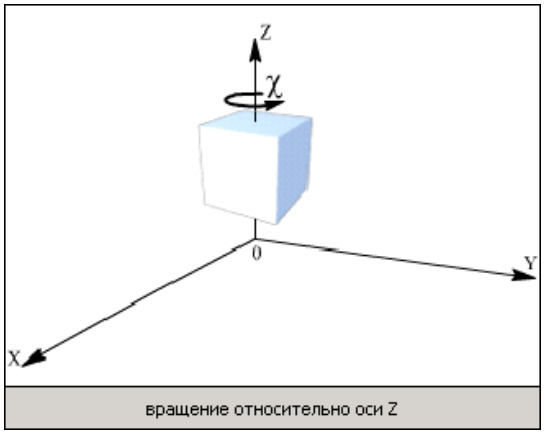


Рис. 66. Поворот куба в трехмерном пространстве на угол χ вокруг оси аппликат

Такой поворот описывается в однородных координатах с помощью матрицы 4-го порядка:

$$[R_z]=\begin{bmatrix} \cos\chi & \sin\chi & 0 & 0 \\ -\sin\chi & \cos\chi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Замечание. Полезно обратить внимание на место знака "-" в каждой из трех приведенных матриц.

Растяжение (сжатие), или масштабирование в пространстве

Матрица растяжения (сжатия) в трехмерном пространстве в однородных координатах имеет следующий вид:

$$D=\begin{bmatrix} \alpha & 0 & 0 & 0 \\ 0 & \delta & 0 & 0 \\ 0 & 0 & \beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

где $\alpha>0$ – коэффициент растяжения (сжатия) вдоль оси абсцисс;
$\delta>0$ – коэффициент растяжения (сжатия) вдоль оси ординат;
$\beta > 0$ – коэффициент растяжения (сжатия) вдоль оси аппликат.

На рис. 67 показано растяжение объекта (куба).

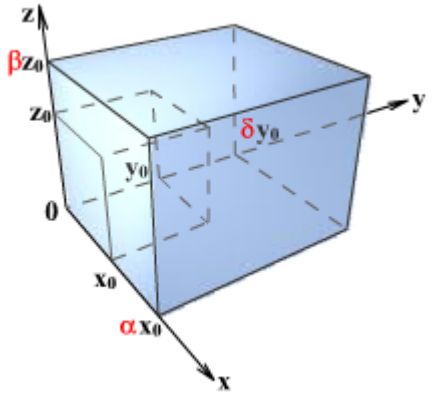


Рис. 67. Растяжение тела в пространстве

Отражение в пространстве

Матрица отражения относительно плоскости *YOZ* имеет вид:

$$M_x=\begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

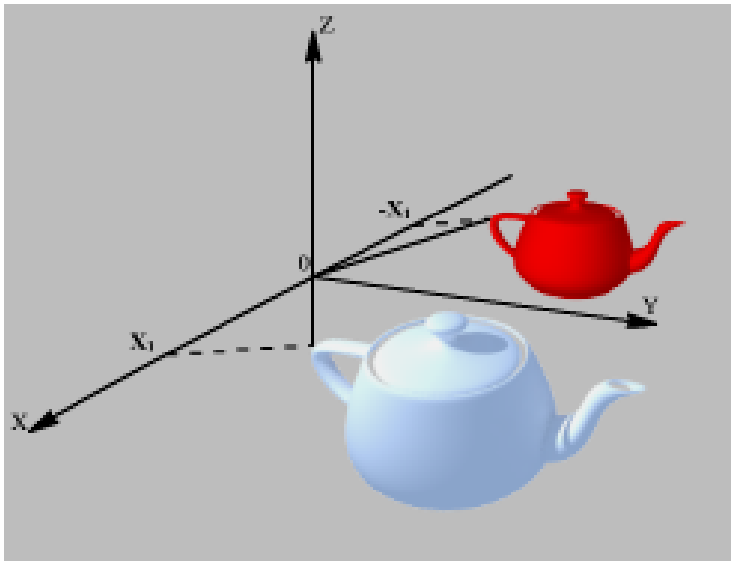


Рис. 68. Отражение относительно плоскости *YOZ* (изменяет знак только проекция на ось абсцисс x_1)

На рис. 68 отраженный чайник условно окрашен в красный цвет для того, чтобы проще было отличить отражение от исходного изображения. В действительности при отражении цвет объекта не изменяется ☺.

Матрица отражения относительно плоскости *XOZ* имеет вид:

$$M_y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Перенос в пространстве

Матрица переноса на величину λ по оси абсцисс, μ по оси ординат и ν – по оси аппликат имеет следующий вид:

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \lambda & \mu & \nu & 1 \end{bmatrix},$$

где (λ, μ, ν) – вектор переноса. На рис. 69 показан перенос куба.

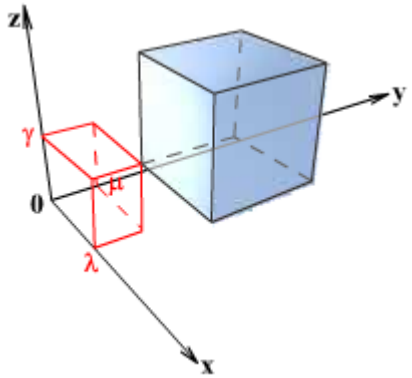


Рис. 69. Перенос тела в пространстве

Замечание. Как и в двумерном случае, все выписанные матрицы невырожденные.

Приведем важный пример построения матрицы сложного преобразования по его геометрическому описанию.

4.3.8. Примеры преобразований

Пример. Построить матрицу вращения на угол φ вокруг прямой L, проходящей через точку A(a, b, c) и имеющую направляющий вектор (l, m, n). Можно считать, что направляющий вектор прямой является единичным:

$$l^2 + m^2 + n^2 = 1.$$

На рис. 70 схематично показано, матрицу какого преобразования требуется найти.

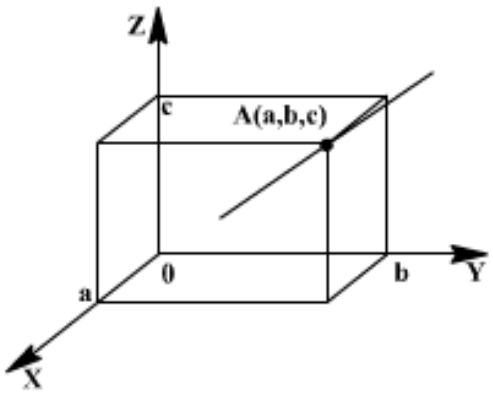


Рис. 70. Поворот вокруг произвольной прямой в пространстве

Решение сформулированной задачи разбивается на несколько шагов. Опишем последовательно каждый из них.

1-й шаг. Перенос на вектор *-A(-a,-b,-c)* при помощи матрицы

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -a & -b & -c & 1 \end{bmatrix}.$$

В результате этого переноса мы добиваемся того, чтобы прямая *L* проходила через начало координат.

2-й шаг. Совмещение оси аппликат с прямой L двумя поворотами вокруг оси абсцисс и оси ординат.

Первый поворот – вокруг оси абсцисс на угол ψ (подлежащий определению). Чтобы найти этот угол, рассмотрим ортогональную проекцию L' исходной прямой L на плоскость $X=0$.

Направляющий вектор прямой L' определяется просто – он равен $(0, m, n)$. Отсюда сразу же вытекает, что

$$\cos\psi = \frac{n}{d}, \quad \sin\psi = \frac{m}{d},$$

где

$$d = \sqrt{m^2 + n^2}.$$

Соответствующая матрица вращения имеет следующий вид:

$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{n}{d} & \frac{m}{d} & 0 \\ 0 & -\frac{m}{d} & \frac{n}{d} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Под действием преобразования, описываемого этой матрицей, координаты вектора (l, m, n) изменятся. Подсчитав их, в результате получим

$$[l, m, n, 1] \cdot R_x = [l, 0, d, 1].$$

3-й шаг. Второй поворот – вокруг оси ординат на угол θ, определяемый соотношениями:

$$\cos\theta = 1, \quad \sin\theta = -d.$$

Соответствующая матрица вращения записывается в следующем виде:

$$R_y = \begin{bmatrix} 1 & 0 & d & 0 \\ 0 & 1 & 0 & 0 \\ -d & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

4-й шаг. Вращение вокруг прямой L на заданный угол φ. Так как теперь прямая L совпадает с осью аппликат, то соответствующая матрица имеет следующий вид:

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

5-й шаг. Поворот вокруг оси ординат на угол $-\theta$ (возвращение).

6-й шаг. Поворот вокруг оси абсцисс на угол $-\psi$ (возвращение).

Замечание: Вращение в пространстве ***некоммутативно****. Это значит, что, перемножив матрицы в другом порядке, получим другой результат! Поэтому порядок, в котором проводятся вращения, является весьма существенным.*

7-й шаг. Перенос на вектор *A(a,b,c)*.

Перемножив найденные матрицы в порядке их построения, получим следующую матрицу:

$$T \cdot R_x \cdot R_y \cdot R_z \cdot (R_y)^{-1} \cdot (R_x)^{-1} \cdot (T)^{-1}.$$

Выпишем окончательный результат, считая для простоты, что ось вращения *L* проходит через начальную точку:

$$\begin{bmatrix} l^2 + \cos\varphi \cdot (1 - l^2) & l \cdot (1 - \cos\varphi) \cdot m + n \cdot \sin\varphi & l \cdot (1 - \cos\varphi) \cdot m + n \cdot \sin\varphi & 0 \\ l \cdot (1 - \cos\varphi) \cdot m - n \cdot \sin\varphi & m^2 + \cos\varphi \cdot (1 - m^2) & m \cdot (1 - \cos\varphi) \cdot n + l \cdot \sin\varphi & 0 \\ l \cdot (1 - \cos\varphi) \cdot n + m \cdot \sin\varphi & m \cdot (1 - \cos\varphi) \cdot n - l \cdot \sin\varphi & n^2 + \cos\varphi \cdot (1 - n^2) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Рассматривая другие примеры подобного рода, мы будем получать в результате невырожденные матрицы вида:

$$A = \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 & 0 \\ \beta_1 & \beta_2 & \beta_3 & 0 \\ \gamma_1 & \gamma_2 & \gamma_3 & 0 \\ \lambda & \mu & \nu & 1 \end{bmatrix}.$$

При помощи таких матриц можно преобразовывать любые плоские и пространственные фигуры.

Пример. Требуется подвергнуть заданному аффинному преобразованию выпуклый многогранник.

Для этого сначала по геометрическому описанию отображения находим его матрицу *A*. Замечая далее, что произвольный выпуклый многогранник однозначно задается набором всех своих вершин

$$V_i(x_i, y_i, z_i), \quad i = 1, \ldots, n,$$

строим матрицу

$$V = \begin{bmatrix} x_1 & y_1 & z_1 & 1 \\ . & . & . & . \\ . & . & . & . \\ x_n & y_n & z_n & 1 \end{bmatrix}.$$

Подвергая этот набор преобразованию, описываемому найденной невырожденной матрицей четвертого порядка *V·A*, мы получаем набор вершин нового выпуклого многогранника – образа исходного.

4.4. Проектирование

Изображение объектов па картинной плоскости связано с еще одной геометрической, операцией – проектированием при помощи пучка прямых.

4.4.1. Виды проектирования

В компьютерной графике используется несколько различных видов проектирования (иногда называемого также проецированием). Наиболее часто используемые на практике виды проектирования – это параллельное проектирование и центральное проектирование.

Для получения проекции объекта на картинную плоскость необходимо провести через каждую его точку прямую из заданного проектирующего пучка (собственного или несобственного) и затем найти координаты точки пересечения этой прямой с плоскостью изображения. В случае центрального проектирования все прямые исходят из одной точки – центра собственного пучка *C* (рис. 71). При параллельном проектировании центр (несобственного) пучка считается лежащим в бесконечности (рис. 72).

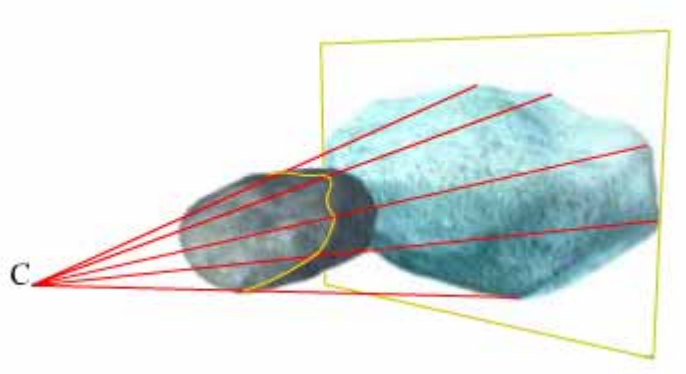


Рис. 71. Пример центральной проекции

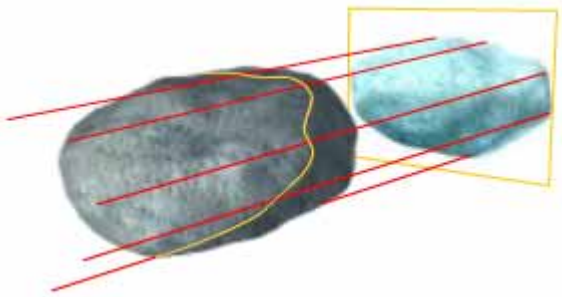


Рис 72. Пример параллельной проекции

Каждый из этих двух основных классов разбивается на несколько подклассов в зависимости от взаимного расположения картинной плоскости и координатных осей. Некоторое представление о видах проектирования могут дать приводимые ниже схемы.

Важное замечание. Использование для описания преобразований проектирования однородных координат и матриц четвертого порядка позволяет упростить изложение и зримо облегчает решение задач геометрического моделирования.

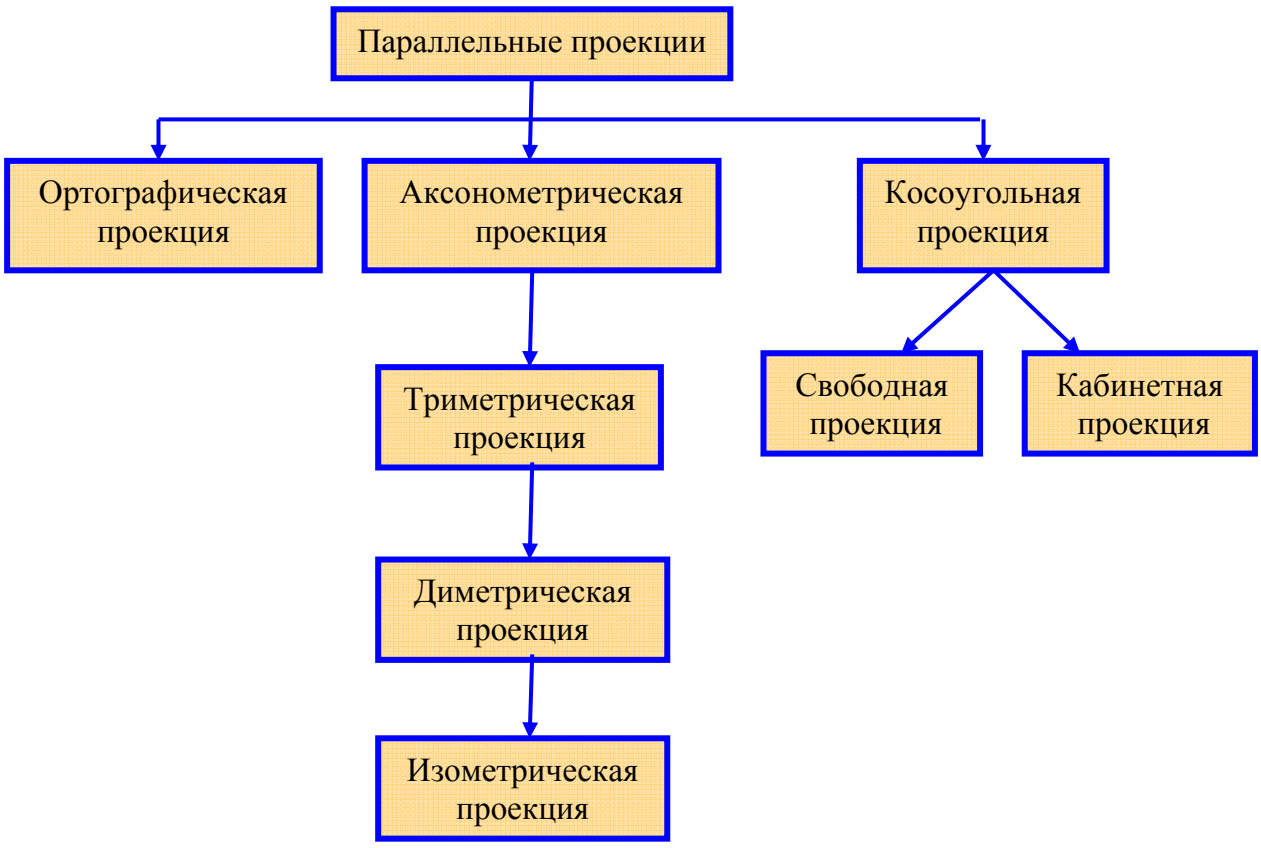


Рис. 73. Классификация параллельных проекций

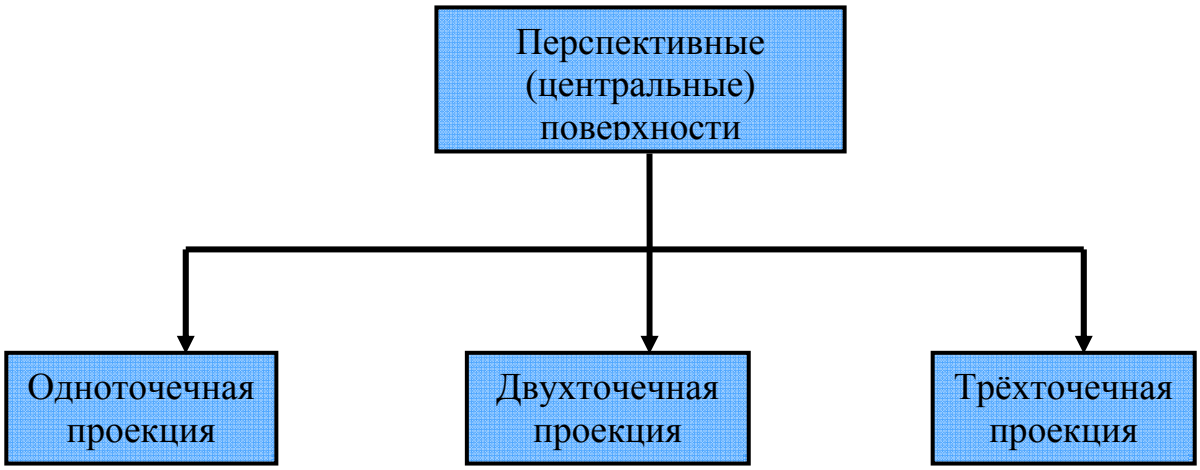


Рис. 74. Классификация центральных проекций

4.4.2. Ортографические проекции

При *ортографической* проекции картинная плоскость совпадает с одной из координатных плоскостей или параллельна ей.

Матрица проектирования вдоль оси *X* на плоскость *YZ* имеет вид:

$$P_x = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

В случае если плоскость проектирования параллельна координатной плоскости (рис. 75), необходимо умножить матрицу P_x на матрицу сдвига. В результате получаем:

$$P_x \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ p & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ p & 0 & 0 & 1 \end{bmatrix}.$$

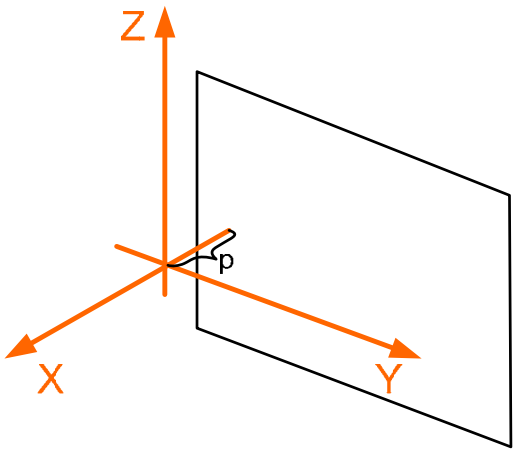


Рис. 75. Случай ортографической проекции, когда картинная плоскость параллельна координатной плоскости *ZOY*

Аналогично записываются матрицы проектирования вдоль двух других координатных осей:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & q & 0 & 1 \end{bmatrix} \quad \text{и} \quad \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & r & 1 \end{bmatrix}.$$

Замечание. Все три полученные матрицы проектирования ***вырожденны***.

4.4.3. Аксонометрические проекции

При *аксонометрической* проекции проектирующие прямые перпендикулярны картинной плоскости.

В соответствии с взаимным расположением плоскости проектирования и координатных осей различают три вида проекций:

1) триметрию – нормальный вектор картинной плоскости (КП) образует с ортами координатных осей попарно различные углы, а следовательно

различные углы образуются между ортами и их проекциями на КП, что и отображено на рис. 76;

2) диметрию – два угла между нормалью картинной плоскости и координатными осями равны, а следовательно равны два угла между этими же ортами и КП (рис. 77);

3) изометрию – все три угла между нормалью, картинной плоскости и координатными осями равны (рис. 78).

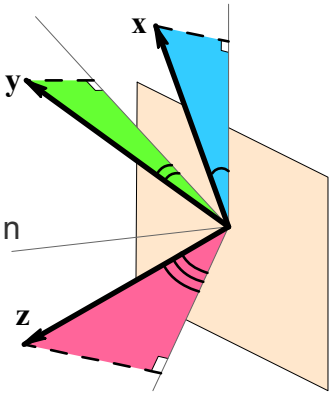


Рис. 76. Триметрия

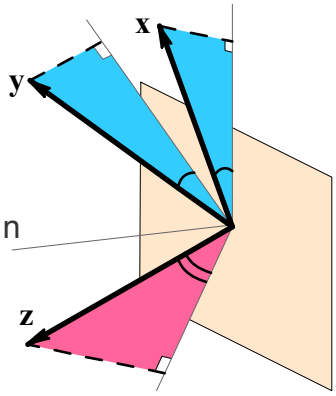


Рис. 77. Диметрия

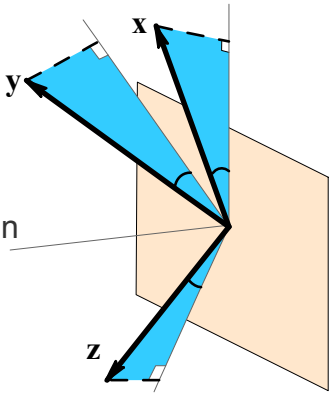


Рис. 78. Изометрия

Каждый из трех видов указанных проекций получается комбинацией поворотов, за которой следует параллельное проектирование. При повороте на угол ψ относительно оси ординат, на угол φ вокруг оси абсцисс и последующего проектирования вдоль оси аппликат получается матрица M:

$$M = \begin{bmatrix} \cos\psi & \sin\varphi\cdot\sin\psi & 0 & 0 \\ 0 & \cos\varphi & 0 & 0 \\ \sin\psi & -\sin\varphi\cdot\cos\psi & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos\psi & 0 & -\sin\psi & 0 \\ 0 & 1 & 0 & 0 \\ \sin\psi & 0 & \cos\psi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\varphi & \sin\varphi & 0 \\ 0 & -\sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

При этом единичные орты координатных осей *X, Y, Z* преобразуются следующим образом:

$$\begin{bmatrix} 1 & 0 & 0 & 1 \end{bmatrix} \cdot M = \begin{bmatrix} \cos\psi & \sin\varphi\cdot\sin\psi & 0 & 1 \end{bmatrix};$$
$$\begin{bmatrix} 0 & 1 & 0 & 1 \end{bmatrix} \cdot M = \begin{bmatrix} 0 & \cos\varphi & 0 & 1 \end{bmatrix};$$
$$\begin{bmatrix} 0 & 0 & 1 & 1 \end{bmatrix} \cdot M = \begin{bmatrix} \sin\psi & -\sin\varphi\cdot\cos\psi & 0 & 1 \end{bmatrix}.$$

Диметрия характеризуется тем, что длины двух проекций совпадают:

$$\cos^2\psi + \sin^2\varphi\cdot\sin^2\psi = \cos^2\varphi\,.$$

Отсюда следует, что

$$\sin^2\psi = tg^2\varphi\,.$$

В случае *изометрии* имеем:

$$\cos^2\psi + \sin^2\varphi\cdot\sin^2\psi = \cos^2\varphi\,,$$
$$\sin^2\psi + \sin^2\varphi\cdot\cos^2\psi = \cos^2\varphi\,.$$

Из последних двух соотношений вытекает, что

$$\sin^2\varphi = \frac{1}{3}, \quad \sin^2\psi = \frac{1}{2}.$$

При *триметрии* длины проекций попарно различны.

4.4.4. Косоугольные проекции

Проекции, для получения которых используется пучок прямых, не перпендикулярных плоскости экрана, принято называть косоугольными. При косоугольном проектировании орта оси Z на плоскость XY (рис. 79) имеем: $(0\ 0\ 1\ 1) \rightarrow (\alpha\ \beta\ 0\ 1)$.

Матрица соответствующего преобразования имеет следующий вид:

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \alpha & \beta & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

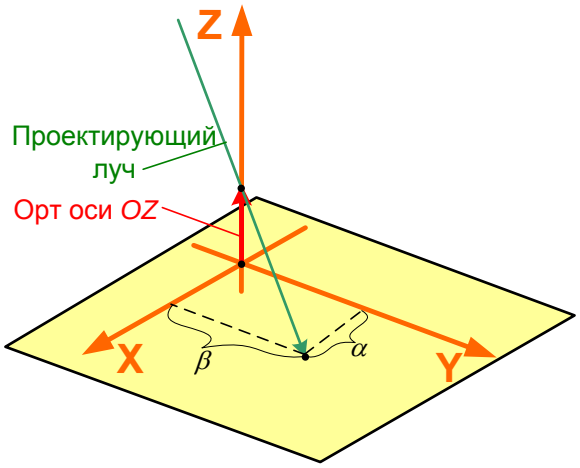


Рис. 79. К понятию косоугольной проекции

Выделяют два вида косоугольных проекций: свободную проекцию (угол наклона проектирующих прямых к плоскости экрана равен половине прямого) и кабинетную проекцию (частный случай свободной проекции – масштаб по третьей оси вдвое меньше).

В случае *свободной* проекции

$$\alpha = \beta = \cos\left(\frac{\pi}{4}\right),$$

в случае *кабинетной* проекции:

$$\alpha = \beta = \frac{1}{2}\cos\left(\frac{\pi}{4}\right).$$

4.4.5. Центральные (перспективные) проекции

Перспективные (центральные) проекции строятся более сложно. Предположим для простоты, что центр проектирования лежит на оси *Z* в точке *C* (*0, 0, c*) и плоскость проектирования совпадает с координатной плоскостью XY (рис. 80). Возьмем в пространстве произвольную точку *M* (*x, y, z*), проведем через нее и точку *C* прямую и запишем соответствующие параметрические уравнения.

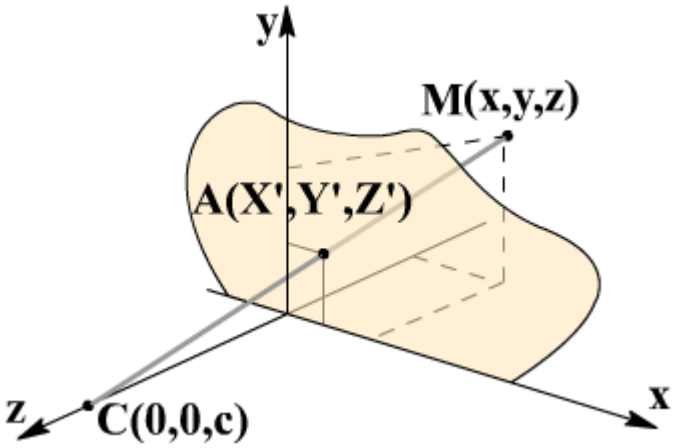


Рис. 80. Пример центральной проекции

Имеем:

$$X' = x \cdot t, \quad Y' = y \cdot t, \quad Z' = c + (z - c) \cdot t .$$

Найдем координаты точки пересечения построенной прямой с плоскостью *XY*. Из условия *Z′ = 0* получаем, что

$$t' = \frac{1}{1 - \frac{z}{c}}$$

и далее: $X' = \frac{1}{1 - \frac{z}{c}} \cdot x, \quad Y' = \frac{1}{1 - \frac{z}{c}} \cdot y .$

Интересно заметить, что тот же самый результат можно получить, привлекая матрицу:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -\frac{1}{c} \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

В самом деле, переходя к однородным координатам, прямым вычислением легко проверить, что

$$\begin{bmatrix} x & y & z & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -\frac{1}{c} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} x & y & 0 & 1-\frac{z}{c} \end{bmatrix}.$$

Вспоминая свойства однородных координат, запишем полученный результат в несколько ином виде:

$$\begin{bmatrix} \frac{x}{1-\frac{z}{c}} & \frac{y}{1-\frac{z}{c}} & 0 & 1 \end{bmatrix}$$

и затем путем непосредственного сравнения убедимся в том, что это координаты той же самой точки.

Замечание. Матрица проектирования, разумеется, ***вырождена****.*

Матрица соответствующего перспективного преобразования (без проектирования) имеет следующий вид:

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -\frac{1}{c} \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Обратим внимание на то, что последняя матрица *не вырождена*. Рассмотрим пучок прямых, параллельных оси *Z*, и попробуем разобраться в том, что с ним происходит под действием матрицы *Q*.

Каждая прямая пучка однозначно определяется точкой (скажем, *M* (*x,y,z*)) своего пересечения с плоскостью *XY* и описывается уравнениями

$$X = x, \quad Y = y, \quad Z = t.$$

Переходя к однородным координатам и используя матрицу *Q*, получаем:

$$\begin{bmatrix} x & y & t & 1 \end{bmatrix} \cdot Q = \begin{bmatrix} x & y & t & 1-\frac{t}{c} \end{bmatrix}, \quad \text{или} \quad \begin{bmatrix} \frac{x}{1-\frac{t}{c}} & \frac{y}{1-\frac{t}{c}} & -c\frac{t}{t-c} & 1 \end{bmatrix}.$$

В пределе при $t \to \infty$ точка (*x, y, t, 1*) преобразуется в (*0, 0, 1, 0*). Для того, чтобы убедиться в этом, следует разделить каждую координату на *t*:

$$\begin{bmatrix} \frac{x}{t} & \frac{y}{t} & 1 & \frac{1}{t} \end{bmatrix}.$$

Точка (*0, 0, -c, 1*) является пределом правой части рассматриваемого равенства при $t \to \infty$:

$$\lim_{t\to\infty} \begin{bmatrix} \frac{x}{1-\frac{t}{c}} & \frac{y}{1-\frac{t}{c}} & -c\frac{t}{t-c} & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & -c & 1 \end{bmatrix}.$$

Тем самым бесконечно удаленный (несобственный) центр (*0, 0, 1, 0*) пучка прямых, параллельных оси *Z*, переходит в точку (*0, 0, - c, 1*) оси *Z*. Вообще каждый несобственный пучок прямых (совокупность прямых,

параллельных заданному направлению), не параллельный картинной плоскости,

$$X = x + l \cdot t, \quad Y = y + m \cdot t, \quad Z = z + n \cdot t, \quad n \neq 0$$

под действием преобразования, задаваемого матрицей Q, переходит в собственный пучок

$$\begin{bmatrix} x + l \cdot t & y + m \cdot t & n \cdot t & 1 \end{bmatrix} \cdot Q = \begin{bmatrix} x + l \cdot t & y + m \cdot t & n \cdot t & 1 - \dfrac{n \cdot t}{c} \end{bmatrix}.$$

Центр этого пучка

$$\begin{bmatrix} -\dfrac{lc}{n} & -\dfrac{mc}{n} & -c & 1 \end{bmatrix}$$

называется точкой схода.

Принято выделять так называемые главные точки схода, которые соответствуют пучкам прямых, параллельных координатным осям. Для преобразования с матрицей Q существует лишь одна главная точка схода. В общем случае (когда оси координатной системы не параллельны плоскости экрана) таких точек три.

Матрица соответствующего преобразования выглядит следующим образом:

$$\begin{bmatrix} 1 & 0 & 0 & -\dfrac{1}{a} \\ 0 & 1 & 0 & -\dfrac{1}{b} \\ 0 & 0 & 1 & -\dfrac{1}{c} \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Пучок прямых, параллельных оси *OX* (*1 0 0 0*), *OY*(*0 1 0 0*) переходит в пучок прямых с центром.

На рис. 81 и 82 изображены проекции куба со сторонами, параллельными координатным осям, с одной и с двумя главными точками схода соответственно:

$$\begin{bmatrix} 1 & 0 & 0 & -\dfrac{1}{a} \end{bmatrix} \ \textit{и} \ \begin{bmatrix} 0 & 1 & 0 & -\dfrac{1}{b} \end{bmatrix}, \ \textit{или} \ \begin{bmatrix} -a & 0 & 0 & 1 \end{bmatrix} \ \textit{и} \ \begin{bmatrix} 0 & -b & 0 & 1 \end{bmatrix}.$$

Точки (*-a, 0, 0*) и (*0, -b, 0*) есть главные точки схода.

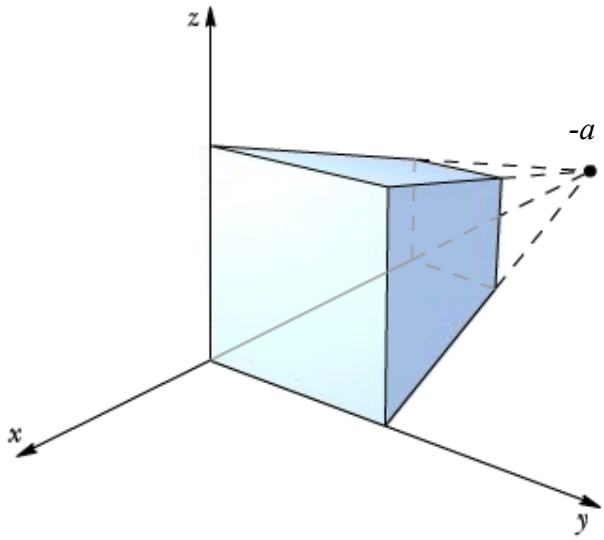


Рис. 81. Проекция куба с одной главной точкой схода

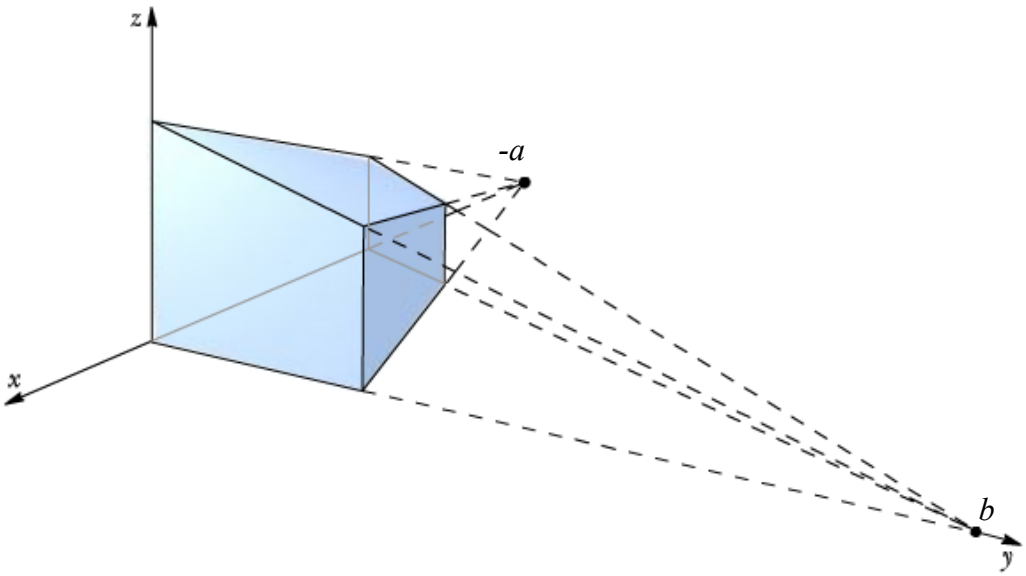


Рис. 82. Проекция куба с двумя главными точками схода

4.5. Растровые алгоритмы

Подавляющее число графических устройств являются растровыми, представляя изображение в виде прямоугольной матрицы (сетки, целочисленной решетки) пикселей (растра), и большинство графических библиотек содержат внутри себя достаточное количество простейших растровых алгоритмов, таких, как:

переведение идеального объекта (отрезка, окружности и др.) в их растровые образы;

обработка растровых изображений.

Тем не менее, часто возникает необходимость и явного построения растровых алгоритмов.

4.5.1. Понятие связности

Достаточно важным понятием для растровой сетки является связность – возможность соединения двух пикселей растровой линией, т. е. последовательным набором пикселей. Возникает вопрос, когда пиксели (x_1,y_1) и (x_2,y_2) можно считать соседними.

Вводится два понятия связности:

• *4-связность*: пиксели считаются соседними, если либо их x-координаты, либо их y-координаты отличаются на единицу (рис. 83):

$$|x_1 - x_2| + |y_1 - y_2| \le 1$$

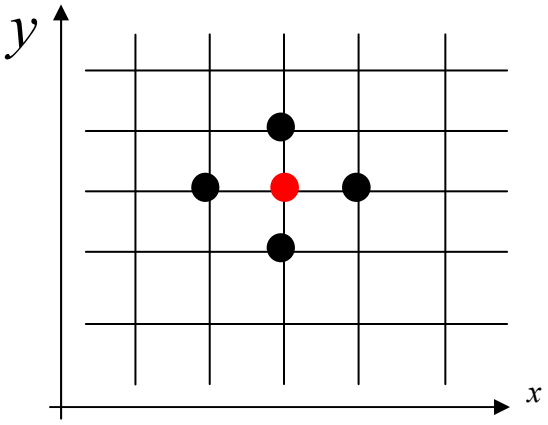


Рис. 83. Пример 4-связности

• *8-связность*: пиксели считаются соседними, если их x-координаты и y-координаты отличаются не более чем на единицу (рис. 84):

$$|x_1 - x_2| \le 1, \quad |y_1 - y_2| \le 1$$

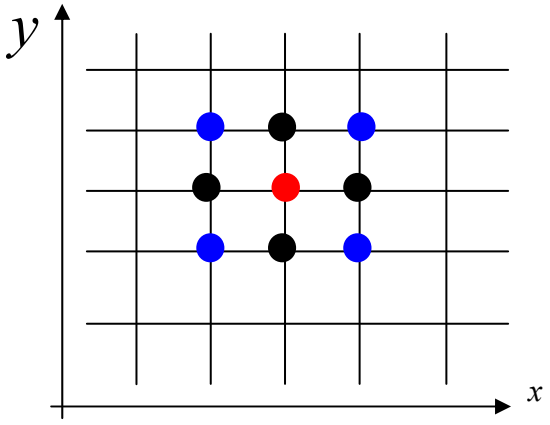


Рис. 84. Пример 8-связности

Понятие 4-связности является более сильным: любые два 4-связных пикселя являются одновременно и 8-связными, но не наоборот. Существуют также целые линии, которые также могут быть 4-связными или 8-связными. На рис. 85 изображена 8-связная линия, а на рис. 86 – 4-связная линия.

В качестве линии на растровой сетке выступает набор пикселей P_1, P_2, ..., P_n, где любые два пикселя P_i, P_{i+1} являются соседними в смысле заданной связности.

Замечание. Так как понятие линии базируется на понятии связности, то естественным образом возникает понятие 4- и 8-связных линий. Поэтому, когда мы говорим о растровом представлении (например, отрезка), следует ясно понимать, о каком именно представлении идет речь. В общем случае растровое представление объекта не является единственным, и возможны различные способы его построения.

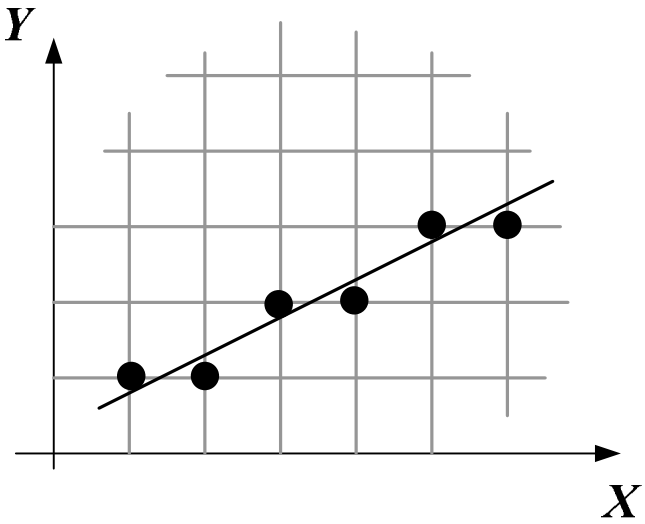


Рис. 85. 8-связная линия

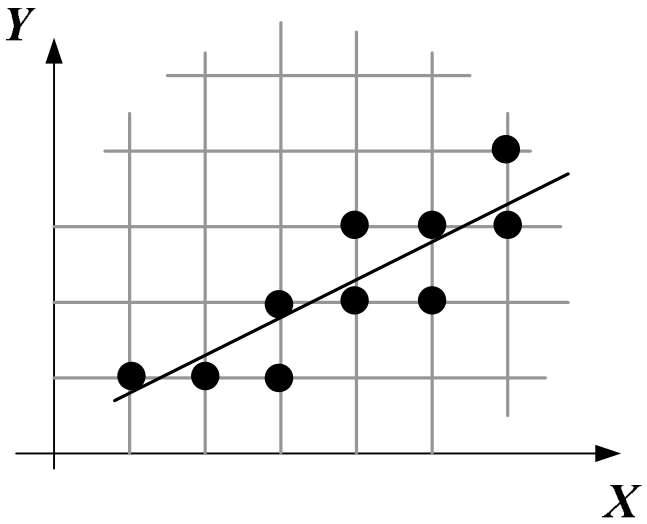


Рис 86. 4-связная линия

4.5.2. Растровое представление отрезка. Алгоритм Брезенхейма

Рассмотрим задачу построения растрового изображения отрезка, соединяющего точки $A(x_a, y_a)$ и $B(x_b, y_b)$. Для простоты будем считать, что

$$0 \le y_b - y_a \le x_b - x_a .$$

Тогда отрезок описывается уравнением

$$y = y_a + \frac{y_b - y_a}{x_b - x_a} \cdot (x - x_a), \quad x \in [x_a, x_b],$$

или иначе

$$y = kx + b,$$

$$k = \frac{y_b - y_a}{x_b - x_a}, \quad b = y_a - kx_a$$

Улучшить внешний вид получаемого растрового отрезка можно за счет округления значений y до ближайшего целого. Фактически это означает, что из двух возможных кандидатов (пикселей, расположенных друг над другом так, что прямая проходит между ними) всегда выбирается тот пиксель, который лежит ближе к изображаемой прямой (закрашенные кружки на рис. 87). Для этого достаточно сравнить дробную часть y со значением *1/2*.

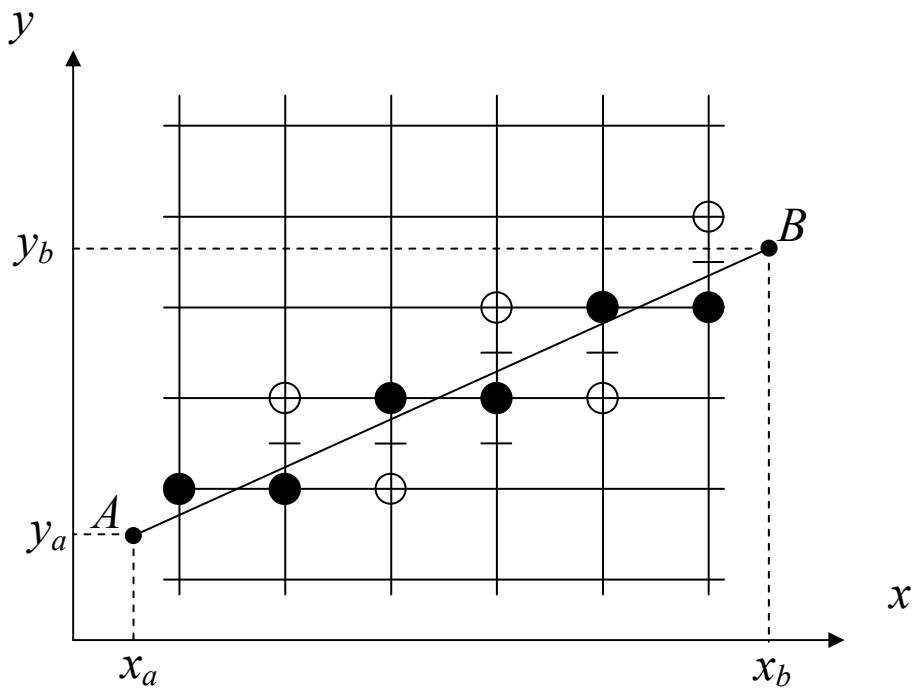


Рис. 87. Отрезок, получаемый в соответствии с алгоритмом Брезенхейма

Пусть $x_0=x_a$, $y_0=y_a$, ..., $x_n=x_b$, $y_n=y_b$ – последовательность изображаемых пикселей, причем $x_i+1-x_i=1$. Тогда каждому значению x_i соответствует число kx_i+b.

Обозначим через c_i дробную часть соответствующего значения функции kx_i+b: $c_i=[kx_i + b]$.

Тогда, если $c_i \le ½$, положим $y_i=[kx_i+b]$**,** в противном случае – $y_i=[kx_i+b]+1$.

Рассмотрим, как изменяется величина c_i при переходе от x_i к следующему значению x_{i+1}.

Само значение функции при этом изменяется на k.

Если $c_i+k \le ½$, то $c_{i+1}=c_i+k$, $y_{i+1}=y_i$.

В противном случае необходимо увеличить y на единицу и тогда приходим следующим соотношениям:

$c_{i+1}=c_i+k-1$, $y_{i+1}= y_i+1$,

так как

$kx_i+b=y_i+c_i$,

$kx_{i+1}+b=y_{i+1}+c_{i+1}$,

а (y_i+1) – целочисленная величина.

Заметим, что $c_0 = 0$, так как точка (x_0 , y_0) лежит на прямой $y=kx+b$.

Замечание. Выбор точки можно трактовать и так: рассматривается середина отрезка между возможными кандидатами и проверяется, где (выше или ниже этой середины) лежит точка пересечения отрезка прямой, после чего выбирается соответствующий пиксель. Это метод срединной точки (midpoint algorithm).

Сравнивать с нулем удобнее, чем с $1/2$, поэтому введем новую вспомогательную величину $d_i=2c-1$, заметив, что, $d_i=2k-1$ (так как $c_1=k$).

Несмотря на то, что и входные данные являются целочисленными величинами и все операции ведутся на целочисленной решетке, алгоритм использует операции с вещественными числами. Чтобы избавиться от необходимости их использования, заметим, что все вещественные числа, присутствующие в алгоритме, являются числами вида $p/\Delta x$, $p \in Z$. Поэтому если помножить величины d_i и k на $\Delta x=x_b-x_a$, то в результате этого останутся только целые числа. Тем самым мы приходим к алгоритму Брезенхейма.

4.6. Алгоритмы вычислительной геометрии

4.6.1. Отсечение отрезка. Алгоритм Сазерленда – Коэна

Необходимость отсечения выводимого изображения по границам некоторой области встречается довольно часто. В простейших ситуациях в качестве такой области, как правило, выступает прямоугольник, например, когда нужно вывести изображение на экран в пределах прямоугольного окна.

Ниже рассматривается достаточно простой и эффективный алгоритм отсечения отрезков по границе произвольного прямоугольника.

Предположим, что у нас есть некоторое окно, внутри которого нарисован отрезок. Т.к. при изменении размера окна возникнет необходимость, чтобы некоторые части отрезка изменялись в соответствии с изменением размера окна, то возникает необходимость применения данного алгоритма.

Четырьмя прямыми вся плоскость разбивается на 9 областей (рис. 88). По отношению к прямоугольнику точки в каждой из этих областей расположены одинаково.

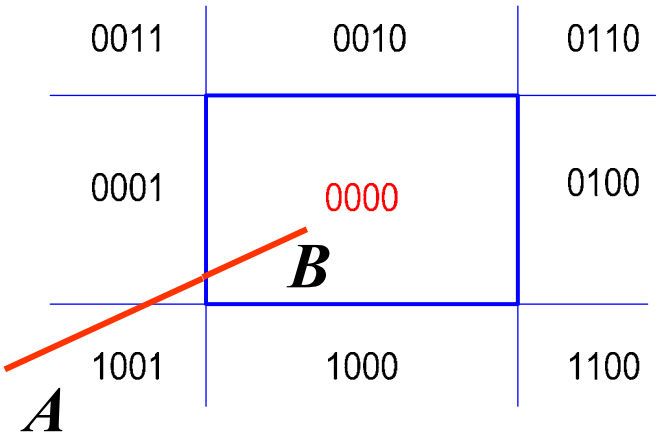


Рис. 88. Последовательность отсечения отрезка по алгоритму Сазерленда-Коэна

Определив, в какие области попали концы рассматриваемого отрезка, легко понять, где именно необходимо произвести отсечение. Для этого каждой области ставится в соответствие 4-битовый код, где установленный бит означает следующее:

бит 0 означает, что точка лежит левее прямоугольника,

бит 1 означает, что точка лежит выше прямоугольника,

бит 2 означает, что точка лежит правее прямоугольника,
бит 3 означает, что точка лежит ниже прямоугольника.

4.5.2. Алгоритм определения принадлежности точки многоугольнику

Одним из методов класса *Polygon* является метод *isInside*, служащий для проверки принадлежности точки многоугольнику.

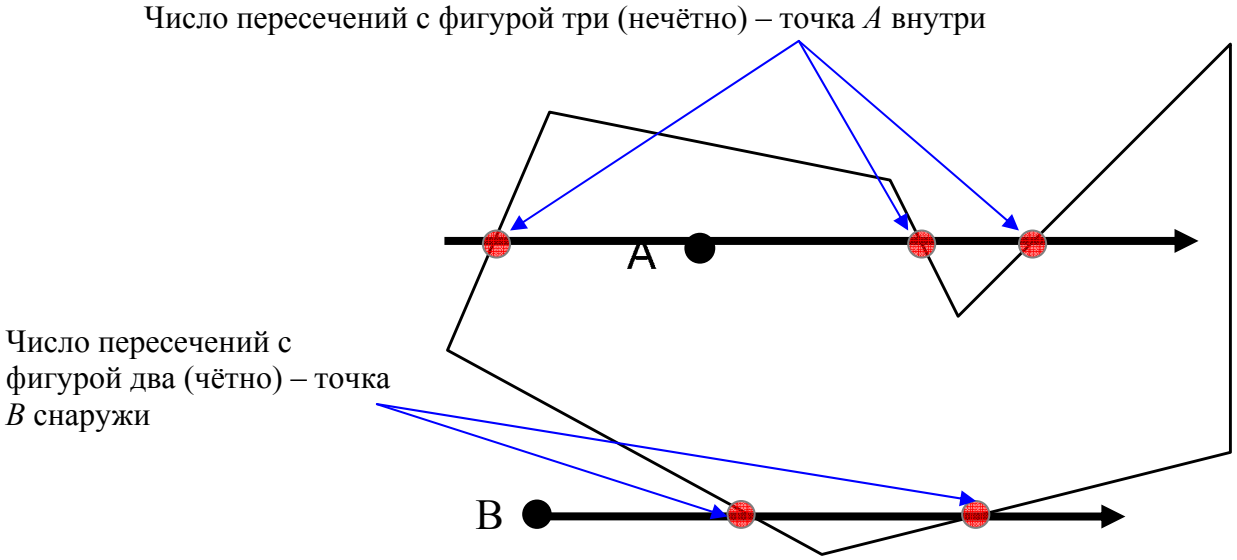


Рис. 89. Примеры случаев принадлежности точки многоугольнику

Для решения этой задачи выпустим из точки *A (x, y)* произвольный луч и найдём количество точек пересечения этого луча с границей многоугольника. Если отбросить случай, когда луч проходит через вершину многоугольника, то решение задачи тривиально – точка лежит внутри, если общее число количество точек пересечения нечетно, и снаружи, если четно (рис. 89).

Ясно, что для любого многоугольника всегда можно построить луч, не проходящий ни через одну из вершин. Однако построение такого луча связано с некоторыми трудностями и, кроме того, проверку пересечения границы многоугольника с произвольным лучом провести сложнее, чем с фиксированным, например горизонтальным.

Возьмем луч, выходящий из произвольной точки *A*, и рассмотрим, к чему может привести прохождение луча через вершину многоугольника. Основные возможные случаи изображены на рис. 90.

В *случае а*, когда ребра, выходящие из соответствующей вершины, лежат по одну сторону от луча, четность количества пересечений не меняется.

Случай б, когда выходящие из вершины ребра лежат по разные стороны от луча, четность количества пересечений изменяется.

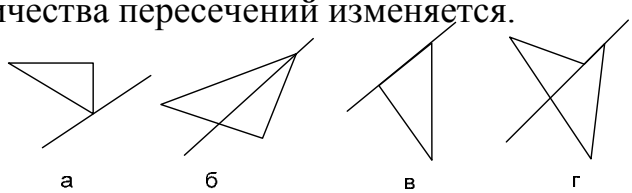


Рис. 90. Возможные варианты прохождения луча через вершину многоугольника

К *случаям в* и *г* такой подход непосредственно неприменим. Несколько изменим его, заметив, что в случаях *а* и *б* вершины, лежащие на луче, являются экстремальными значениями в тройке вершин соответствующих отрезков. В других же случаях экстремума нет.

Исходя из этого, можно построить следующий алгоритм: выпускаем из точки *A* горизонтальный луч в направлении оси *Ox* и все ребра многоугольника, кроме горизонтальных, проверяем на пересечение с этим лучом. В случае, когда луч проходит через вершину, т. е. формально пересекает сразу два ребра, сходящихся в этой вершине, засчитаем это пересечение только для тех ребер, для которых эта вершина является верхней.

4.5.3. Закраска области, заданной цветом границы

Рассмотрим область, ограниченную набором пикселей заданного цвета, и точку (*x*, *y*), лежащую внутри этой области.

Задача заполнения области заданным цветом в случае, когда область не является выпуклой, может оказаться довольно сложной.

Простейший алгоритм, хотя и абсолютно корректно заполняющий даже самые сложные области, является слишком неэффективным, так как для всякого уже отрисованного пикселя функция вызывается еще 3 раза и, кроме того, этот алгоритм требует слишком большого стека из-за большой глубины рекурсии (рис. 91).

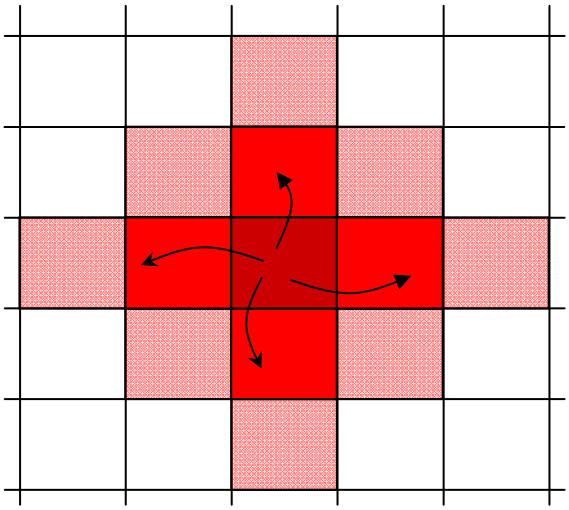


Рис. 91. Алгоритм закраски 4-х соседних пикселей относительно заданного

Поэтому для решения задачи закраски области предпочтительнее алгоритмы, способные обрабатывать сразу целые группы пикселей, т. е. использовать их «связность» – если данный пиксель принадлежит области, то скорее всего его ближайшие соседи также принадлежат данной области.

Ясно, что по заданной точке *A(x, y)* отрезок $[x_l, x_r]$, где $x_l = x_{left}$ – координата левой границы, x_{right} – координата правой границы) максимальной длины, проходящий через эту точку и целиком содержащийся в области, построить

несложно. После заполнения этого отрезка необходимо проверить точки, лежащие непосредственно над и под ним. Если при этом мы найдем незаполненные пиксели, принадлежащие данной области, то для их обработки рекурсивно вызывается функция.

Этот алгоритм способен работать с областями самой сложной формы (рис. 92).

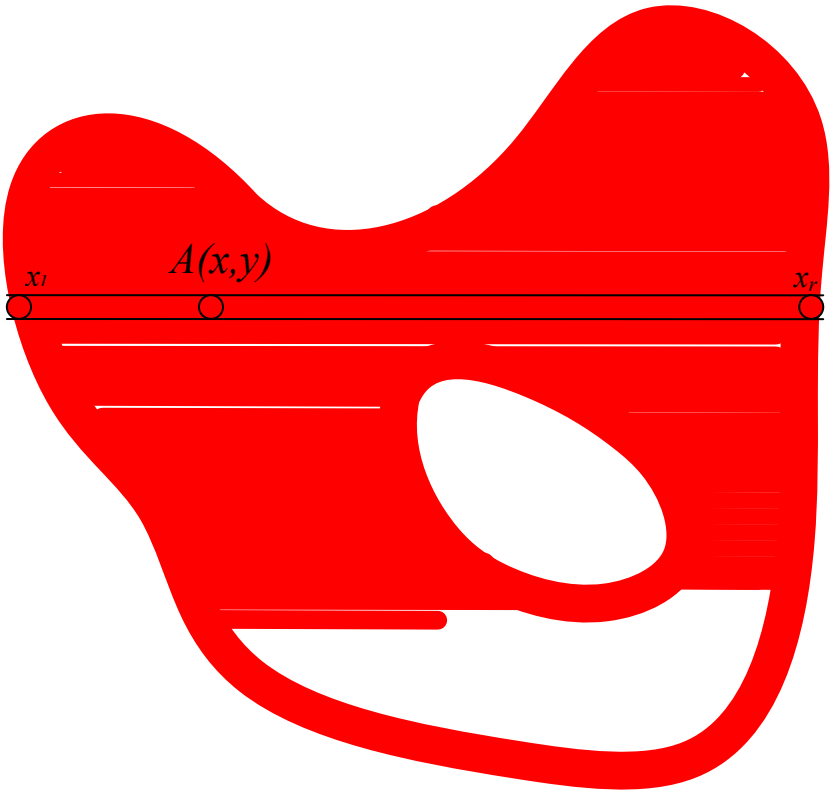


Рис. 92. Закраска сложной области по линейкам пикселей

4.7. Закрашивание

4.7.1. Функция закрашивания

Световая энергия, падающая на поверхность от источника света, может быть поглощена, отражена и пропущена. Количество поглощенной, отраженной и пропущенной энергии зависит от длины световой волны. При этом цвет поверхности объекта определяется поглощаемыми длинами волн.

Свойства отраженного света зависят от формы и направления источника света, а также от ориентации освещаемой поверхности и ее свойств. Свет, отраженный от объекта, может быть диффузным и зеркальным: диффузно отраженный свет рассеивается равномерно по всем направлениям, тогда как зеркальное отражение происходит от внешней поверхности объекта.

Свет точечного источника отражается от идеального рассеивателя по закону косинусов Ламберта:

$$I = I_l k_d \cdot \cos\theta, \quad 0 \le \theta \le \pi/2,$$

где I интенсивность отраженного света;

I_l – интенсивность точечного источника;

k_d – коэффициент диффузного отражения (константа, $0 \le k \le 1$);

θ – угол между направлением на источник света и (внешней) нормалью к поверхности (рис. 93).

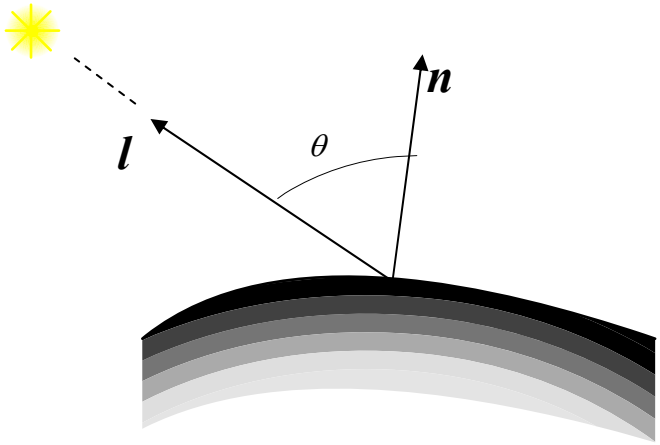


Рис. 93. Пример расположения единичных векторов нормали (вектор ***n***) к освещаемой поверхности, вектора определяющего направление на источник (вектор ***l***) и угла меду ними (угол θ)

На объекты реальных сцен падает еще и рассеянный свет, соответствующий отражению света от других объектов. Поскольку точный расчет рассеянного освещения требует значительных вычислительных затрат, в компьютерной графике при вычислении интенсивности поступают так:

$$I = I_a \cdot k_a + I_l \cdot k_d \cdot \cos\theta, \quad 0 \le \theta \le \pi/2,$$

где I_a – интенсивность рассеянного света;

k_a – постоянный коэффициент диффузного отражения рассеянного света, $0 \le k_a \le 1$.

Интенсивность света, естественно, зависит от расстояния d от объекта до источника света. Для того чтобы учесть еще и этот фактор, пользуются следующей моделью освещения:

$$I = I_a \cdot k_a + \frac{I_l \cdot k_d}{d + k} \cdot \cos\theta,$$

где k произвольная постоянная, обеспечивающая адекватность модели при малых расстояниях до источника освещения.

Интенсивность зеркально отраженного света зависит от угла падения, длины волны и свойств вещества. Так как физические свойства зеркального отражения довольно сложны, то в простых моделях освещения обычно пользуются следующей эмпирической моделью (моделью Фонга):

$$I_s = I_l k_s \cos^p \alpha,$$

где k_s – экспериментальная постоянная;

α – угол между отраженным лучом и вектором наблюдения;

p – степень, аппроксимирующая пространственное распределение света.

Объединяя последние две формулы, получаем окончательно модель (рис. 94) освещения (функцию закраски), используемую для расчета интенсивности (или тона) точек поверхности объекта (т.е. пикселей изображения):

$$I = I_a \cdot k_a + \frac{I_l}{d+k} \cdot (k_d \cdot \cos\theta + k_s \cdot \cos^p \alpha).$$

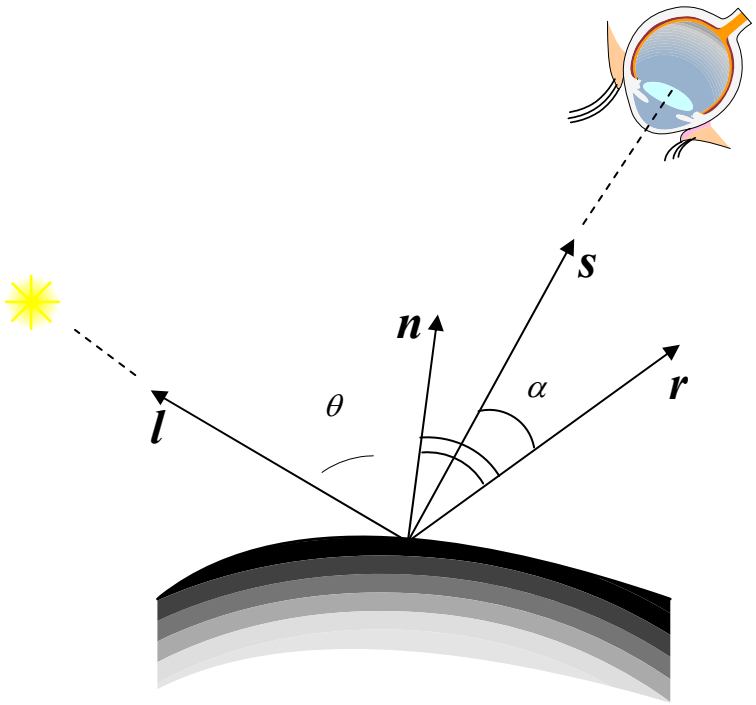


Рис. 94. Модель освещения

Функцию закрашивания, используя единичные векторы внешней нормали ***n***, а также единичные векторы, определяющие направления на источник (вектор ***l***), отраженного луча (вектор ***r***) и наблюдения (вектор ***s***), можно записать в виде скалярных произведений:

$$I = I_a \cdot k_a + \frac{I_l}{d+k} \cdot \left[k_d(\boldsymbol{n},\boldsymbol{l}) + k_s(\boldsymbol{r},\boldsymbol{s})^p\right].$$

Замечание: Чтобы получить цветное изображение, необходимо найти функции закраски для каждого из трех основных цветов красного, зеленого и синего. Поскольку цвет зеркально отраженного света определяется цветом падающего, то постоянная k_s считается одинаковой для каждого из этих цветов.

Если точечных источников света несколько (m), модель освещения может быть представлена следующим образом:

$$I = I_a \cdot k_a + \sum_{j=1}^{m} \frac{I_{l_j}}{d+k} \cdot (k_d \cdot \cos\theta_j + k_s \cdot \cos^{p_j} \alpha_j).$$

Если освещаемая поверхность в рассматриваемой точке гладкая (имеет касательную плоскость), то вектор внешней нормали вычисляется непосредственно. В случае многогранной поверхности векторы внешних нормалей можно найти только для ее граней. Что касается направлений

векторов внешних нормалей на ребрах и в вершинах этой поверхности, то их значения можно найти только приближенно.

Пусть, например, грани, сходящиеся в данной вершине, лежат в плоскостях, описываемых уравнениями:

$$A_i x + B_i y + C_i z + D_i = 0, \quad i = 1, \ldots, m$$

Можно считать, что нормальные векторы этих плоскостей (A_i, B_i, C_i), $i=1,...,m$ являются векторами внешних нормалей для рассматриваемой многогранной поверхности (если какой-то из нормальных векторов не является внешним, то достаточно поменять знаки его координат на противоположные). Складывая эти векторы, получаем вектор, определяющий направление приближенной нормали

$$V_1V_2 \times V_1V_3, \quad V_1V_3 \times V_1V_4, \quad V_1V_4 \times V_1V_2$$

$$(A, B, C) = \sum_{i=1}^{m} (A_i, B_i, C_i)$$

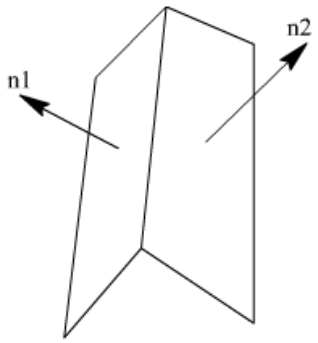


Рис. 95. Представление граней, примыкающих к ребру

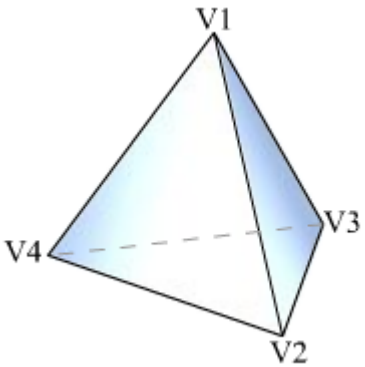


Рис. 96. Многогранная поверхность, заданная вершинами: V_1, V_2, V_3, V_4

Замечание. Для определения направления приближенной нормали в точке, лежащей на ребре многогранной поверхности, достаточно сложить векторы внешних нормалей, примыкающих к этому ребру граней рассматриваемой поверхности. Можно поступить и по-иному. А именно, аппроксимировать переменный вектор нормали вдоль ребра многогранной поверхности при помощи уже найденных векторов внешних нормалей в вершинах, прилегающих к рассматриваемому ребру.

Пусть многогранная поверхность задана своими вершинами. Тогда векторы, определяющие направления приближенных внешних нормалей в ее

вершинах можно найти, используя векторные произведения, построенные на векторах, идущих вдоль ребер, исходящих из соответствующих вершин. Например, для того, чтобы определить внешнюю нормаль в вершине V_1, необходимо сложить векторные произведения.

Замечание. Если перед сложением найденные векторные произведения пронормировать, то полученная сумма будет отличаться от предыдущей и по длине, и по направлению.

Для отыскания направления вектора отражения напомним, что единичные векторы падающего света $\boldsymbol{l}$, нормали к поверхности $\boldsymbol{n}$ и отражения лежат в одной плоскости, причем угол падения равен углу отражения.

Рассмотрим модель освещения с одним точечным источником и предположим, что свет падает вдоль оси Z. Тогда координаты единичного вектора отражения

$$\boldsymbol{r} = (r_x, r_y, r_z)$$

определяются по формулам:

$$r_x = 2n_x n_z$$
$$r_y = 2n_y n_z$$
$$r_z = 2(n_z)^2 - 1,$$

где $\boldsymbol{n} = (n_x, n_y, n_z)$ – единичный вектор нормали к поверхности.

Если же свет от источника, падает не по оси аппликат, то проще всего поступить так: выбрать новую координатную систему так, чтобы ее начало совпадало с рассматриваемой точкой, касательная плоскость к поверхности была плоскостью xy, а нормаль к поверхности в этой точке шла вдоль оси Z. В этой новой системе координаты векторов $\boldsymbol{r}$ и $\boldsymbol{l}$ будут связаны соотношениями:

$$r_x = -l_x, \quad r_y = -l_y, \quad r_z = l_z.$$

Для того чтобы получить исходные координаты вектора отражения, необходимо выполнить обратное преобразование.

4.7.2. Метод постоянного закрашивания

Рассмотрим произвольную сцену, составленную из полигональных (многогранных) фигур. Простейший способ ее построения заключается в том, что на каждой из граней выбирается по точке, для нее определяется освещенность, и затем вся грань закрашивается с использованием найденной освещенности. Предложенный алгоритм, который можно назвать *алгоритмом постоянного закрашивания*, обладает, однако, одним большим недостатком: полученное изображение имеет неестественный многогранный вид. Это объясняется тем, что определяемая подобным образом освещенность сцены не является непрерывной величиной, а имеет кусочно-постоянный характер.

Существуют специальные методы закрашивания, позволяющие создавать иллюзию гладкости. Опишем два известных метода построения сглаженных изображений.

4.7.3. Закрашивание методом Гуро

Наиболее простым из таких методов является метод Гуро (*Gouraud*), который основывается на определении освещенности грани в ее вершинах с последующей билинейной интерполяцией получившихся величин на всю грань.

Обратимся к рис. 97, на котором изображена выпуклая четырехугольная грань. Предположим, что интенсивности в ее вершинах V_1, V_2, V_3 и V_4 известны и равны соответственно $I_{V_1}, I_{V_2}, I_{V_3}, I_{V_4}$.

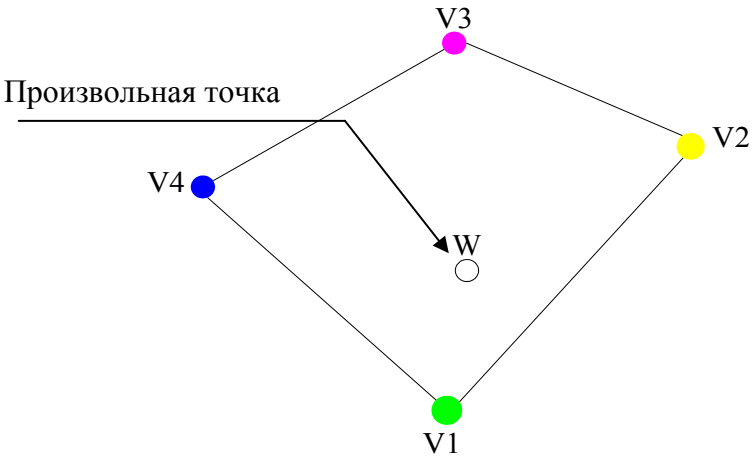


Рис. 97. Интенсивность, заданная в вершинах выпуклой четырехугольной грани

Пусть W – произвольная точка грани. Для определения интенсивности (освещенности) в этой точке проведем через нее горизонтальную прямую. Обозначим через U и V точки пересечения проведенной прямой с границей грани.

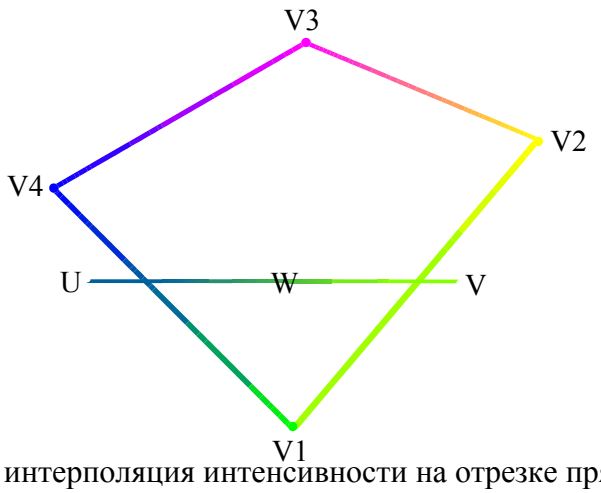


Рис. 98. Линейная интерполяция интенсивности на отрезке прямой, принадлежащей грани, при закрашивании по методу Гуро

Будем считать, что интенсивность на отрезке UV изменяется линейно (рис. 98.), то есть

$$I_W = (1-t)\cdot I_U + t\cdot I_V,$$

где

$$t = \frac{|UW|}{|UV|}, \quad 0 \le t \le 1.$$

Для определения интенсивности в точках U и V вновь воспользуемся линейной интерполяцией, также считая, что вдоль каждого из ребер границы интенсивность изменяется линейно.

Тогда интенсивность в точках U и V вычисляется по формулам

$$I_U = (1-u)\cdot I_{V_4} + u \cdot I_{V_1}$$
$$I_V = (1-v)\cdot I_{V_1} + v \cdot I_{V_2},$$

где

$$u = \frac{|V_4U|}{|V_4V_1|}, \quad 0 \le u \le 1; \quad v = \frac{|V_1V|}{|V_1V_2|}, \quad 0 \le v \le 1.$$

Метод Гуро обеспечивает непрерывное изменение интенсивности при переходе от одной грани к другой без разрывов и скачков, и позволяет определить интенсивность в каждой точке четырехугольной грани (рис. 99).

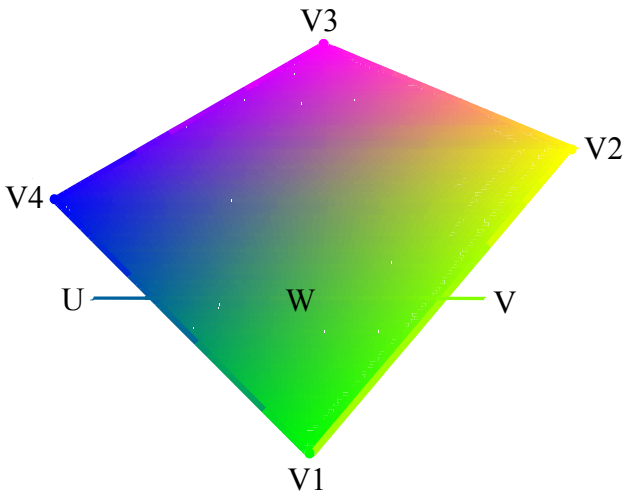


Рис. 99. Выпуклая четырехугольная грань, закрашенная по методу Гуро

Еще одним преимуществом этого метода является его инкрементальный характер: грань рисуется в виде набора горизонтальных отрезков, причем так, что интенсивность последующего пикселя отрезка отличается от интенсивности предыдущего на величину постоянную для данного отрезка. Кроме того, при переходе от отрезка к отрезку значения интенсивности в его концах также изменяются линейно.

Таким образом, процесс рисования грани слагается из следующих шагов:

1) проектирование вершин грани на экран;

2) отыскание интенсивностей в вершинах;

3) определение координат концов очередного отрезка и значений интенсивности в них линейной интерполяцией;

4) рисование отрезка с линейным изменением интенсивности между его концами.

В результате закраска сферы методом Гуро мы получим картину, показанную на рис. 100.

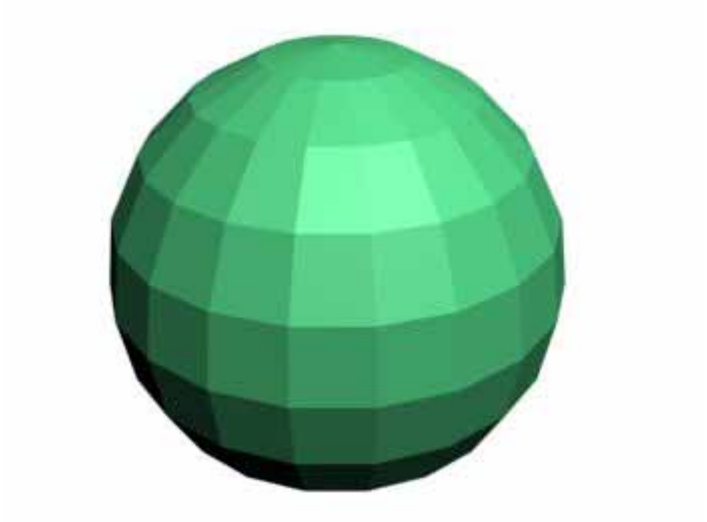


Рис. 100. Пример закраски методом Гуро

4.7.4. Закрашивание методом Фонга

Как и в методе Гуро, в методе Фонга (*Phong*) при закраске используется определение интенсивности и интерполирование, но интерполируется не значение интенсивности по уже известным её значениям в опорных точках, а интерполируется значение вектора внешней нормали, которое далее используется для вычисления значения интенсивности каждого пикселя. Поэтому закраска требует большего объёма вычислений. Зато изображение получается более реалистичным.

Для каждой точки строится вектор внешней нормали или его эквивалент. Затем этот вектор используется для вычисления освещенности в данной точке по формуле:

$$I = I_a \cdot k_a + \frac{I_l}{d+k} \cdot (k_d \cdot \cos\theta + k_s \cdot \cos^p \alpha)$$

Схема интерполяции при закраске Фонга аналогична интерполяции в закраске Гуро, а именно: для определения вектора нормали в точке W – $\boldsymbol{n}_w$, через эту точку проводят горизонтальную прямую

$$\boldsymbol{n}_w = \frac{(1-t)\cdot \boldsymbol{n}_u + t \cdot \boldsymbol{n}_v}{\left|(1\text{-}t)\cdot \boldsymbol{n}_u + t \cdot \boldsymbol{n}_v\right|}, \quad t = \frac{|UW|}{|UV|}.$$

Нормали $\boldsymbol{n_u}$ и $\boldsymbol{n_v}$ определяются также с помощью линейной интерполяции по векторам нормали в концевых точках соответствующих рёбер рассматриваемых граней:

$$\boldsymbol{n}_u = (1-u)\cdot \boldsymbol{n}_{V_4} + u \cdot \boldsymbol{n}_{V_1}, \quad u = \frac{|UV_1|}{|V_4V_1|};$$

$$\boldsymbol{n}_v = (1-v)\cdot \boldsymbol{n}_{V_4} + v \cdot \boldsymbol{n}_{V_2}, \quad v = \frac{|VV_2|}{|V_1V_2|}.$$

В результате закраски сферы методом Фонга мы получим картину, показанную на рис. 101.

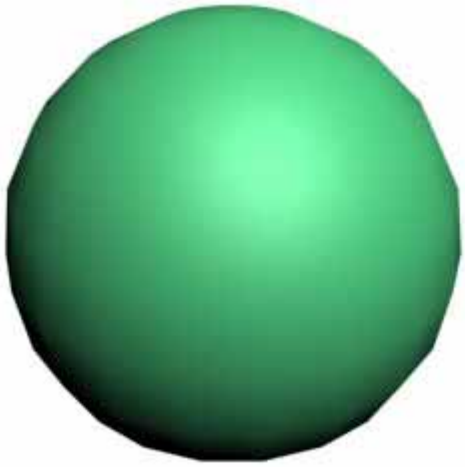


Рис. 101. Пример закраски методом Фонга

4.8. Удаление невидимых линий и поверхностей

Одной из важнейших задач трехмерной графики является следующая: определить, какие части объектов (ребра, грани), находящихся в трехмерном пространстве, будут видны при заданном способе проектирования, а какие будут закрыты от наблюдателя другими объектами. На рис. 102 в качестве примера неправильного удаления невидимых линий показан так называемый куб Эшера, зачастую используемый для демонстрации оптической иллюзии.

В качестве возможных видов проектирования традиционно рассматриваются параллельное и центральное (перспективное). Само проектирование осуществляется на так называемую картинную плоскость (экран): через каждую точку каждого объекта проводится проектирующий луч (проектор) к картинной плоскости. Все проекторы образуют пучок либо параллельных лучей (при параллельном проектировании), либо лучей, выходящих из одной точки (центральное проектирование). Пересечение проектора с картинной плоскостью дает проекцию точки. Видимыми будут только те точки, которые вдоль направления проектирования расположены ближе всего к картинной плоскости.

Несмотря на кажущуюся простоту, задача удаления невидимых линий и поверхностей является достаточно сложной и зачастую требует очень больших объемов вычислений. Поэтому существует целый ряд различных методов решения этой задачи, включая и методы, опирающиеся на аппаратные решения.

Эти методы различаются по следующим основным параметрам:

- способу представления объектов;
- способу визуализации сцены;
- пространству, в котором проводится анализ видимости;
- виду получаемого результата (его точности).

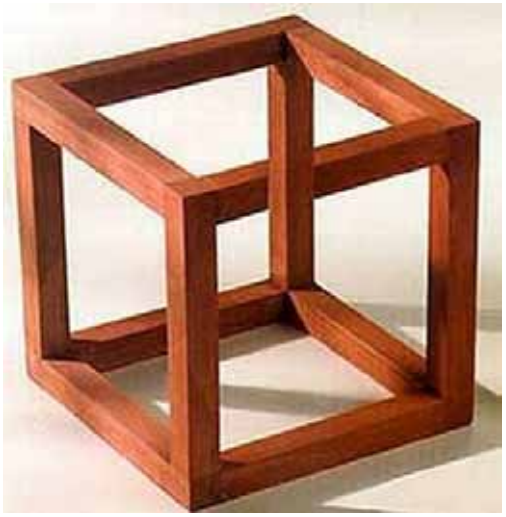


Рис. 102. Куб Эшера как пример неправильного удаления невидимых линий

В качестве возможных способов представления объектов могут выступать аналитические (явные и неявные), параметрические и полигональные. Далее будем считать, что все объекты представлены набором выпуклых плоских граней, например треугольников (полигональный способ), которые могут пересекаться одна с другой только вдоль ребер. Координаты в исходном трехмерном пространстве будем обозначать через (x, y, z), а координаты в картинной плоскости – через (X, Y). Будем также считать, что на картинной плоскости задана целочисленная растровая решетка – множество точек (i,j), где i и j – целые числа.

Классификация методов удаления невидимых линий и поверхностей приведена на рис. 103.

Рис. 103. Классификация методов удаления невидимых линий и поверхностей

Если это не оговорено особо, будем считать для простоты, что проектирование осуществляется на плоскость OXY. Проектирование при этом происходит либо параллельно оси OZ, т. е. задается формулами:

$$X = x,\ Y = y,$$

либо является центральным с центром, расположенным на оси OZ, и задается формулами:

$$X = \frac{x}{z},\quad Y = \frac{y}{z}.$$

Существуют два различных способа изображения трехмерных тел – каркасное (*wireframe* – рисуются только ребра) и сплошное (рисуются закрашенные грани). Тем самым возникают два типа задач – удаление невидимых линий (ребер для каркасных изображений) и удаление невидимых поверхностей (граней для сплошных изображений).

Анализ видимости объектов можно производить как в исходном трехмерном пространстве, так и на картинной плоскости. Это приводит к разделению методов на два класса:

- методы, работающие непосредственно в пространстве самих объектов;
- методы, работающие в пространстве картинной плоскости, т. е. работающие с проекциями объектов.

Получаемый результат представляет собой либо набор видимых областей или отрезков, заданных с машинной точностью (имеет непрерывный вид), либо информацию о ближайшем объекте для каждого пикселя экрана (имеет дискретный вид).

4.8.1. Построение графика функции двух переменных. Линии горизонта

Рассмотрим задачу построения графика функции двух переменных $z=f(x,y)$ в виде сетки координатных линий $x=const$ и $y=const$.

При параллельном проектировании вдоль оси Oz проекцией вертикальной линии в объектном пространстве будет вертикальная линия на картинной плоскости (экране). Легко убедиться в том, что в этом случае точка $p(x,y,z)$ переходит в точку $((p, e_1), (p, e_2))$ на картинной плоскости, где

$$e_1 = [\cos\varphi, \sin\varphi, 0],$$

$$e_2 = [\sin\varphi \cdot \sin\psi, -\cos\varphi \cdot \sin\psi, \cos\psi],$$

а направление проектирования имеет вид:

$$e_3 = [\sin\varphi \cdot \cos\psi, -\cos\varphi \cdot \cos\psi, -\sin\psi],$$

где

$$\varphi \in [0, 2\pi\ ], \quad \psi \in [-\pi/2, \pi/2].$$

Чаще всего применяется полигональный способ представления графика: функция приближается прямоугольной матрицей значений функции в узлах сетки, а сам график представляется наборами ломаных линий, соответствующих постоянным значениям x и y.

Рассмотрим сначала построение графика функции в виде набора линий, соответствующих постоянным значениям y, считая, что углы φ и ψ подобраны таким образом, что при $y_1>y_2$ плоскость $y = y_1$ расположена ближе к картинной плоскости, чем плоскость $y = y_2$.

Каждая линия семейства $z=f(x, y_i)$ лежит в своей плоскости $y = y_i$, причем все эти плоскости параллельны и, следовательно, не могут пересекаться. Из этого следует, что при $y_j > y_i$ линия $z=f(x, y_j)$ не может закрывать собой линию $z=f(x, y_i)$ и, значит, каждая линия $z=f(x,y_j)$ может быть закрыта только предыдущими линиями $z=f(x, y_i)$, $i=1,\ldots,j-1$.

Таким образом, возможен следующий алгоритм построения графика функции *z=f(x,y)*: линии рисуются в порядке удаления (возрастания *y – front-to-back*) и при рисовании очередной линии выводится только та ее часть, которая ранее нарисованными линиями не закрывается.

Для определения частей линии $z = f(x, y_k)$, которые не закрывают ранее нарисованных линий, вводятся так называемые линии горизонта, или контурные линии.

Изначально линии горизонта не инициализированы, поэтому первая линия выводится полностью (так как она ближе всего расположена к наблюдателю, то закрывать ее ничто не может). После этого линии горизонта инициализируются так, что в выводимых точках они совпадают с линией, выведенной первой.

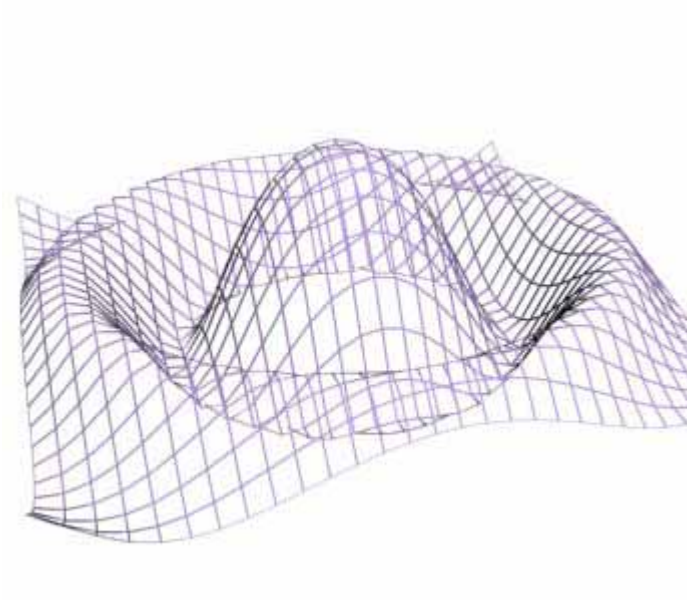


Рис. 104. График двух функций БЕЗ удаления невидимых линий.

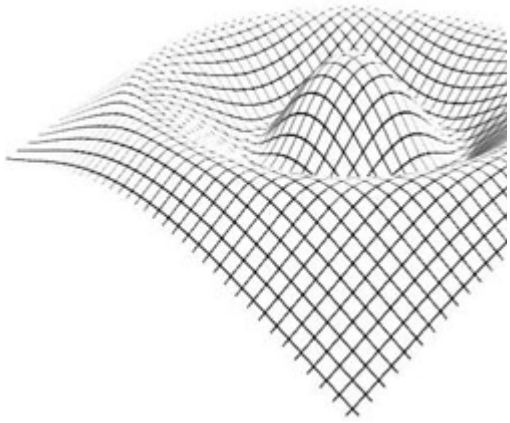


Рис. 105. График двух функций с удаленными невидимыми линиями.

Вторая линия также выводится полностью, и линии горизонта корректируются следующим образом: нижняя линия горизонта в любой из точек равна минимуму из двух уже выведенных линий, верхняя – максимуму. Рассмотрим область экрана между верхней и нижней линиями горизонта – она является проекцией части графика функции, заключенной в полосе $y_1 \le y \le y_2$, и, очевидно, находится ближе, чем все остальные линии вида $z = f(x, y_i)$, $i > 2$. Поэтому те части линий, которые при проектировании попадают в эту область, указанной частью графика закрываются и при данном способе проектирования не видны.

Тем самым следующая линия будет рисоваться только в тех местах, где ее проекция лежит вне области, задаваемой контурными линиями. Пусть проекцией линии $z = f(x, y_k)$ на картинную плоскость является линия $Y=Y_k(X)$, где *(X,Y)* – координаты на картинной плоскости, причем *Y* – вертикальная

координата. Контурные линии $Y^k_{max}(X)$ и $Y^k_{min}(X)$ определяются следующими соотношениями:

$$Y^k_{\max}(X) = \max_{1 \le i \le k-1} Y_i(x),$$

$$Y^k_{\min}(X) = \min_{1 \le i \le k-1} Y_i(x).$$

На экране рисуются только те части линии $Y=Y_k\ (x)$, которые, находятся выше линии $Y^k_{max}(X)$ или ниже линии $Y^k_{min}(X)$.

Такой алгоритм называется методом плавающего горизонта.

4.8.2. Отсечение нелицевых граней

Рассмотрим многогранник, для каждой грани которого задан единичный вектор внешней нормали (рис. 106). Несложно заметить, что если вектор нормали грани $\boldsymbol{n}$ составляет с вектором $\boldsymbol{l}$, задающим направление проектирования, тупой угол (вектор нормали направлен от наблюдателя), то эта грань заведомо не может быть видна. Такие грани называются нелицевыми. Если соответствующий угол является острым, грань называется лицевой.

При параллельном проектировании условие на угол, для того чтобы грань была видна, можно записать в виде неравенства $(\boldsymbol{n},\boldsymbol{l})>0$, поскольку направление проектирования от грани не зависит.

При центральном проектировании с центром в точке c вектор проектирования для точки p будет равен $\boldsymbol{l} = \boldsymbol{c} - \boldsymbol{p}$.

Для определения того, является заданная грань лицевой или нет, достаточно взять произвольную точку p этой грани и проверить выполнение условия:

$$(\boldsymbol{n},\boldsymbol{l})>0.$$

Знак этого скалярного произведения не зависит от выбора точки на грани, а определяется тем, в каком полупространстве относительно плоскости, содержащей данную грань, лежит центр проектирования. Так как при центральном проектировании проектирующий луч зависит от грани (и не зависит от выбора точки на грани), то лицевая грань может стать нелицевой, а нелицевая может стать лицевой даже при параллельном сдвиге. При параллельном проектировании сдвиг не изменяет углов и то, является ли грань лицевой или нет, зависит только от угла между нормалью к грани и направлением проектирования.

Заметим, что если по аналогии с определением принадлежности точки многоугольнику пропустить через произвольную точку картинной плоскости проектирующий луч к объектам сцены, то число пересечений луча с лицевыми гранями будет равняться числу пересечений луча с нелицевыми гранями.

В случае, когда сцена представляет собой один выпуклый многогранник, удаление нелицевых граней полностью решает задачу удаления невидимых граней.

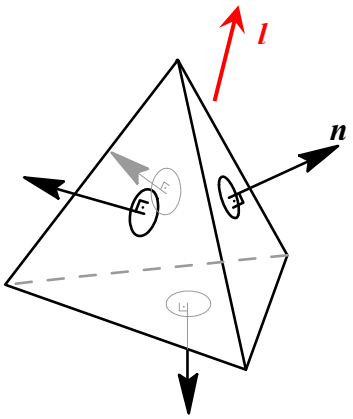


Рис. 106. Многогранник, с заданными для каждой грани, векторами ***l,n***.

Хотя в общем случае предложенный подход и не решает задачи удаления полностью, но, тем не менее, позволяет примерно вдвое сократить количество рассматриваемых граней вследствие того, что нелицевые грани всегда не видны; что же касается лицевых граней, то в общей ситуации части некоторых лицевых граней могут быть закрыты другими лицевыми гранями.

Ребра между нелицевыми гранями также всегда не видны. Однако ребро между лицевой и нелицевой гранями вполне может быть и видимым.

4.8.3. Алгоритм Робертса

Первым алгоритмом удаления невидимых линий был алгоритм Робертса, требующий, чтобы каждая грань была выпуклым многоугольником. Опишем этот алгоритм.

Сначала отбрасываются все ребра, обе определяющие грани которых являются нелицевыми (ни одно из таких ребер заведомо не видно). Следующим шагом является проверка на закрывание каждого из оставшихся ребер со всеми лицевыми гранями многогранника. Возможны следующие случаи:

• грань ребра не закрывает (рис. 107 а);

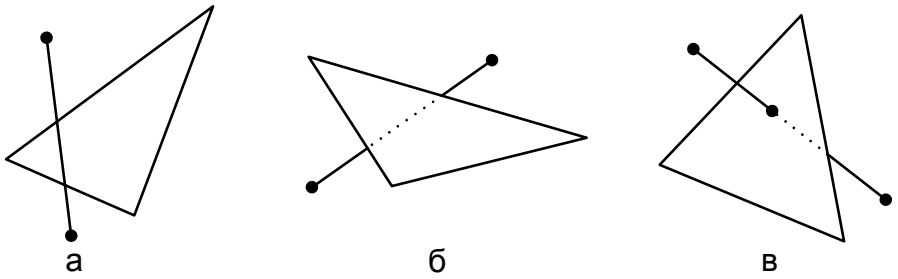


Рис. 107. Варианты расположения ребра и грани

• грань полностью закрывает ребро (тогда оно удаляется из списка рассматриваемых ребер) (рис. 107 б);

• грань частично закрывает ребро (в этом случае ребро разбивается на несколько частей, видимыми из которых являются не более двух; само ребро

удаляется из списка, но в список проверенных ребер добавляются те его части, которые данной гранью не закрываются) (рис. 107 в).

Рассмотрим, как осуществляются эти проверки.

Пусть задано ребро AB, где точка A имеет координаты (x_a, y_a), а точка B – (x_b, y_b).

Прямая, проходящая через отрезок AB, задается уравнениями

$$x = x_a + t \cdot (x_b - x_a)$$
$$y = y_a + t \cdot (y_b - y_a),$$

причем сам отрезок соответствует значениям параметра

$$0 \le t \le 1.$$

Данную прямую можно задать неявным образом как

$$F(x, y) = 0,$$

где

$$F(x, y) = (y_b - y_a) \cdot (x - x_a) - (y_b - x_a) \cdot (y - y_a).$$

Предположим, что проекция грани задается набором проекций вершин $P_1, \ldots, P_k$ с координатами (x_i, y_i), $i = 1,\ldots,n$. Обозначим через F_i значение функции F в точке P_i и рассмотрим i-й отрезок проекции грани P_i, P_i+1. Этот отрезок пересекает прямую AB тогда и только тогда, когда функция F принимает значения разных знаков на концах этого отрезка, а именно при

$$F_i F_{i+1} \le 0.$$

Случай, когда $F_{i+1} = 0$ будем отбрасывать, чтобы дважды не засчитывать прямую, проходящую через вершину, для обоих выходящих из нее отрезков. Итак, мы считаем, что пересечение имеет место в двух случаях:

$$F_i \ge 0, \quad F_{i+1} < 0,$$
$$F_i \le 0, \quad F_{i+1} > 0.$$

Точка пересечения определяется соотношениями

$$x = x_i + s \cdot (x_{i+1} - x_i)$$
$$y = y_i + s \cdot (y_{i+1} - y_i),$$

где $s = \dfrac{F_i}{F_i - F_{i+1}}$.

Отсюда легко находится значение параметра t:

$$t = \frac{x - x_a}{x_b - x_a}, \quad |x_b - x_a| \ge |y_b - y_a|,$$
$$t = \frac{y - y_a}{y_b - y_a}, \quad |y_b - y_a| > |x_b - x_a|.$$

Возможны следующие случаи:

1. Отрезок не имеет пересечения с проекцией грани, кроме, быть может, одной точки. Это может иметь место, когда

• прямая AB не пересекает ребра проекции (рис. 108 а);

• прямая AB пересекает ребра проекций в двух точках t_1 и t_2, но либо $t_1<0$, $t_2<0$, либо $t_1 > 1$, $t_2 > 1$ (рис. 108 б);

• прямая AB проходит через одну вершину, не задевая внутренности треугольника (рис. 108 в).

Очевидно, что в этом случае соответствующая грань никак не может закрывать собой ребро *AB*.

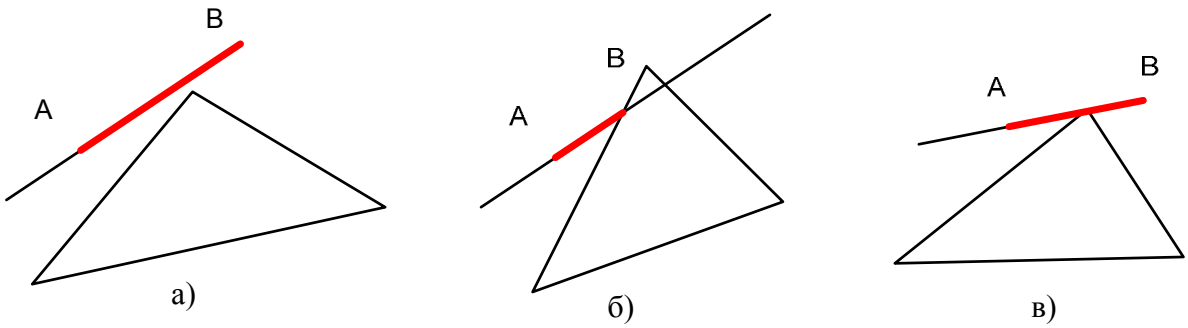
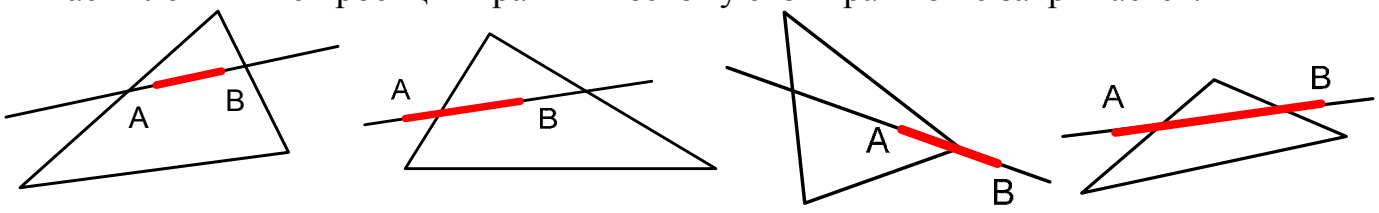


Рис. 108. Пример пересечения отрезка с проекциями граней

2. Проекция ребра полностью содержится внутри проекции грани (рис. 109 а). Тогда есть две точки пересечения прямой *AB* и границы грани и $t_1<0<1<t_2$. Если грань, лежит ближе к картинной плоскости, чем ребро, то ребро полностью невидимо и удаляется.

3. Прямая *AB* пересекает ребра проекции грани в двух точках и либо $t_1<0\le t_2\le 1$, либо $0<t_1\le 1<t_2$ (рис. 109 б и в).

Если ребро *AB* находится дальше от картинной плоскости, чем соответствующая грань, то оно разбивается на две части, одна из которых полностью закрывается гранью и потому отбрасывается. Проекция второй части лежит вне проекции грани и поэтому этой гранью не закрывается.



а б в г

Рис. 109. Пример пересечения отрезка с проекциями граней

1. Прямая *AB* пересекает ребра проекции грани в двух точках, причем (рис. 109 г) $0<t_1<t_2<1$.

Если ребро *AB* лежит дальше от картинной плоскости, чем соответствующая грань, то оно разбивается на три части, средняя из которых отбрасывается.

Для определения того, что лежит ближе к картинной плоскости – отрезок *AB* (проекция которого лежит в проекции грани) или сама грань, через эту грань проводится плоскость

$$(\boldsymbol{n}, \boldsymbol{p}) + c = 0$$

(где $\boldsymbol{n}$ – нормальный вектор грани), разбивающая все пространство на два полупространства. Если оба конца отрезка *AB* лежат в том же полупространстве, в котором находится и наблюдатель, то отрезок лежит ближе к грани; если оба конца находятся в другом полупространстве, то

отрезок лежит дальше. Случай, когда концы лежат в разных полупространствах, здесь невозможен (это означало бы, что отрезок AB пересекает внутреннюю часть грани).

Если общее количество граней равно n, то временные затраты для данного алгоритма составляют $O(n^2)$.

Количество проверок можно заметно сократить, если воспользоваться разбиением картинной плоскости.

Разобьем видимую часть картинной плоскости (экран) на $N_1 \cdot N_2$ равных частей (клеток) и для каждой клетки A_{ij} построим список всех лицевых граней, чьи проекции имеют с данной клеткой непустое пересечение.

Для проверки произвольного ребра на пересечение с гранями отберем сначала все те клетки, которые проекция данного ребра пересекает. Ясно, что проверять на пересечение с ребром имеет смысл только те грани, которые содержатся в списках этих клеток.

В качестве шага разбиения обычно выбирается $O(l)$, где l – характерный размер ребра в сцене. Для любого ребра количество проверяемых граней практически не зависит от общего числа граней и совокупные временные затраты алгоритма на проверку пересечений составляют $O(n)$, где n – количество ребер в сцене.

Поскольку процесс построения списков заключается в переборе всех граней, их проектировании и определении клеток, в которые попадают проекции, то затраты на составление всех списков также составляют $O(n)$.

4.8.4. Алгоритм Аппеля. Количественная невидимость

Рассмотрим произвольное гладкое выпуклое тело в пространстве. Взяв произвольную точку P на границе этого тела, назовем ее лицевой, если $(\boldsymbol{n},\boldsymbol{l})>0$, где $\boldsymbol{n}$ – вектор внешней нормали к границе в этой точке. Если же $(\boldsymbol{n},\boldsymbol{l})<0$, то данная точка является нелицевой (и, соответственно, невидимой). Это определение подобно определению лицевой и нелицевой грани (п.4.8.2.). В силу гладкости поверхности у лицевой (нелицевой) точки существует достаточно малая окрестность, целиком состоящая из лицевых (нелицевых) точек и проектирующаяся на картинную плоскость взаимно однозначно (не закрывая саму себя) (рис. 110).

У точек, для которых $(\boldsymbol{n},\boldsymbol{l}) = 0$, подобной окрестности (состоящей только из лицевых или только нелицевых точек) может не существовать. Такие точки, в отличие от (регулярных) точек называются нерегулярными (особыми) точками проектирования (рис. 111).

В общем случае множество всех нерегулярных точек $(\boldsymbol{n},\boldsymbol{l}) = 0$ образует на поверхности рассматриваемого объекта гладкую кривую, называемую контурной линией. Эта линия разбивает поверхность выпуклого тела на две части, каждая из которых однозначно проектируется на картинную плоскость и целиком состоит из регулярных точек. Одна из этих частей является полностью видимой, а другая – полностью невидимой.

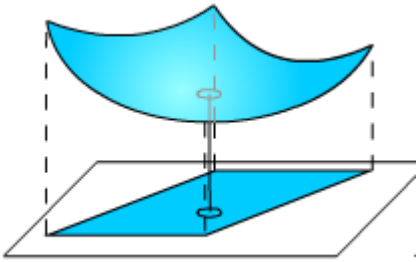


Рис. 110. Пример регулярных точек

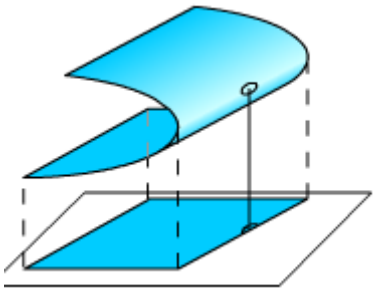


Рис. 111. Пример НЕрегулярных точек

Попытаемся обобщить этот подход на случай одного или нескольких невыпуклых тел.

Множество всех контурных линий (их уже может быть несколько) разбивает границы тел на набор связных частей (компонент), каждая из которых по-прежнему взаимнооднозначно проектируется на картинную плоскость и состоит либо из лицевых, либо из нелицевых точек.

Никакая из этих частей не может закрывать себя при проектировании, однако возможны случаи, когда одна такая часть закрывает другую. Чтобы это учесть, введем числовую характеристику невидимости – так называемую количественную невидимость точки, определив ее как количество точек, закрывающих при проектировании данную точку. Точка оказывается видимой только в том случае, когда ее количественная невидимость равна нулю.

Количественная невидимость является кусочно-постоянной функцией и может изменять свое значение лишь в тех точках, проекции которых на картинную плоскость лежат в проекции одной из контурных линий.

Итак, проекции контурных линий разбивают картинную плоскость на области, каждая из которых является проекцией части объекта, а сами поверхности, ограничивающие тела, разбиваются контурными линиями на однозначно проектирующиеся фрагменты с постоянной количественной невидимостью.

В общем случае при проектировании гладких поверхностей возникает два основных типа особенностей (все остальные возможные особенности могут быть приведены к ним сколь угодно малыми «шевелениями»): 1) линии складки, являющиеся регулярными проекциями контурных линий на картинную плоскость и представляющие собой регулярные кривые на поверхности, взаимно однозначно проектирующиеся на картинную плоскость (рис. 110), и 2) изолированные точки сборки, которые лежат на линиях складки (контурных линиях) и являются особыми точками проектирования контурных линий на картинную плоскость.

Рассмотрим теперь изменение количественной невидимости вдоль самой контурной линии.

Можно показать, что она может измениться только в двух случаях – при прохождении позади контурной линии и в точках сборки. В первом случае происходит загораживание складкой другого участка поверхности, и количественная невидимость изменяется на два. Во втором случае происходит

загораживание поверхностью самой себя, и количественная невидимость изменяется на единицу.

Таким образом, для определения видимости достаточно найти контурные линии и их проекциями разбить всю картинную плоскость на области, являющиеся видимыми частями проекций объектов сцены.

В результате мы приходим к следующему алгоритму: на границах тел выделяется множество контурных линий *C*. Каждая из этих линий разбивается на части в тех точках, где она закрывается при проектировании на картинную плоскость какой-либо линией этого множества, проходящей в точке закрывания ближе к картинной плоскости. Контурные линии разбиваются и в точках сборки. В результате получается множество линий, на каждой из которых количественная невидимость одна и та же (постоянна).

В случае, когда рассматриваются поверхности, не являющиеся границами сплошных тел, к множеству контурных линий *C* следует добавить и граничные линии этих поверхностей.

Если рассматриваемые объекты являются лишь кусочно-гладкими, то к множеству линий *C* следует добавить также линии излома (линии, где происходит потеря гладкости).

Данный метод может быть применен и для решения задачи удаления невидимых поверхностей – получившиеся линии разбивают картинную плоскость на области, каждая из которых соответствует видимой части одного из объектов сцены.

Метод с успехом можно применить и для работы с полигональными объектами. Одним из вариантов такого применения является *алгоритм Аппеля*.

Аппель вводит понятие *количественной невидимости QI* (*quontitative invisibility*) точки как число лицевых граней, ее закрывающих. Это несколько отличается от определения, введенного ранее, однако существо подхода остается неизменным.

Контурная линия полигонального объекта состоит из тех ребер, для которых одна из проходящих граней является лицевой, а другая – нелицевой.

Так, для многогранника на рис. 112 контурной линией является ломаная *ABCIJDEKLGA*.

Рассмотрим, как меняется количественная невидимость вдоль ребра. Для определения видимости ребер произвольного многогранника сначала берется какая-либо его вершина и ее количественная невидимость определяется непосредственно.

Далее прослеживается изменение количественной невидимости вдоль каждого из ребер, выходящих из этой вершины. Эти ребра проверяются на прохождение позади контурной линии, и их количественная невидимость в соответствующих точках изменяется. При прохождении ребра позади контурной линии количественная невидимость точек ребра изменяется на единицу. Те части отрезка, для которых количественная невидимость равна нулю, сразу же рисуются.

Следующим шагом является определение количественной невидимости для ребер, выходящих из новой вершины, и т. д.

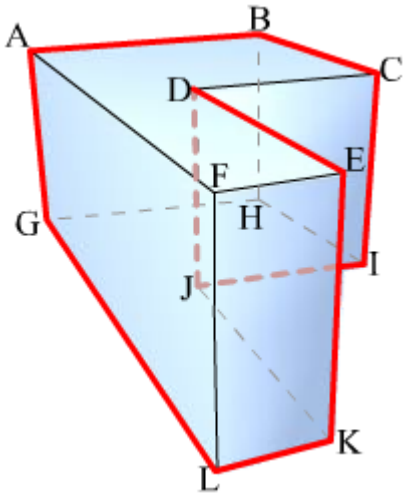


Рис. 112. Пример контурной линии

В результате определяется количественная невидимость всех ребер связной компоненты сцены, содержащей исходную вершину (и при этом видимые части ребер этой компоненты сразу же рисуются).

В случае, когда рассматривается изменение количественной невидимости вдоль ребра, выходящего из вершины, принадлежащей контурной линии, необходимо проверить, не закрывается ли это ребро одной из граней, выходящей из этой вершины (как, например, грань *DEKJ* закрывает ребро *DJ*, и это является аналогом точки сборки).

Так как для реальных объектов количество ребер, входящих в контурную линию, намного меньше общего числа ребер (если общее количество ребер равно n, то количество ребер, входящих в контурную линию, – $O(\sqrt{n})$), алгоритм Аппеля является более эффективным, чем алгоритм Робертса.

На поверхности многогранников можно выделить набор линий такой, что каждая контурная линия независимо от направления проектирования обязательно пересечет хотя бы одну из линий этого набора. Таким образом, для отыскания контурных линий необязательно перебирать все ребра – достаточно проверить заданный набор и восстановить контурные линии от точек пересечения с этим набором.

Замечание. Для повышения эффективности данного алгоритма возможно использование разбиения картинной плоскости – для каждой клетки разбиения строится список ребер контурной линии, чьи проекции пересекают данную клетку.

4.9. Удаление невидимых граней

Задача удаления невидимых граней является заметно более сложной, чем задача удаления невидимых линий, хотя бы по общему объему возникающей информации. Если практически все методы, служащие для удаления невидимых линий, работают в объектном пространстве и дают точный результат, то для удаления невидимых поверхностей существует большое

число методов, работающих только в картинной плоскости, а также смешанных методов.

4.9.1. Метод трассировки лучей

Наиболее естественным методом для определения видимости граней является метод трассировки лучей (вариант, используемый только для определения видимости, без отслеживания отраженных и преломленных лучей обычно называется *ray casting*), при котором для каждого пикселя картинной плоскости определяется ближайшая к нему грань, для чего через этот пиксель выпускается луч, находятся все точки его пересечения с гранями и среди них выбирается ближайшая.

Одним из преимуществ этого метода является простота, универсальность (он может легко работать не только с полигональными моделями; возможно использование *Constructive Solid Geometry*) и возможность совмещения определения видимости с расчетом цвета пикселя.

Еще одним несомненным плюсом метода является большое количество методов оптимизации, позволяющих работать с сотнями тысяч граней и обеспечивающих временные затраты порядка $O(C \cdot log(n))$, где C – общее количество пикселей па экране, а n – общее количество объектов в сцене. Более того, существуют методы, обеспечивающие практическую независимость временных затрат от количества объектов.

4.9.2. Метод *z*-буфера

Одним из самых простых алгоритмов удаления невидимых граней и поверхностей является метод *z*-буфера (буфера глубины), где для каждого пикселя, как и в методе трассировки лучей, находится грань, ближайшая к нему вдоль направления проектирования, однако здесь циклы по пикселям и по объектам меняются местами.

Поставим в соответствие каждому пикселю (x, y) картинной плоскости кроме цвета $c(x, y)$, хранящегося в видеопамяти, его расстояние до картинной плоскости вдоль направления проектирования $z(x, y)$ (его глубину).

Массив глубин инициализируется $+\infty$, т.е. изначально (при отсутствии объектов сцены) в *z*-буфер записаны +1, соответствующие нормированной $+\infty$. Для вывода на картинную плоскость произвольной грани она переводится в растровое представление на картинной плоскости и затем для каждого пикселя этой грани находится его глубина. В случае если эта глубина меньше значения глубины, хранящегося в *z*-буфере, пиксель рисуется и его глубина (в долях единицы) заносится в *z*-буфер.

Метод *z*-буфера работает исключительно в пространстве картинной плоскости и не требует никакой предварительной обработки данных (рис. 113). Порядок, в котором грани выводятся на экран, не играет никакой роли.

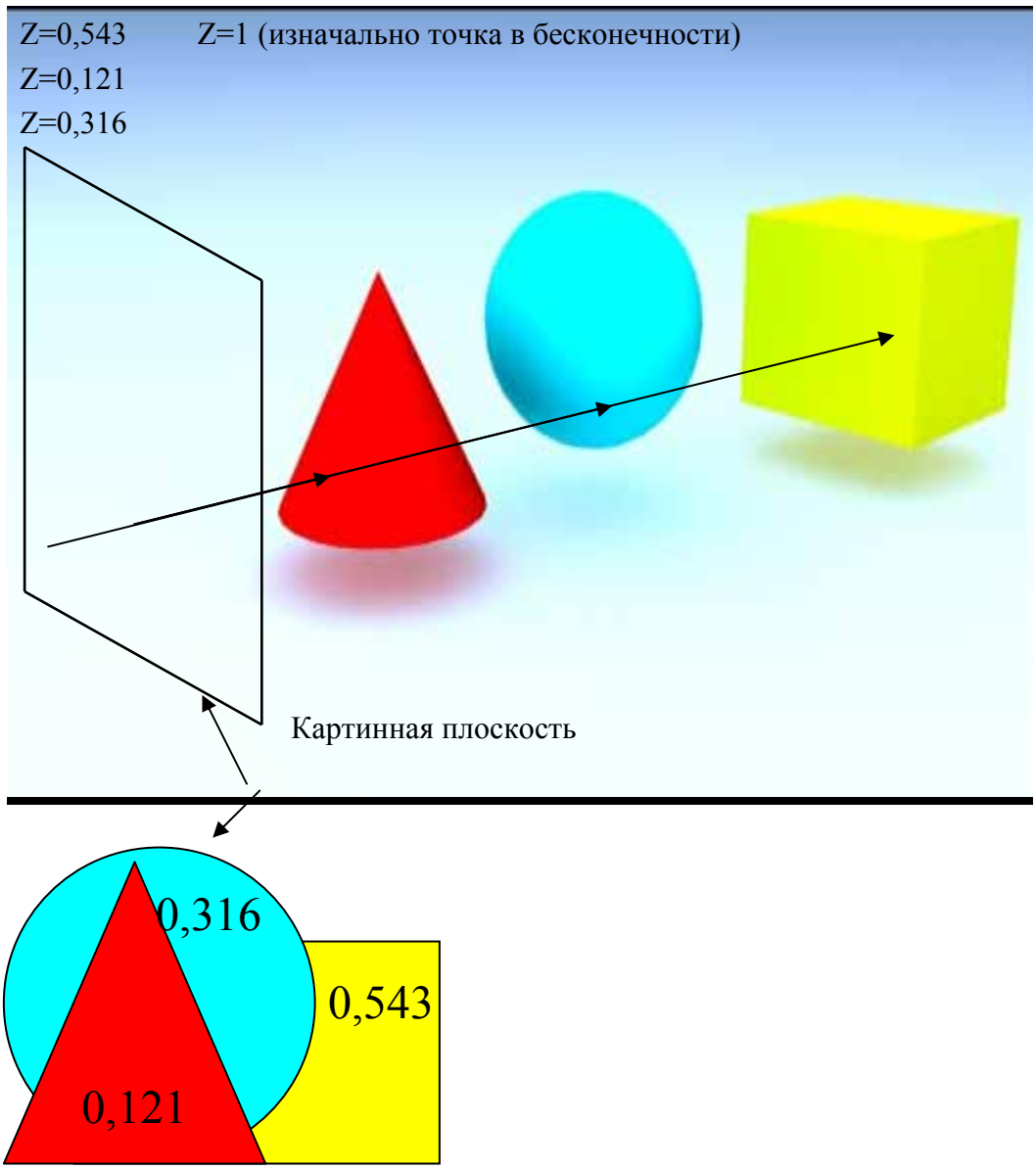


Рис. 113. Удаление невидимых граней на картинной плоскости методом *z*-буфера

Для экономии памяти можно отрисовывать не все изображение сразу, а рисовать по частям. Для этого картинная плоскость разбивается на части (обычно это горизонтальные полосы) и каждая такая часть обрабатывается независимо. Размер памяти под буфер определяется размером наибольшей из этих частей.

Одним из основных недостатков *z*-буфера (помимо большого объема требуемой под буфер памяти) является избыточность вычислений: осуществляется вывод всех граней вне зависимости от того, видны они или нет. И если, например, данный пиксель накрывается десятью различными лицевыми гранями, то для каждого соответствующего пикселя каждой из этих десяти граней необходимо произвести расчет цвета. При использовании сложных моделей освещенности (например, модели Фонга) и текстур эти вычисления могут потребовать слишком больших временных затрат.

4.9.3. Алгоритмы упорядочения

Подход, использованный ранее для построения графика функции двух переменных и основанный на последовательном выводе на экран в определенном порядке всех граней, может быть успешно использован и для построения более сложных сцен.

Тем самым методы упорядочения выносят сравнение по глубине за пределы циклов и производят сортировку граней явным образом. Методы упорядочения являются гибридными методами, осуществляющими сравнение и разбиение граней в объектном пространстве, а для непосредственного наложения одной грани на другую использующими растровые свойства дисплея.

Упорядочим все лицевые грани таким образом, чтобы при их выводе в этом порядке получалось корректное изображение сцены. Для этого необходимо, чтобы для любых двух граней P и Q та из них, которая при выводе может закрывать другую, выводилась позже. Такое упорядочение обычно называется *back-to-front*, поскольку сначала выводятся более далекие грани, а затем более близкие.

Существуют различные методы построения подобного упорядочения. Вместе с тем нередки и случаи, когда заданные грани упорядочить нельзя (рис. 114). Тогда необходимо произвести дополнительное разбиение граней так, чтобы получившееся после разбиения множество граней уже можно было упорядочить.

Заметим, что две любые выпуклые грани, не имеющие общих внутренних точек, можно упорядочить всегда. Для невыпуклых граней это в общем случае неверно.

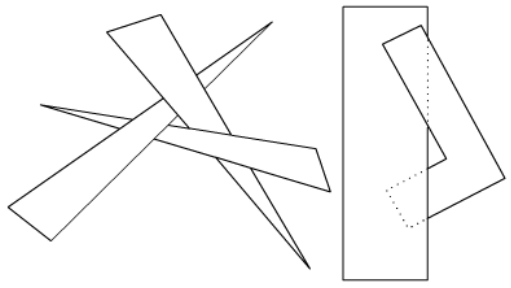


Рис. 114. Пример изображения, которое нельзя упорядочить

Метод сортировки по глубине. Алгоритм художника

Этот метод является самым простым из методов, основанных на упорядочении граней. Как художник сначала рисует более далекие объекты, а затем поверх них более близкие, так и метод сортировки по глубине сначала упорядочивает грани по мере приближения к наблюдателю, а затем выводит их в этом порядке.

Метод основывается на следующем простом наблюдении: если для двух граней A и B самая дальняя точка грани A ближе к наблюдателю (картинной

плоскости), чем самая ближняя точка грани *B*, то грань *B* никак не может закрыть грань *A* от наблюдателя.

Поэтому если заранее известно, что для любых двух лицевых граней ближайшая точка одной из них находится дальше, чем самая дальняя точка другой, то для упорядочения граней достаточно просто отсортировать их по расстоянию от наблюдателя (картинной плоскости).

Однако такое не всегда возможно: могут встретиться такие пары граней, что самая дальняя точка одной находится к наблюдателю не ближе, чем самая близкая точка другой.

На практике часто встречается следующая реализация этого алгоритма: множество всех лицевых граней сортируется по ближайшему расстоянию до картинной плоскости (наблюдателя) и потом эти грани выводятся в порядке приближения к наблюдателю. В качестве алгоритмов сортировки можно использовать либо быструю сортировку, либо поразрядную.

Метод хорошо работает для целого ряда типичных сцен, включая, например, построение изображения нескольких непересекающихся простых тел.

Расстояние от точки *p* до наблюдателя, расположенного в начале координат, при параллельном проектировании вдоль единичного вектора *l* задается следующей формулой:

$$d = (\mathbf{p}, \mathbf{l}).$$

Хотя подобный подход и работает в подавляющем большинстве случаев, однако возможны ситуации, когда просто сортировка по расстоянию до картинной плоскости не обеспечивает правильного упорядочения граней (рис. 115), – так, грань *B* будет ошибочно выведена раньше, чем грань *A*; поэтому после сортировки желательно проверить порядок, в котором грани будут выводиться.

Предлагается следующий алгоритм этой проверки. Для простоты будем считать, что рассматривается параллельное проектирование вдоль оси *Oz*. Перед выводом очередной грани *P* следует убедиться, что никакая другая грань *Q*, которая стоит в списке позже, чем *P*, и проекция которой на ось *Oz* пересекается с проекцией грани *P* (если пересечения нет, то порядок вывода *P* и *Q* определен однозначно), не может закрываться гранью *P*. В этом случае грань *P* действительно должна быть выведена раньше, чем грань *Q*. Ниже приведены 4 теста в порядке возрастания сложности проверки:

1. Пересекаются ли проекции этих граней на ось *Ox*?
2. Пересекаются ли проекции этих граней на ось *Oy*?

Если хотя бы на один из этих двух вопросов получен отрицательный ответ, то проекции граней *P* и *Q* на картинную плоскость не пересекаются и, следовательно, порядок, в котором они выводятся, не имеет значения. Поэтому будем считать, что грани *P* и *Q* упорядочены верно. Для проверки выполнения этих условий очень удобно использовать ограничивающие тела. В случае, когда оба эти теста дали утвердительный ответ, проводятся следующие тесты.

1. Находятся ли грань *P* и наблюдатель по разные стороны от плоскости, проходящей через грань *Q* (рис. 116)?

4. Находятся ли грань *Q* и наблюдатель по одну сторону от плоскости, проходящей через грань *P* (рис. 117) ?

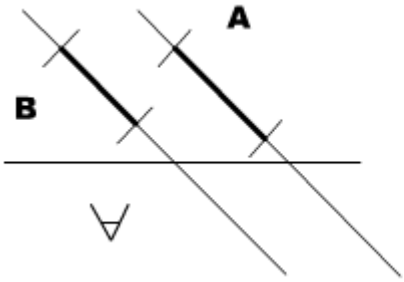


Рис. 115.

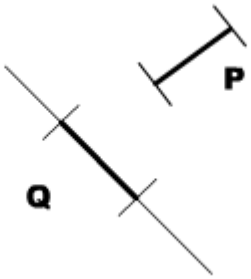


Рис. 116

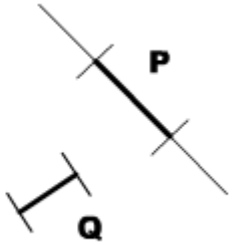


Рис. 117

Примеры проверки последовательности вывода граней

Если хотя бы на один из этих вопросов получен утвердительный ответ, то считаем, что грани *P* и *Q* упорядочены верно, и сравниваем *P* со следующей гранью.

В случае, если ни один из тестов не подтвердил правильность упорядочения граней *P* и *Q*, проверяем, не следует ли поменять эти грани местами. Для этого проводятся тесты, являющиеся аналогами тестов 3 и 4 (очевидно, что снова проводить тесты 1 и 2 не имеет смысла):

3'. Находятся ли грань *Q* и наблюдатель по разные стороны от плоскости, проходящей через грань *P*?

4'. Находятся ли грань *P* и наблюдатель по одну сторону от плоскости, проходящей через грань *Q*?

В случае, если ни один из тестов 3, 4, 3', 4' не позволяет с уверенностью определить, какую из этих двух граней нужно выводить раньше, одна из них разбивается плоскостью, проходящей через другую грань и вопрос об упорядочении целой грани и частей разбитой грани легко решается при помощи тестов 3 или 4 (3' или 4').

Возможны ситуации, когда несмотря на го, что грани *P* и *Q* упорядочены верно, их разбиение все же будет произведено (алгоритм создает избыточные разбиения). Подобный случай изображен на рис. 118, где для каждой вершины указана ее глубина.

Методу упорядочения присущ тот же недостаток, что и методу *z*-буфера, а именно необходимость вывода всех лицевых граней. Чтобы избежать этого, можно его модифицировать следующим образом: грани выводятся в обратном порядке – начиная с самых близких граней и заканчивая самыми далекими (*front-to-back*). При выводе очередной грани рисуются только те пиксели, которые еще не были выведены. Как только весь экран будет заполнен, вывод граней можно прекратить.

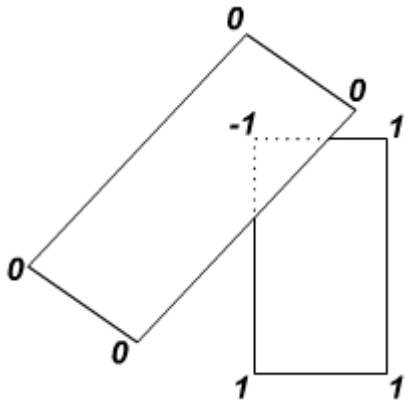


Рис. 118. Пример упорядочения граней по глубине

Метод двоичного разбиения пространства

Существует другой, довольно элегантный и гибкий способ упорядочения граней. Каждая плоскость в объектном пространстве разбивает все пространство на два полупространства. Считая, что эта плоскость не пересекает ни одну из граней сцены, получаем разбиение множества всех граней на два непересекающихся множества (кластера); каждая грань попадает в тот или иной кластер в зависимости от того, в каком полупространстве относительно плоскости разбиения эта грань находится.

Ясно, что ни одна из граней, лежащих в полупространстве, не содержащем наблюдателя, не может закрывать собой ни одну из граней, лежащих в том же полупространстве, в котором находится и наблюдатель (с небольшими изменениями это работает и для параллельного проектирования).

Для построения правильного изображения сцены необходимо сначала выводить грани из дальнего кластера, а затем из ближнего. Применим предложенный подход для упорядочения граней внутри каждого кластера. Для этого выберем две плоскости, разбивающие каждый из кластеров на два подкластера. Повторяя описанный процесс до тех пор, пока в каждом получившемся кластере останется не более одной грани (рис. 119).

Получаем в результате двоичное дерево (*Binary Space Partitioning Tree*). Узлами этого дерева являются плоскости, производящие разбиение.

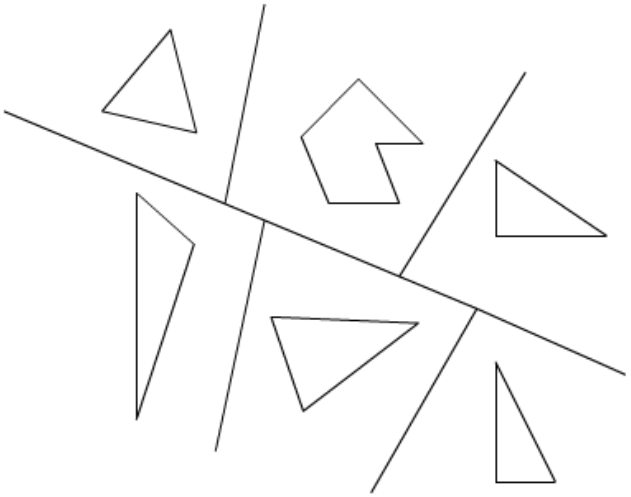


Рис. 119. Пример двоичного разбиения пространства

Пусть плоскость, производящая разбиение, задается уравнением $(p, n) = d$. Тогда если оказывается, что $(p,n)>d$, то это соответствует вершине поддерева, содержащейся в положительном полупространстве, а если $(p,n)<d$, то – в отрицательном полупространстве. Обычно в качестве разбивающей плоскости выбирается плоскость, проходящая через одну из граней. Все грани, пересекаемые этой плоскостью, разбиваются вдоль нее, а получившиеся при разбиении части помещаются в соответствующие поддеревья.

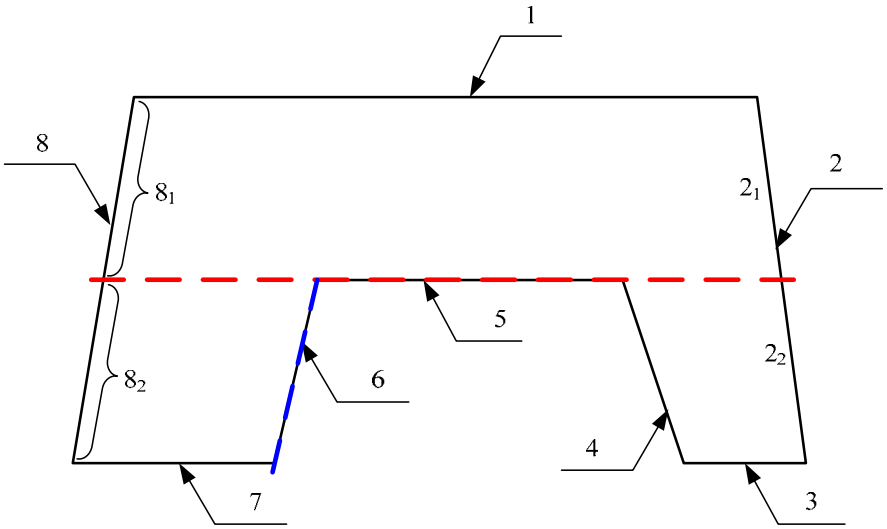


Рис. 120. Разбиение граней на кластеры

Пример. Рассмотрим сцену, показанную на рис. 120. Плоскость, проходящая через грань 5, разбивает грань 2 на части 2_1 и 2_2, а грань 8 – на части 8_1 и 8_2. При этом все множество образовавшихся граней распадается на два кластера: (1, 8_1, 2_1) и (2_2, 3, 4, 6, 7, 8_2). Выбрав для первого кластера новую плоскость разбиения, проходящую через грань 6, получим два подкластера: (7, 8_2) и (2_2, 3, 4). Каждое следующее разбиение будет выделять по одной грани из оставшихся кластеров. В результате получим BSP-дерево, показанное на рис. 121.

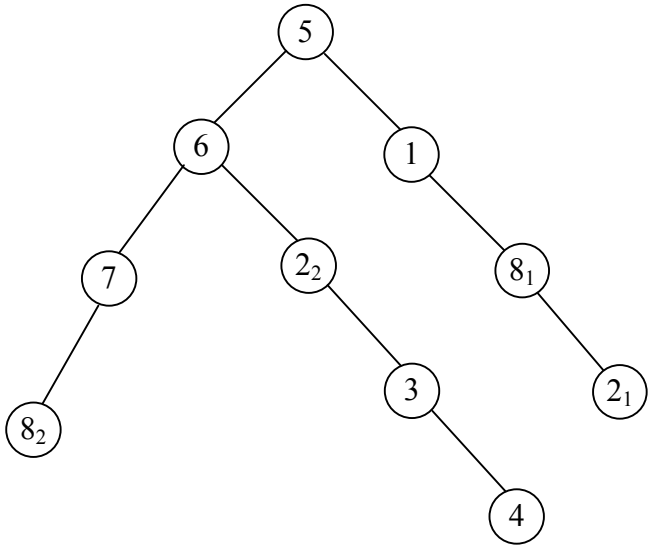


Рис. 121. BSP-дерево для примера рис. 120

4.9.4. Метод построчного сканирования

Метод построчного сканирования является примером метода, удачно использующего растровые свойства картинной плоскости для упрощения исходной задачи и сведения ее к серии простых задач в пространстве меньшей размерности.

Все изображение на картинной плоскости (экране) можно представить как состоящее из горизонтальных (или вертикальных) линий пикселей – строк (или столбцов). Каждой такой строке пикселей соответствует сечение сцены плоскостью, проходящей через соответствующую строку и наблюдателя (для параллельного проектирования – проходящей через строку и параллельную направлению проектирования), при наших допущениях.

Пересечением секущей плоскости со сценой будет множество непересекающихся (за исключением концов) отрезков, высекаемых на гранях секущей плоскостью (рис. 122).

В результате мы приходим к задаче удаления невидимых частей для отрезков на секущей плоскости при проектировании на прямую, являющуюся результатом пересечения с ней картинной плоскости. Тем самым получается задача с размерностью на единицу меньше, чем исходная задача, – вместо определения того, какие части граней закрывают друг друга при проектировании на плоскость, необходимо определить, какие части отрезков закрывают друг друга при проектировании на прямую.

Существуют различные методы решения задачи удаления невидимых частей отрезков. Одним из наиболее простых является использование одномерного *z*-буфера, совмещающего крайнюю простоту с весьма небольшими затратами памяти даже при высоком разрешении картинной плоскости. К тому же существуют аппаратные реализации этого подхода. С другой стороны, для определения видимых частей можно воспользоваться и аналитическими (непрерывными) методами.

Заметим, что изменение видимости отрезков может происходить лишь в их концах. Поэтому достаточно проанализировать взаимное расположение концов отрезков с учетом глубины.

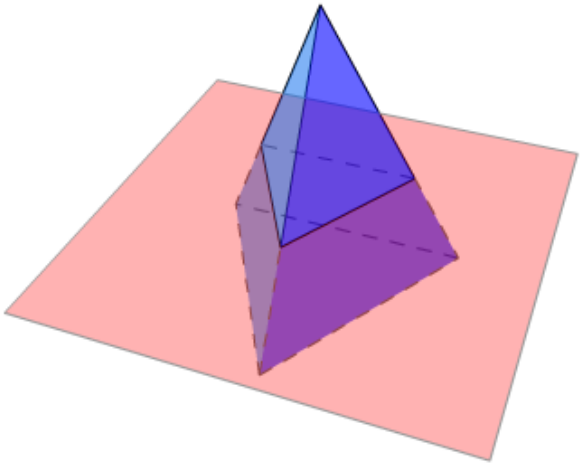


Рис. 122. Пример метода построчного сканирования

4.9.5. Алгоритм Варнака

Алгоритм Варнака является еще одним примером алгоритма, основанного на разбиении картинной плоскости на части, для каждой из которых исходная задача может быть решена достаточно просто. Разобьем видимую часть картинной плоскости на четыре равные части и рассмотрим,

каким образом могут соотноситься между собой проекции граней и получившиеся части картинной плоскости.

Возможны 4 различных случая:

1. Проекция грани полностью накрывает область (рис. 123 а);
2. Проекция грани пересекает область, но не содержится в ней полностью (рис. 123 б);
3. Проекция грани целиком содержится внутри области (рис. 123 в);
4. Проекция грани не имеет общих внутренних точек с рассматриваемой областью (рис. 123 г).

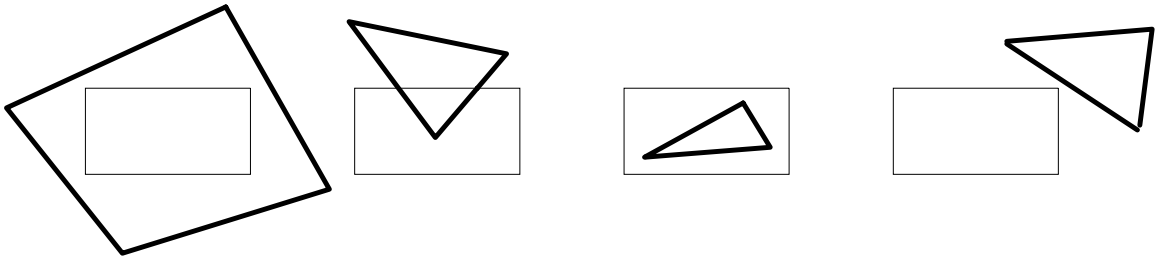


Рис. 123. Случаи пересечений проекций граней и картинной плоскости

Очевидно, что в последнем случае грань вообще никак не влияет на то, что видно в данной области.

Сравнивая область с проекциями всех граней, можно выделить случаи, когда изображение, получающееся в рассматриваемой области, определяется сразу:

• проекция ни одной грани не попадает в область;

• проекция только одной грани содержится в области или пересекает область; в этом случае грань разбивает всю область на две части, одна из которых соответствует этой грани;

• существует грань, проекция которой полностью накрывает данную область, и эта грань расположена к картинной плоскости ближе, чем все остальные грани, проекции которых пересекают данную область; в этом случае вся область соответствует этой грани.

Если ни один из рассмотренных трех случаев не имеет места, то снова разбиваем область на 4 равные части и проверяем выполнение этих условий для каждой из частей. Те части, для которых видимость таким образом определить не удалось, разбиваем снова и т. д. (рис. 124).

Естественно возникает вопрос о критерии, на основании которого прекращать разбиение (иначе оно может продолжаться до бесконечности).

В качестве очевидного критерия можно взять размер области: как только размер области станет не больше размера 1 пикселя, то производить дальнейшее разбиение не имеет смысла и для данной области ближайшая к ней грань определяется явно.

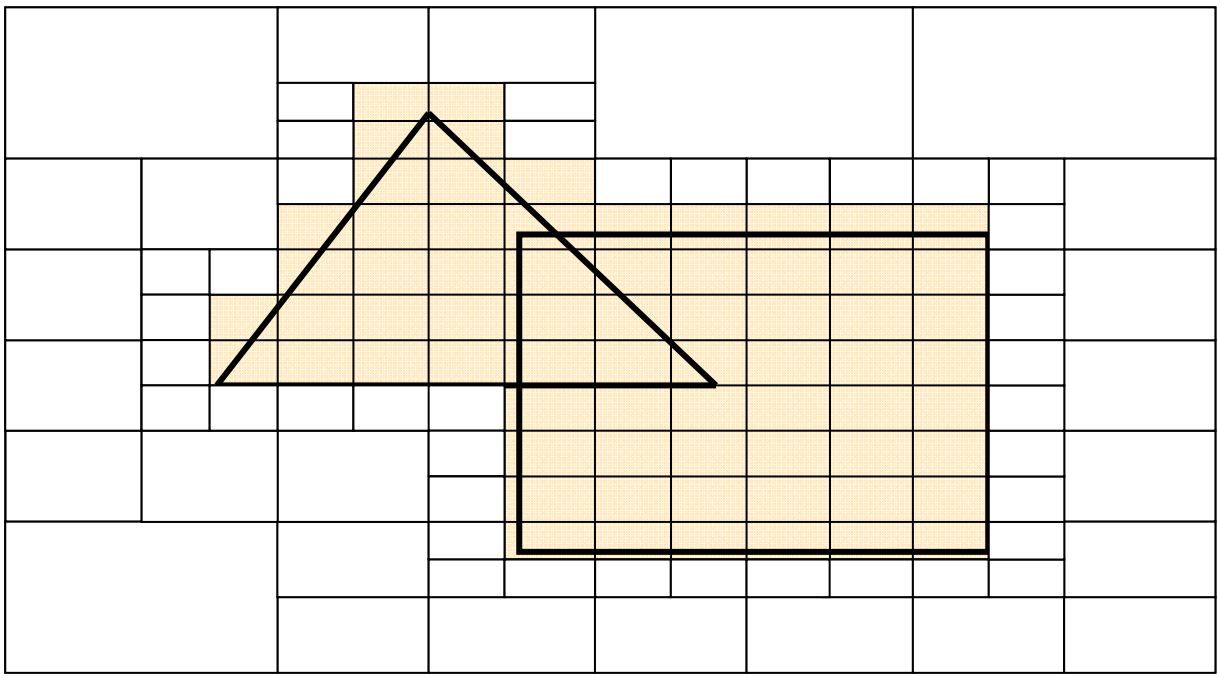


Рис. 124. Пример определения видимости грани методом Варнака

4.10. Геометрические сплайны

Историю сплайнов принято отсчитывать с 1946 года. Функции, подобные тем, что сейчас называют сплайнами, были известны математикам давно, начиная как минимум с Леонарда Эйлера, но их интенсивное изучение началось, фактически, только в середине XX века. В 1946 году Исаак Шёнберг впервые употребил этот термин в качестве обозначения класса полиномиальных сплайнов. Сначала сплайны рассматривались как удобный инструмент в теории и практике приближения функций. Однако довольно скоро область их применения начала быстро расширяться и обнаружилось, что существует очень много сплайнов самых разных типов. Сплайны стали активно использоваться в численных методах, в системах автоматического проектирования и автоматизации научных исследований, во многих других областях человеческой деятельности и, конечно, в компьютерной графике.

Сам термин «сплайн» происходит от английского *spline*. Именно так называется гибкая стальная полоска, при помощи которой чертежники проводили через заданные точки плавные кривые. В былые времена подобный способ построения плавных обводов различных тел, таких, как, например, корпус корабля, кузов автомобиля, а потом фюзеляж или крыло самолета, был довольно широко распространен в практике машиностроения. В результате форма тела задавалась при помощи набора очень точно изготовленных сечений – плазов. Появление компьютеров позволило перейти от этого, плазово-шаблонного, метода к более эффективному способу задания поверхности обтекаемого тела. В основе этого подхода к описанию поверхностей лежит использование сравнительно несложных формул, позволяющих

восстанавливать облик изделия с необходимой точностью. Ясно, что для большинства тел, встречающихся на практике, вряд ли возможно отыскание простых универсальных формул, которые описывали бы соответствующую поверхность глобально, то есть, как принято говорить, в целом. Это означает, что при решении задачи построения достаточно произвольной поверхности обойтись небольшим количеством формул, как правило, не удается. Вместе с тем аналитическое описание (описание посредством формул) внешних обводов изделия, то есть задание в трехмерном пространстве двумерной поверхности, должно быть достаточно экономным. Это особенно важно, когда речь идет об обработке изделий на станках с числовым программным управлением. Обычно поступают следующим образом: задают координаты сравнительно небольшого числа опорных точек, лежащих на искомой поверхности, и через эти точки проводят плавные поверхности. Именно так поступает конструктор при проектировании кузова автомобиля (ясно, что на этой стадии процесс проектирования сложного объекта содержит явную неформальную составляющую). На следующем шаге конструктор должен получить аналитическое представление для придуманных кривых или поверхностей. Вот для таких задач и используются сплайны.

Средства компьютерной графики, особенно визуализация, существенно помогают при проектировании, показывая конструктору, что может получиться в результате, и, давая ему многовариантную возможность сравнить это с тем, что сложилось у него в голове.

Достаточно типичной является следующая задача: по заданному массиву точек на плоскости ($2D$) или в пространстве ($3D$) построить кривую либо проходящую через все эти точки (задача интерполяции), либо проходящую вблизи от этих точек (задача сглаживания).

Совершенно естественно возникают вопросы:

1) в каком классе кривых искать решение поставленной задачи?

2) как искать?

4.10.1. Сплайн-функции. Случай одной переменной

Вначале определим допустимый класс кривых. Он должен быть таким, чтобы:

а) решение было единственным;

б) построенная кривая изменялась плавно.

Пусть на плоскости задан набор точек

$$(x_i, y_i), \quad i = 0, \ldots, m,$$

таких, что они упорядочены следующим образом:

$$x_0 < x_1 < \ldots < x_m,$$

т. е. точки пронумерованы в порядке возрастания вдоль оси Ox, это позволяет искать кривую в классе графиков функций.

Прежде всего, можно воспользоваться представлением графика функции в виде многочлена. Например, существует интерполяционный многочлен Лагранжа:

$$L_m(x) = \sum_{i=0}^{m} y_i \cdot \frac{\omega_m(x)}{(x - x_i) \cdot \omega_m(x_i)},$$

где

$$\omega_m(x) = \prod_{j=0}^{m} (x - x_j).$$

График этого полинома (многочлена) проходит через все заданные точки (рис. 125). Многочлен однозначно определяется набором своих коэффициентов, число которых совпадает с количеством точек в заданном наборе ($m+1$).

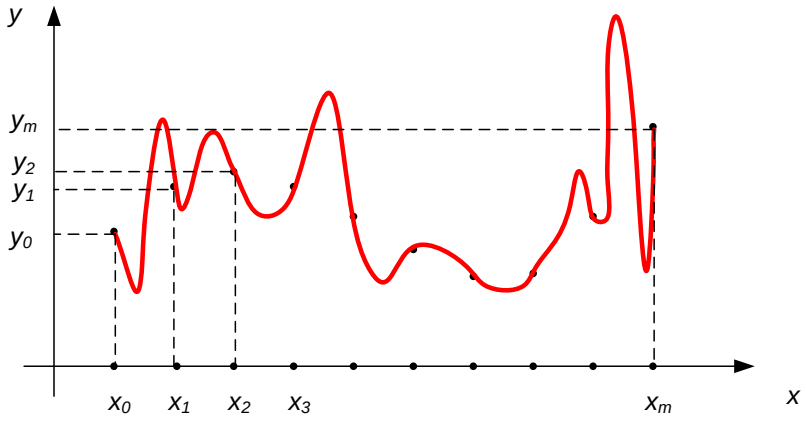


Рис. 125. Полином Лагранжа порядка 5.

Недостатки:

1. Степень многочлена Лагранжа равна m, если заданных точек ($m+1$), поэтому чем больше точек, тем больше степень полинома Лагранжа, и хотя он гарантированно пройдет через все заданные точки, но если его порядок велик, то отклонение от ожидаемых значений в промежутках между ними может быть существенным.

2. Изменение всего лишь одной точки требует полного пересчета коэффициентов интерполяционного многочлена, при этом вид кривой может существенно измениться.

С другой стороны, самый простой способ построения искомой кривой – соединение точек набора прямолинейными отрезками. При этом получается ломаная. При такой кусочно-линейной интерполяции нужно найти всего $2m$ чисел, но при этом мы не получаем необходимой гладкости кривой (с математической точки зрения в точках сопряжения отрезков прямой первые производные терпят разрыв).

Нужно найти класс функций, которое сохранили бы достоинство простоты (сравнительно небольшое количество вычислений) при отсутствии вышеуказанных недостатков. Для этого используют многочлены, но строят их последовательно – звено за звеном. В результате получается *полиномиальный многозвенник*. При этом важно выбрать степень используемых многочленов и

тщательно подобрать коэффициенты многочленов из условий гладкого сопряжения соседних звеньев то, что получается в результате такого подхода и называется *сплайн–функцией* или просто *сплайном*:

$$y = S(x).$$

Сплайн–функция S(x) обладает следующими свойствами:

С довольно большой точностью часть графика функции между двумя любыми соседними опорами (точками) можно приблизить к многочленам 3-ей степени и на всём промежутке от x_0 до x_m эта функция дважды непрерывно дифференцируема. Такая функция относится к интерполяционным кубическим сплайнам.

Интерполяционный кубический сплайн – это функция *S(x)*, обладающая следующими свойствами:

1) График функции проходит через каждую точку заданного массива

$$S(x_i) = y_{i,} \quad i = 0, \ldots, m.$$

2) На каждом из отрезков

$$\left|x_i, x_{i+1}\right|, \quad i = 0, \ldots, m-1$$

функция является многочленом 3-ей степени

$$S(x) = \sum_{j=0}^{3} (a_j)^i \cdot (x - x_i)^j.$$

3) На всём отрезке от x_0 до x_m функция *S(x)* имеет непрерывную производную.

На каждом отрезке сплайн *S(x)* определяется четырьмя коэффициентами, поэтому для полного построения сплайна на всём большом отрезке надо найти *4m* чисел.

Третье условие будет выполнено, если потребовать непрерывность сплайна во всех внутренних узлах x_i (i *=1,2,...m-1*) это даёт *m-1* условие на коэффициент. Ещё *m-1* условие даёт требование непрерывности первой производной во внутренних узлах и ещё *m-1* – второй.

Всего получается *4m–2* условий на коэффициенты. Недостающие 2 условия для полного определения коэффициентов получают, например, задавая значение 1-ых производных на концах отрезков:

$$S'(x_0) = l_0,$$
$$S'(x_m) = l_m.$$

Эти условия называются граничными.

4.10.2. Сплайновые кривые

Будем использовать параметрические уравнения кривой. Параметрически заданной кривой называется множество γ точек M (x, y, z), координаты x, y, z которых определяются соотношениями:

$$x = x(t), \quad y = y(t), \quad z = z(t),$$

которые называются параметрическими уравнениями кривой γ, где *x(t), y(t), z(t)* функции, непрерывные на заданном интервале $a \le t \le b$.

Если ввести новую переменную

$$u = \frac{t-a}{b-a},$$

то можно представить, что $a=0,\ b=1$, т. е. осуществить нормализацию параметров функции.

Часто используют векторную форму записи параметрических уравнений:

$$\boldsymbol{r}(t) = \begin{bmatrix} x(t) \\ y(t) \\ z(t) \end{bmatrix}, \quad 0 \le t \le 1.$$

Параметр t задает ориентацию параметризованной кривой $\boldsymbol{r(t)}$ (порядок прохождения точек при монотонном возрастании параметра).

Дальше можно задавать первую производную кривой и если $\dot{\boldsymbol{r}}(t) = 0$ в каждой точке, то кривая называется регулярной, т. е. в каждой точке кривой существует касательная.

4.10.3. Сглаживающие кривые. Кривая Безье

Кривой Безье, определяемой массивом V (рис. 126), называется кривая, определяемая векторным уравнением:

$$\boldsymbol{r}(t) = \sum_{i=0}^{m} C_m^i \cdot t^i \cdot (1-t)^{m-i} \cdot V_i, \quad t \in [0,1],$$

где

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

биномиальный коэффициент. В комбинаторике C^i_m дает число сочетаний из m элементов по i.

Кривая Безье обладает замечательными свойствами:

1) она является гладкой кривой;

2) начинается в точке V_0 и заканчивается в точке V_m, касаясь при этом отрезков $V_0\,V_1$, и $V_{m-1}V_m$ контрольной ломаной;

3) функциональные коэффициенты $C^i_m t^i (1-t)^{m-i}$ при вершинах V_i, $i=0,I,....m$, представляет собой универсальные многочлены (многочлены Бернштейна); они неотрицательны, и их сумма равна единице:

$$\sum_{i=0}^{m} C_m^i \cdot t^i \cdot (1-t)^{m-i} = \left[t + (1-t)\right]^m = 1$$

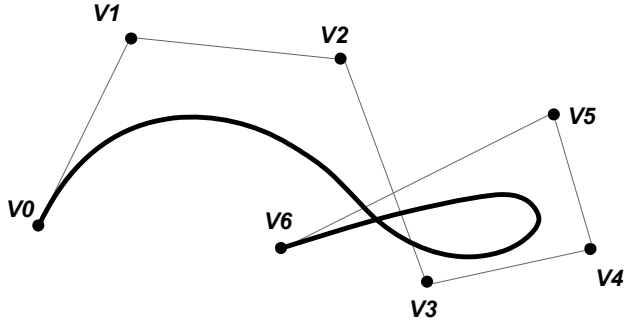


Рис. 126. Кривая Безье, построенная по заданной последовательности массива точек V

Поэтому кривая Безье целиком лежит в выпуклой оболочке, порождаемой массивом точек.

При *m=3* получаем элементарную кубическую кривую Безье, определяемую четверкой точек V_0, V_1, V_2, V_3 и описываемую векторным параметрическим уравнением:

$$r(t) = \left\{\left[(1-t)\cdot V_0 + 3t\cdot V_1\right]\cdot(1-t) + 3t^2\cdot V_2\right\}\cdot(1-t) + t^3\cdot V_3, \quad t\in[0,1].$$

Порядок точек в заданном наборе влияет на вид кривой Безье (рис. 127).

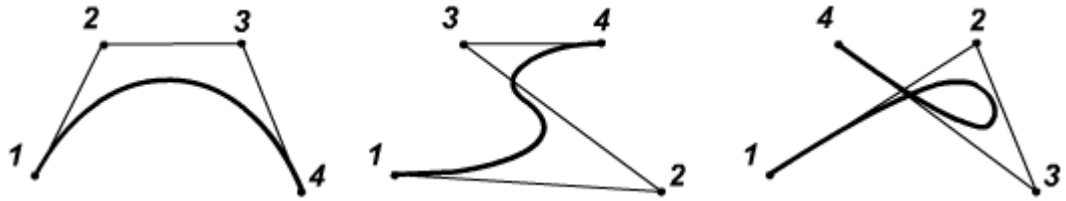


Рис. 127. Примеры построения кривых Безье в зависимости от порядка задания вершин массива

Действительно кривая Безье является сглаживающей, а не интерполяционной, так как проходит не через все точки заданного массива, а только через начальную и конечную точки.

4.10.4. Сплайновые поверхности

Сплайновая поверхность является расширением понятия сплайновой кривой. Сплайн-функция *S(x)* становится двухмерной *S(x,y)*, сохраняя при этом свои основные свойства: непрерывность и гладкость. Так распространение интерполяционых кубических сплайнов на двухмерный случай позволяет получать гладкие поверхности практически любой формы. Сплайновые поверхности значительно улучшают фотореалистичность объектов, создаваемых средствами компьютерной графики, в частности по сравнению с полигональными моделями. При этом математическое описание усложняется не намного.

Глава 5. Основы художественного конструирования технических изделий и графических интерфейсов

5.1. Развитие технической эстетики и художественного конструирования в России и за рубежом

В 1918 г. – были созданы: ВХУТЕМАС (Всероссийские художественно-технические мастерские); ВХУТЕИН (Всероссийский художественно-технический институт); ICSID – международная организация.

В 1925 г. – начало развиваться как направление художественное конструирование.

Направление *стайлинг* (от англ. *stile* – стиль) подразумевает внешнюю эстетизацию изделий без затрагивания основы.

В 1944 г. – был создан британский совет по художественной эстетике.

В 1966 г. – был создан британский Центр промышленного дизайна (Industrial Design Center).

5.2. Цели, которые преследует развитие дизайна

1. Повышение качества изделий.
2. Повышение экономической эффективности.
3. Оптимизация взаимоотношений средств производства.

Промышленный дизайн – тотальное проектирование промышленных изделий и предметной среды.

Необходимо осознание специфики промышленного проектирования как особой сферы человеческой деятельности, целью которой является создание всего предметного мира, окружающего человека, по законам красоты и функциональности.

Художественный образ – это отражение предмета окружающего мира.

Техническая эстетика (греч. — мастерство искусства) — это научная дисциплина, изучающая закономерности формирования гармоничной предметной среды жизни и деятельности человека методами и средствами дизайна. Можно сказать, что *техническая эстетика* – теория промышленного искусства (дизайна).

Предмет технической эстетики – область деятельности конструктора-художника.

Методологической основой технической эстетики служит *эстетика* вообще – наука об общих законах освоения мира человеком и их проявлении во всех сферах человеческой деятельности.

Техническая эстетика подразделяется на общетеоретическую и прикладную части. Техническая эстетика изучает:

2) общественную природу и закономерности возникновения дизайна;
3) основные принципы и закономерности дизайна;

4) проблемы стиля и мастерства.

5.3. Основные принципы технической эстетики

1. В своих требованиях к промышленным изделиям ТЭ исходит из того, что предметный мир, окружающий человека в быту и на производстве, должен составлять единое целое. Его основное назначение – возможно лучше служить человеку.
2. Количество предметов, окружающих человека, должно быть минимальным, а польза от них – максимальной.
3. Собранные вместе, такие предметы должны дополнять друг друга функционально, создавая разнообразные рациональные комплексы.

Производственная эстетика – раздел общей эстетики, изучающий закономерности формирования производственной среды и эстетических отношений в труде.

Художественное конструирование – творческий метод промышленного искусства, т.е. процесс создания изделий.

Художественное конструирование должно обеспечить:

1) удобство пользования;
2) максимальное соответствие условиям эксплуатации;
3) создание высоких эстетических качеств.

5.4. Эргономика и ее проблемы

Эргоно́мика (от др.-греч. *ἔργον* — работа и *νόμος* — закон) — научная дисциплина, комплексно изучающая производственную деятельность человека и ставящая целью её оптимизацию. Еще одно похожее определение:

Эргономика – это наука, изучающая функциональные возможности и ограничения человека в трудовых процессах с целью создания для него оптимальных условий труда.

Эргономика возникла в 1920-х годах в связи со значительным усложнением техники, которой должен был управлять человек в своей деятельности. Первые исследования по эргономике начали проводиться в Великобритании, США и Японии. Термин эргономика был принят в Великобритании в 1949 году, когда группа английских учёных положила начало организации Эргономического исследовательского общества. В СССР в 20-е годы предлагалось название *эргология*, в США раньше имелось собственное наименование — исследование человеческих факторов, а в ФРГ — *антропотехника*, но в настоящее время наибольшее распространение получил английский термин. В последнее время эргономика отходит от классического определения и перестаёт быть строго связана с производственной деятельностью. Определение, принятое Международной Эргономической Ассоциацией (IEA) в 2007 году: «Эргономика — это область приложения научных знаний о человеке к проектированию предметов, систем и окружений, используемых им».

Эргономика изучает действия человека в процессе работы, скорость освоения им новой техники, затраты его энергии, производительность и интенсивность при конкретных видах деятельности. Современная эргономика подразделяется на *микроэргономику*, *мидиэргономику* и *макроэргономику*:

Ми́кроэргономика (иногда её неверно упоминают как миниэргономику) занимается исследованием и проектированием систем «человек-машина». В частности, проектирование интерфейсов программных продуктов находится в ведении микроэргономики.

Ми́диэргономика занимается изучением и проектированием систем «человек–коллектив», «коллектив–организация», «коллектив–машина», «человек–сеть». Именно мидиэргономика исследует производственные взаимодействия на уровне рабочих мест и производственных задач. К ведению мидиэргономики, в частности, относится проектирование структуры организации и помещений; планирование и установление расписания работ; гигиена и безопасность труда.

Ма́кроэргономика исследует и проектирует более общие системы, такие как «человек–общество», «организация–система организаций».

При изучении и создании эффективных управляемых человеком систем, в эргономике используется системный подход. *Эргономика* – комплексная наука, использующая данные многих различных наук. Для оптимизации управляемых человеком систем эргономика использует научные изыскания психологии, физиологии (особенно нейрофизиологии), гигиены и безопасности труда, социологии, культурологии и многих технических, инженерных и информационных дисциплин. Можно сказать, что эргономика занимается изучением человеческого фактора, тогда как, например, технические науки – изучением технических факторов. Некоторые термины эргономики стали широко употребляться в быту, например, *человекоча́с* (мера временной ёмкости деятельности).

В настоящее время открытия эргономики используются не только на производстве, но также в быту, в спорте и даже в искусстве.

Проблемы эргономики

1. Анализ задач человека в системах управления и способов связи с другими компонентами системы.
2. Исследование групповой деятельности операторов, обслуживающих систему управления.
3. Анализ структуры деятельности оператора: определяются те требования, которым должен удовлетворять оператор с точки зрения подготовки, инженерной психологии.
4. Исследование факторов, влияющих на эффективность работы операторов.
5. Изучение процессов приема человеком информации на разных носителях.

6. Анализ процессов переработки и хранения информации, а также формирования решений человеком–оператором.
7. Исследование управляющих воздействий человека-оператора на органы управления системой.

Первые 4 проблемы из вышеперечисленных относятся к системе человек-машина. Последние 3 проблемы относятся собственно к человеку-оператору.

5.5. Принципы и закономерности художественного конструирования

1. Соответствие формы содержанию проектируемого объекта, изделия, системы.
2. Учет взаимодействия с окружающей средой.
3. Комплексное решение при проектировании любых промышленных изделий следующих вопросов:

– функциональных;
– конструктивно-технологических;
– экономических;
– эргономических;
– эстетических.

5.6. Композиция как средство выражения художественных качеств форм

Композиция – изучает закономерности построения формы, а также формирование структуры изделия в зависимости от его назначения.

5.6.1. Виды композиции

В зависимости от формы изделия различают следующие виды композиции:
– объемные,
– линейные,
– плоские.

В зависимости от формы взаимосвязей объема и пространства, как элементов формы, различают следующие виды композиции:
– с функциональным внутренним пространством,
– с цельным объемом.

Формы можно разделить по взаимоположению с другими формами пространства, а также по отношению к человеку. Формы:
– фронтальная форма (например, клавиатура),
– объемная (например, в архитектуре),
– объемно-пространственная форма (дома вдоль улицы)

Каждая из этих форм может подразделяться на симметричную и асимметричную, статическую и динамическую.

Статической обычно называют форму, имеющую центр симметрии (шар, тор, куб). Для статических объектов главная плоскость совпадает с направлением силы тяжести. Для динамических объектов ось композиции совпадает с направлением движения.

5.6.2. Единство композиции

Единство композиции состоит в согласовании всех частей или деталей объектов их направленности, и подчиненности главной в функциональном и композиционном отношении части.

Главное должно быть крупнее второстепенного, а второстепенное должно быть определенным образом ориентировано в пространстве, чтобы указывать, где находится главное.

5.7. Средства гармонизации формы промышленных объектов

1. Пропорция
2. Масштаб
3. Ритм
4. Контраст
5. Нюанс

5.7.1. Пропорция

Часто при конструировании используются иррациональные соотношения в геометрической пропорции, например, так называемое «*золотое сечение*», которое использует пропорцию:

$\frac{\sqrt{5}-1}{2} = 0{,}618$, т.е. *0,618 : 0,382* , или более грубо: *0,62 : 0,38* – см. рис. 126.

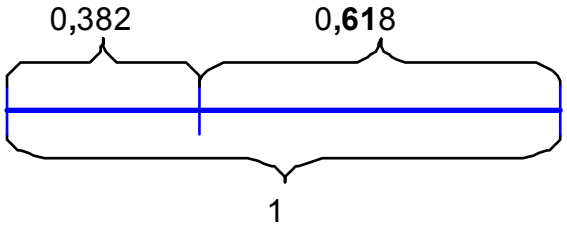


Рис. 128. Пропорция «золотого сечения».

Это соотношение было получено в античные времена с помощью циркуля, как показано на рис. 129.

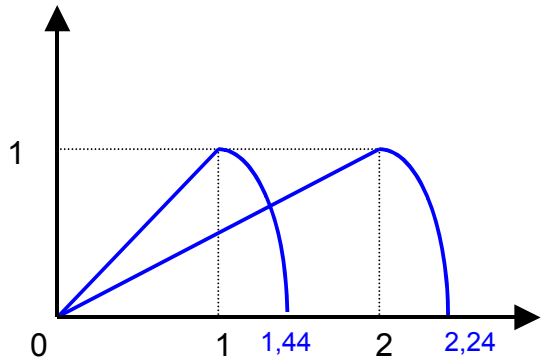


Рис. 129. Получение пропорции «золотого сечения» с помощью циркуля

5.7.2. Масштаб

Масштаб - показывает как размер (любой: линейный, площадь или объем, например) объекта относится к эталону (рис. 130).

$$M = \frac{L}{L_э}.$$

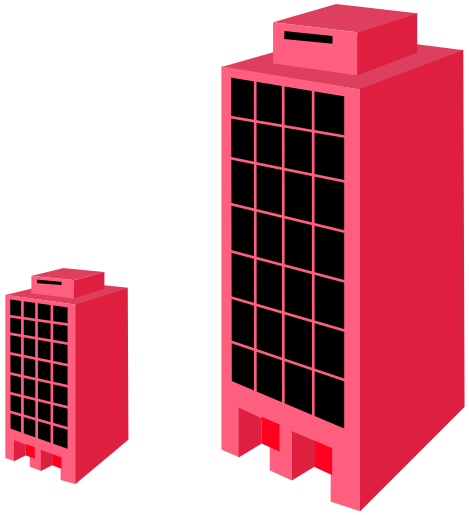


Рис. 130. Пример масштабного 3-мерного изображения

5.7.3. Ритм

Ритм – показывает тенденцию изменения частоты повторения некоторого элемента формы.
Например, частота линий в конструкции или рисунке (рис. 131):

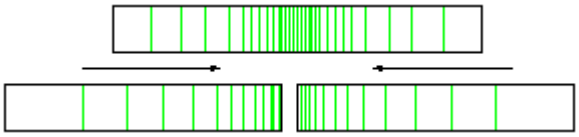


Рис. 131. Пример использования ритма

5.7.4. Контраст

Контраст – использует противоположности, крайности; например: черный – белый; большой – маленький (рис. 132); громкий – тихий; быстрый – медленный и т.д.

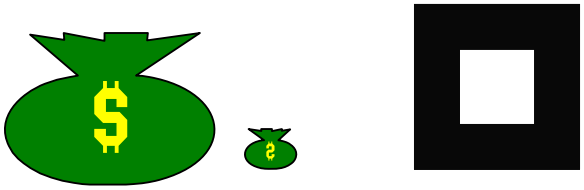


Рис. 132. Примеры, иллюстрирующие контраст

5.7.5. Нюанс

Нюанс – противоположно контрасту; что-то близкое по свойствам к рассматриваемому объекту (рис. 133).

Рис. 133. Примеры, иллюстрирующие понятие нюанса

5.8. Практические рекомендации по проектированию графических интерфейсов

5.8.1. Технология «живого» интерфейса

Объектное интерактивное программирование обеспечивает создание эффективных интерфейсов для специальных задач.

Появилась технология так называемого «живого» интерфейса, которая превращает взаимодействие пользователя с информационной системой в процесс, напоминающий общение людей.

Помимо клавиатуры для взаимодействия с операционной платформой стали использоваться мышь, световое перо, микрофон, динамик, система шлем-дисплей (система управления информационным процессом, состоящая как минимум из шлема и пары экранов, выполненных в виде очков).

5.8.2. Основные принципы построения интерфейсов

1. Золотое сечение
2. Кошелек Миллера
3. Принцип группировки
4. Бритва Оккама, или KISS
5. Видимость отражает полезность
6. Умное заимствование

Золотое сечение

Золотое сечение – это самая комфортная для глаза пропорция, способствующая наилучшему зрительному восприятию и появлению ощущения красоты и гармонии.

Золотое сечение – это такое пропорциональное деление отрезка на неравные части, при котором весь отрезок так относится к большей части, как сама большая часть относится к меньшей.

Отрезки золотой пропорции выражаются бесконечной иррациональной дробью b=0,618..., если весь принять за единицу, a = 0,382. Отношение же отрезков a и b составляет 1,618.

Прямоугольник с таким отношением сторон стали называть золотым прямоугольником. Он также обладает интересными свойствами. Если от него отрезать квадрат, то останется вновь золотой прямоугольник. Этот процесс можно продолжать до бесконечности. А если провести диагональ первого и второго прямоугольника, то точка их пересечения будет принадлежать всем получаемым золотым прямоугольникам.

Есть и золотой треугольник (равнобедренный треугольник, у которого отношение длины боковой стороны к длине основания равняется 1,618), и золотой кубоид (прямоугольный параллелепипед с ребрами, имеющими длины 1,618, 1 и 0,618).

С развитием дизайна и технической эстетики действие закона золотого сечения распространилось на конструирование машин, мебели и т. д. Проектирование компьютерных интерфейсов – не исключение. Формы диалоговых окон и элементов управления, стороны которых относятся как 1,618, очень привлекательны для пользователей.

Например, очень много положительных эмоций у пользователей программы *Chameleon Clock* (*http://www.softshape.com*) вызывает такая, казалось бы, обыденная вещь, как вид диалогового окна *Свойства*. А все потому, что при его проектировании использовался именно принцип золотого сечения.

Кошелек Миллера

Этот принцип назван так в честь ученого-психолога Г. А. Миллера, который исследовал кратковременную память, проверяя выводы, сделанные ранее его коллегой, Г. Эббингаузом, который пытался выяснить, сколько информации может запомнить человек без каких-либо специальных мнемонических приемов. Оказалось, что емкость памяти ограничена семью цифрами, семью буквами или названиями семи предметов. Это «магическое число» семь, служащее своего рода меркой памяти, и было проверено Миллером, который показал, что память действительно в среднем не может хранить более семи элементов; в зависимости от сложности элементов это число может колебаться в пределах от пяти до девяти.

Если необходимо в течение короткого времени сохранить информацию, включающую больше семи элементов, мозг почти бессознательно группирует эту информацию таким образом, чтобы число запоминаемых элементов не превышало предельно допустимого. Например, номер банковского счета 30 637 402 710, состоящий из одиннадцати элементов, будет, скорее всего, запоминаться как 30 63 740 27 10, т. е. как пять числовых элементов, или восемь слов (тридцать, шестьдесят, три, семьсот, сорок, двадцать, семь, десять).

Применяя принцип кошелька Миллера в дизайне интерфейсов, следует группировать элементы в программе (кнопки на панелях инструментов, пункты меню, закладки, опции на этих закладках и т. п.) с учетом этого правила— т. е. не более семи в группе, в крайнем случае — девяти. Например, четырнадцать кнопок на верхней панели, между которыми нет ни одного разделителя, воспринимаются гораздо хуже, чем кнопки, которые разделены на группы.

Итак, принцип кошелька Миллера говорит о семи плюс-минус двух элементах. Но если взглянуть на программы, интерфейс которых совершенствовался годами (тот же *Microsoft Word*), то можно заметить, что число объектов (пунктов меню, кнопок на панелях инструментов) в группах доходит до шести-семи довольно редко, а в основном элементы сгруппированы по три-четыре объекта. Такие небольшие группы объектов наиболее хорошо воспринимаются взглядом пользователя, уже слегка утомленного сложными интерфейсами современных программ.

Очевидно, при проектировании интерфейсов программ верхнюю границу кошелька Миллера – семь-девять элементов – нужно применять очень осторожно, стараясь обходиться группами, содержащими, максимум, пять объектов.

Принцип группировки

Согласно этому правилу, экран программы должен быть разбит на ясно очерченные блоки элементов, может быть, даже с заголовком для каждого блока. При этом группировка, естественно, должна быть осмысленной: как расположение элементов в группах, так и расположение самих групп друг от друга должны быть продуманы.

Примеров реализации этого принципа очень много: это уже упоминавшиеся при разговоре о кошельке Миллера пункты меню, кнопочные панели инструментов, а также сгруппированные по назначению флажки и переключатели, с помощью которых настраиваются параметры работы программы в диалоговых окнах *Свойства*, *Настройка* и т. п.

Бритва Оккама, или KISS

Философский принцип, носящий название «Бритва Оккама», гласит: «Не множить сущности без надобности». Или, как говорят американцы, KISS ("*Keep It Simple, Stupid*" — «Не усложняй, болван»).

На языке интерфейсов это означает, что:

1) любая задача должна решаться минимальным числом действий;
2) логика этих действий должна быть очевидной для пользователя;
3) движения курсора и даже глаз пользователя должны быть оптимизированы.

Простым на первый взгляд требованиям из этого списка на самом деле не так уж легко следовать. Для проектирования сложного по своим функциям и простого для понимания интерфейса требуется немалые опыт, знания и особое чутье.

Принцип KISS перекликается с несколькими из эвристических правил Якоба Нильсена[4] — «Эстетичный и минималистический дизайн», «Равенство между системой и реальным миром», «Понимание лучше, чем запоминание». KISS более универсален и применяется практически во всех сферах человеческой деятельности, в том числе и в программировании.

Видимость отражает полезность

Смысл этого принципа состоит в том, чтобы вынести самую важную информацию и элементы управления на первый план и сделать их

[4] Якоб Нильсен (Jakob Nielsen) – основатель и руководитель компании "Nielsen Norman Group", которую он создал вместе с Дональдом Норманом, бывшим вице-президентом "Apple Computer". До 1998 года Якоб Нильсен работал ведущим инженером в "Sun Microsystems". Нильсен имеет более 30 патентов США, причем большая часть его изобретений направлена на облегчение работы пользователей Интернета. Он – автор девяти книг по пользовательским интерфейсам, web-дизайну, структуре сайтов и прочим аспектам web-usability.

легкодоступными пользователю, а менее важную – переместить, например, в меню.

Этот вопрос уже затрагивался при разговоре о принципе Якоба Нильсена «Эстетичный и минималистический дизайн», правда, в привязке к отражению полезности.

Отличие принципа «Видимость отражает полезность» как раз и состоит в том, что интерфейс программы должен быть построен вокруг объектов, с которыми манипулирует пользователь, и отражать состояние текущего объекта. Реализацию этого принципа мы видим каждый раз, когда пользуемся компьютером: контекстные панели инструментов в программах пакета Microsoft Office, которые меняются в зависимости от того, с какой частью программы (редактором, предварительным просмотром, рисованием и т. п.) и в данный момент работает пользователь.

Умное заимствование

Заимствование широко распространенных приемов дизайна интерфейсов и удачных находок авторов конкурирующих программ позволяет резко сократить время обучения и повысить комфорт пользователя. При работе он будет использовать уже приобретенные навыки – этот вопрос затрагивает и принцип равенства между системой и реальным миром.

Заимствование чужих интерфейсных находок не является чем-то зазорным. Программы, лидирующие па рынке, являются неистощимым источником вдохновения для разработчиков более мелких программ, поразительно напоминающих легендарный Norton Commander: FAR, Volcov Commander, DOS Navigator, DISCo Commander.

Следует комплексно применять основные положения каждого из приведённых принципов, поскольку каждый из них в отдельности наиболее правильно и просто помогает решить какой-то узкий круг одних задач, но может оказаться не пригодным для других. Так, например, сложно сгруппировать кнопки на панели управления по принципу золотого сечения, нет смысла отказываться от заимствования проверенных методов представления информации.

5.9. Примеры проектирования графических интерфейсов оператора

Современные программные системы проектирования и управления, такие как SCADA (*Supervisory Control and Data Acquisition*) – система диспетчерского управления и сбора данных, – помимо прочих функций позволяют быстро и эффективно создавать удобные графические интерфейсы диспетчеров.

SCADA-системы в целом используются на нижних уровнях автоматизированных систем управления технологическими процессами (АСУТП), т.е. для непосредственного взаимодействия с технологическим оборудованием, и выполняют следующие функции:

1) сбор первичной информации от датчиков;
2) хранение, обработка и визуализация данных;
3) управление и регистрация аварийных сигналов;
4) связь с корпоративной информационной сетью;
5) автоматизированная разработка прикладного программного обеспечения.

5.9.1. SCADA-система Trace Mode

На рис. 134 приведен пример интерфейса оператора, контролирующего процесс подготовки питьевой воды. Этот графический интерфейс диспетчера спроектирован с помощью отечественной SCADA-системы Trace Mode 5.0.

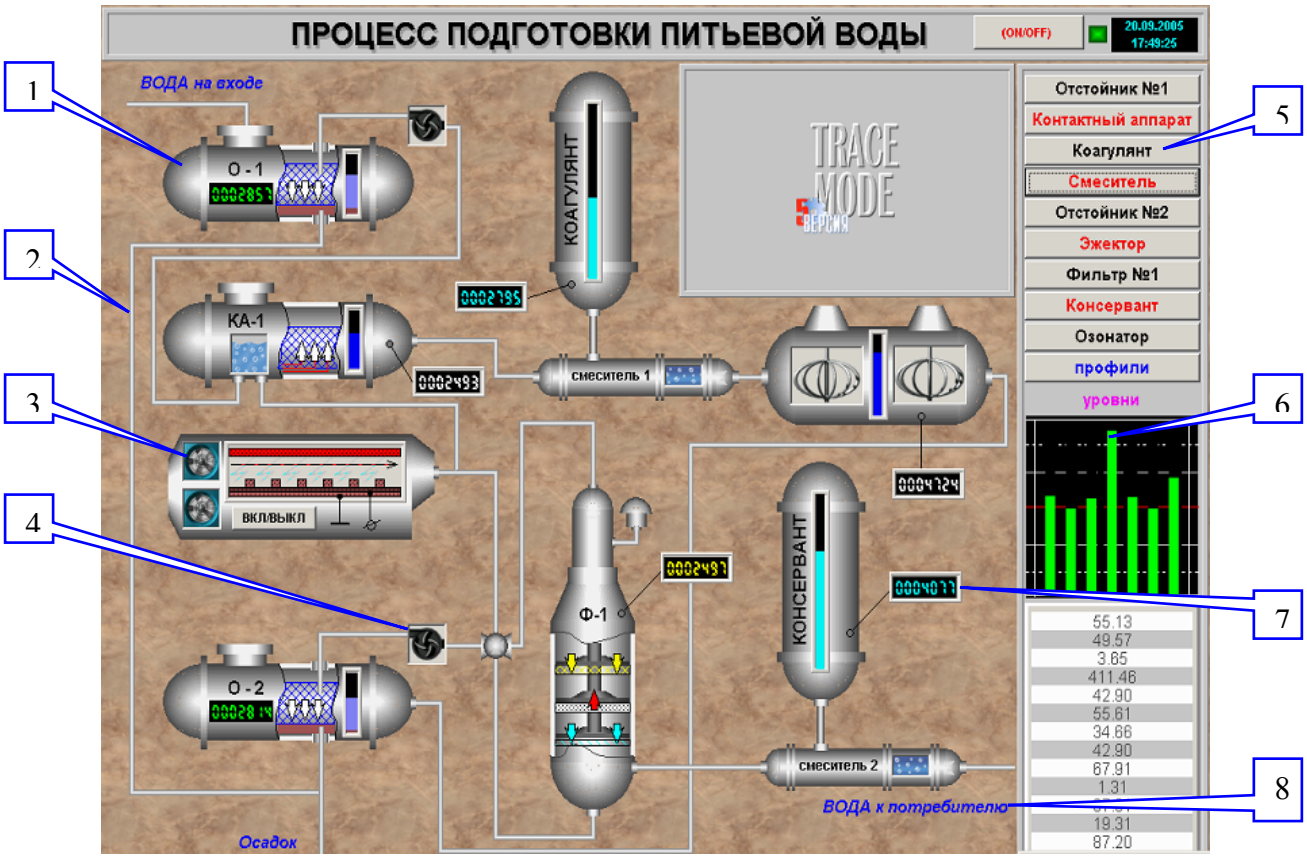


Рис. 134. Пример графического интерфейса оператора, контролирующего процесс подготовки питьевой воды, созданного в системе Trace Mode

На рис.134 обозначено: 1 – емкость (отстойник); 2– трубопровод; 3 – анимация; 4 – вентилятор; 5 – кнопка; 6 – гистограмма; 7 – динамический текст; 8 – статический текст.

Следует обратить внимание на некоторые детали интерфейса, соответствующие данным ранее рекомендациям по проектированию, облегчающим работу оператора с графическими объектами. Например, количество динамических объектов, за которыми должен одновременно следить диспетчер, незначительно превышает число «кошелька Миллера» – 7+2=9. Вход для воды предусмотрен вверху слева, а выход – с противоположной стороны, т.е. внизу справа (контраст). Гистограмма изображена ярко зелеными столбиками на черном фоне, что очень хорошо читается даже при беглом взгляде. Текст на включенных кнопках меняет цвет с черного на красный, привлекая, таким образом, внимание оператора, и т.д.

5.9.2. SCADA-система Citect

На рис. 135 и 136 приведен пример графического интерфейса оператора, контролирующего процесс фильтрации меди и получения обогащенного медного концентрата. Данный графический интерфейс создан с помощью австралийской SCADA-системы Citect.

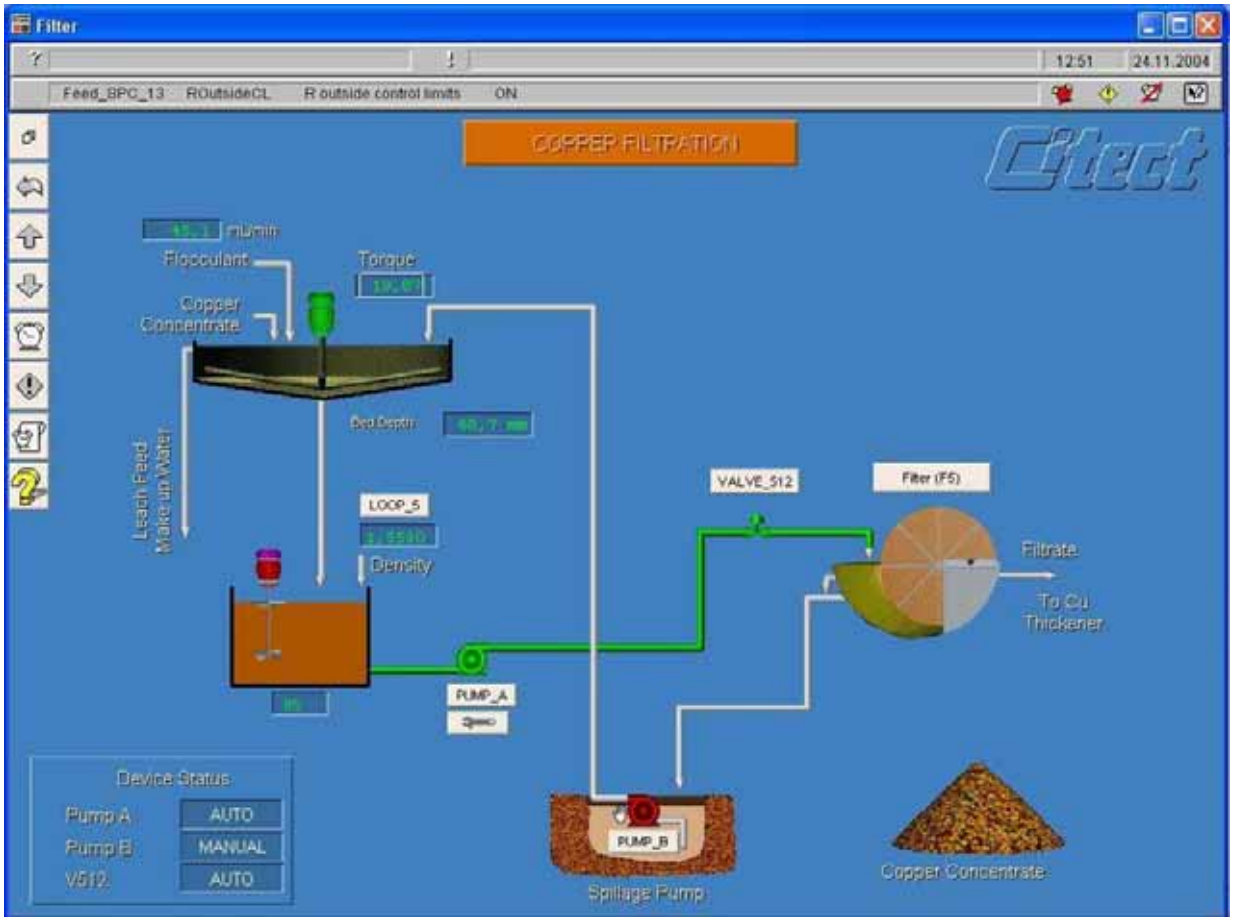


Рис. 135. Пример графического интерфейса оператора, контролирующего процесс фильтрации меди, созданного в системе Citect

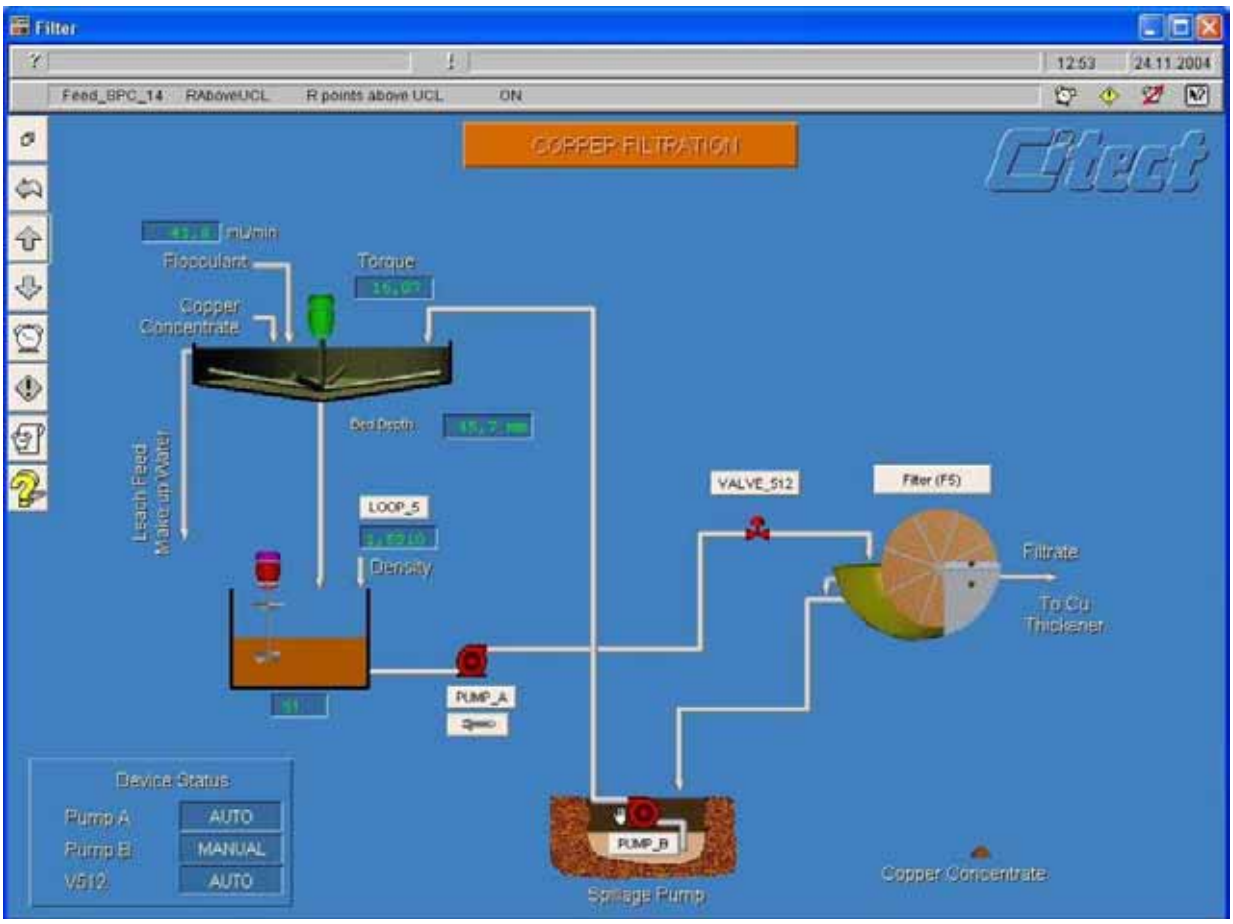


Рис. 136. Пример графического интерфейса оператора, контролирующего процесс фильтрации меди, созданного в системе Citect. Скриншот другой стадии процесса

Здесь также хорошо просматриваются соблюдаемые принципы построения графических интерфейсов («кошелек Миллера», контраст оранжевого на голубом фоне, «зеленый – открыто, красный – закрыто» и т.п.), хотя мы с австралийцами не договаривались! Следовательно, объективные принципы проектирования графических интерфейсов действительно существуют и применяются во всем мире.

На рис. 137 и 138 показаны две разные стадии процесса управления сложным объектом – краном, оснащенным схватом, на мобильной платформе. Данные элементы графического интерфейса также созданы с помощью SCADA-системы Citect, версия 5.42.

Рис. 137. Пример элементов графического интерфейса оператора, контролирующего процесс управления сложным объектом – мобильным краном, оснащенным схватом. Созданы с помощью системы Citect 5.42

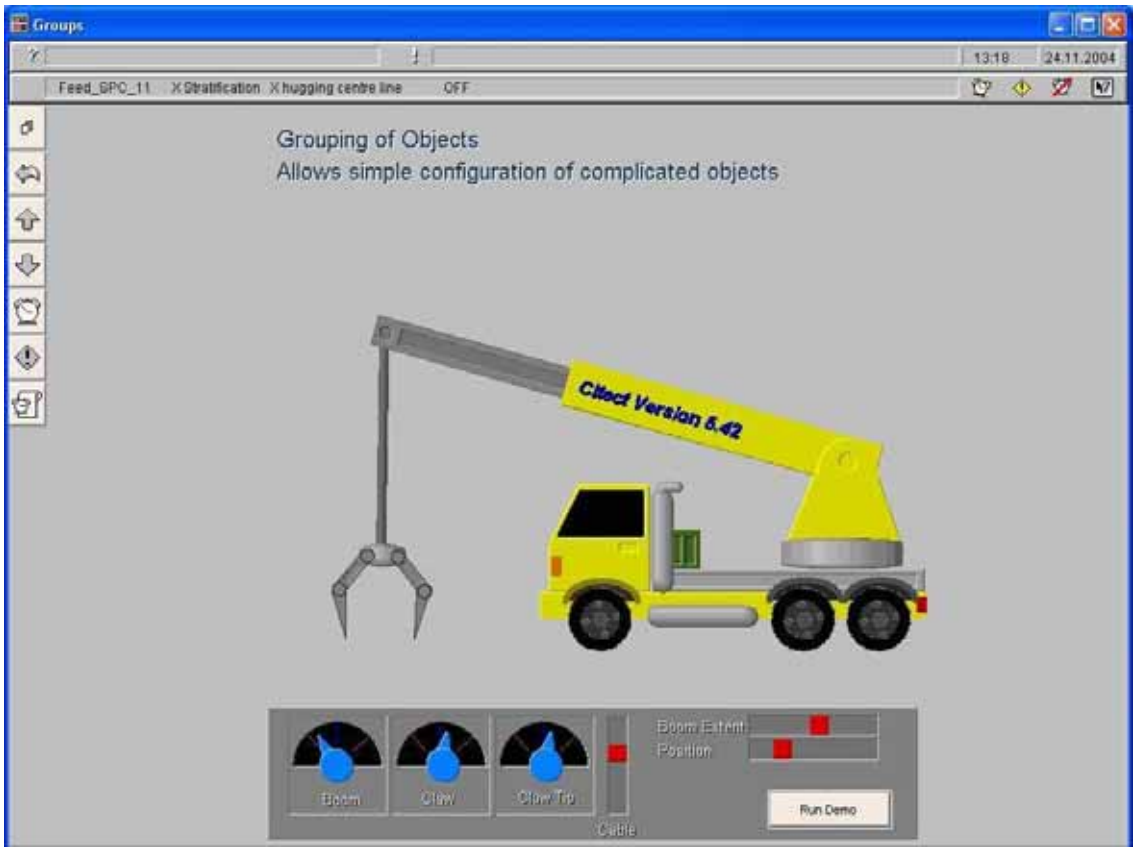


Рис. 138. Пример элементов графического интерфейса оператора, контролирующего процесс управления мобильным краном со схватом, созданных с помощью системы Citect 5.42. Другая стадия процесса

Заключение

Подводя итоги, выражаю надежду на то, что читатели, давшие себе труд изучить эту книгу, стали лучше представлять себе, каким именно образом человек воспринимает графическую информацию, какие ограничения и какие предпочтения он при этом имеет. Такое понимание необходимо при проектировании графических интерфейсов для операторов, управляющих самыми различными процессами. Знание общих принципов построения графических интерфейсов оператора позволяет применять их в разнообразных сферах прикладной деятельности, как инженерам-проектировщикам, так и программистам, а также специалистам по системам управления с человеком и по информационным системам.

С другой стороны, изучение основ компьютерной графики призвано вооружить инженера знаниями о том, как воплотить в реальность задуманный удобный и эффективный графический интерфейс для оператора, управляющего данным конкретным процессом, будь то динамический объект, технологический процесс или что-либо иное. Приемы и алгоритмы, используемые при создании графических объектов, позволяют заглянуть на «кухню» компьютерной графики и понять, каким образом несколькими щелчками мышью получаются сложные фигуры и модели, как они трансформируются, преображаются, окрашиваются, двигаются. Эти знания могут пригодиться в еще более обширных прикладных областях, так как при исследованиях, проектировании, представлении результатов анализа различных ситуаций, демонстрации разнообразных проектов справедливой остается поговорка: «Лучше один раз увидеть, чем сто раз услышать». Увиденное наводит на мысли, которые иначе не могли бы появиться. Таким образом, инструментарий, используемый при создании графических интерфейсов оператора, в более широкой перспективе может быть применен студентами, аспирантами, инженерами для достижения креативных и исследовательских целей.

Список литературы

1. Шикин Е.В., Боресков А.В. Компьютерная графика. Полигональные модели. – М.: «Диалог-МИФИ», 2000. – 464 с.
2. Куликовский Л.Ф., Мотов В.В. Теоретические основы информационных процессов. – М.: «Высшая школа», 1987. – 248 с.
3. Башмаков А.И., Башмаков И.А. Интеллектуальные информационные технологии. – М.: Издательство МГТУ им. Н.Э.Баумана, 2005. – 304 с.
4. Р.С. Гиляревский. Информатика как наука об информации. М.: ФАИР, 2006.
5. Якоб Нильсен. Веб-дизайн. – СПб: Символ-Плюс, 2003. – 512 с. – ISBN 5-93286-004-9.