Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

О. А. Чумаков, Н. А. Стасевич

ОСНОВЫ СИСТЕМ АВТОМАТИЗИРОВАННОГО ПРОЕКТИРОВАНИЯ

Рекомендовано УМО по образованию в области информатики и радиоэлектроники в качестве учебно-методического пособия для специальности 1-53 01 07 «Информационные технологии и управление в технических системах»

Минск БГУИР 2012

УДК 004.896(042.4)
ББК 30.2-5-05я73
Ч-90

Р е ц е н з е н т:
заведующий кафедрой автоматизации технологических процессов и электротехники учреждения образования «Белорусский государственный технологический университет», кандидат технических наук, доцент И. Ф. Кузьмицкий;

кафедра «Информационные системы и технологии» Белорусского государственного национального технического университета (протокол №3 от 26.11.2011 г.)

Чумаков, О. А.
Ч-90 Основы систем автоматизированного проектирования : учеб.-метод. пособие / О. А. Чумаков, Н. А. Стасевич. – Минск : БГУИР, 2012. – 95 с.
ISBN 978-985-488-621-3.

В пособии рассматриваются принципы автоматизации проектирования, а также вопросы архитектуры и информационного обеспечения САПР. Подробно изложены современные технологии проектирования, включая интегрированные системы CAD/CAM/CAE и концепцию CALS.

Для специальности 1-53 01 07 «Информационные технологии и управление в технических системах».

УДК 004.896(042.4)
ББК 30.2-5-05я73

ISBN 978-985-488-621-3

СОДЕРЖАНИЕ

Введение

В настоящее время при изготовлении чертежей и прочей конструкторской документации системы автоматизированного проектирования (САПР) практически полностью вытеснили традиционный способ черчения – кульман. Использование компьютера предоставляет конструкторам и технологам множество преимуществ в изготовлении чертежей, освобождает их от рутинной работы, а также резко повышает производительность труда (по некоторым оценкам в 2–2,5 раза). САПР ориентированы на работу в интерактивном режиме, предоставляя проектировщику оперативный доступ к графической информации, простой и эффективный язык управления ее обработкой с практически неограниченными возможностями контроля результатов. В результате удается автоматизировать самую трудоемкую часть работы (в процессе традиционного проектирования на разработку и оформление чертежей приходится около 70 % от общих трудозатрат конструкторской работы, 15 % – на организацию и ведение архивов и 15 % – собственно на проектирование, включающее в себя разработку конструкции, расчеты, согласования и т. д.).

В данном конспекте лекций раскрываются принципы автоматизации проектирования, а также рассматриваются вопросы архитектуры и информационного обеспечения САПР. Подробно изложены современные технологии проектирования, включая интегрированные системы CAD/CAM/CAE и концепцию CALS. Приводятся базовые сведения о принципах построения систем графического моделирования, а также об основных графических стандартах. Проводится сравнительный анализ наиболее распространенных систем графического моделирования: DUCT, CATIA, AutoCAD, I-DEAS, Unigraphics.

В пособии также изложены основные принципы работы автоматизированных систем для проектирования изделий и подготовки их производства (без углубления и детализации конкретных систем). Рассмотрены компоненты САПР, основные концепции графическою программирования, системы автоматизированной разработки чертежей, системы геометрического моделирования, представление кривых и поверхностей и работа с ними, стандарты обмена данными между системами.

1. Современные технологии проектирования

1.1. Интегрированные системы CAD/CAM/CAE

Определяющими факторами успеха в промышленном производстве являются уменьшение времени выхода продукции на рынок, повышение ее качества и снижение стоимости. Практическая реализация этих требований обусловливает необходимость модернизации проектно-технологических и производственных процессов как в рамках отдельных предприятий, так и в условиях «расширенного предприятия», объединяющего всех поставщиков, соисполнителей и участников проектирования и производства продукции. В настоящее время наиболее радикальным средством решения задач модернизации является внедрение *интегрированных информационных технологий* на базе использования современных средств вычислительной техники и сетевых решений. К числу наиболее эффективных технологий, дающих весомый выигрыш в короткие сроки, принадлежат системы автоматизированного проектирования, инженерного анализа и технологической подготовки (CAD/CAM/CAE), а также системы управления производственной информацией (PDM).

Первым наиболее значительным результатом в области разработки программных средств САПР является создание интерактивных графических редакторов (систем автоматизированного черчения). В сущности, все графические редакторы работают одинаково: для них определены элементарные геометрические объекты (примитивы), а также процедуры манипулирования с этими примитивами (редактирование). Поэтому в таких редакторах реализованы упрощенные представления о процессе проектирования как о процессе создания геометрических объектов путем манипуляции с набором геометрических примитивов. Очевидно, что такие представления недостаточно точно отражают работу инженера-конструктора, не позволяют ему отличить ее от деятельности чертежника, которая полностью ограничивается рамками изготовления технической документации.

Специализация графических редакторов для САПР привела к появлению ряда утилит, одни из которых встраивались в ядро редактора (например утилита образмеривания), а другие предполагалось применять как независимые сервисные программы (утилита параметрического проектирования и пр.). Специализация, безусловно, улучшила эффективность использования САПР, но ничего не изменила принципиально. В настоящее время развитие программных средств САПР идет в направлении решения довольно небольшого круга проблем, к которым в первую очередь относятся: *эффективность твердотельного моделирования, параметризация,* а также *ассоциативность и программный интерфейс*.

1.2. Жизненный цикл изделия и его этапы

Анализ развития информационных технологий в производственных задачах показывает, что основной тенденцией является все более полный охват стадий жизненного цикла продукции. Гибкие производственные системы решали задачи, касающиеся исключительно производства изделий. В компьютеризированном интегрированном производстве круг решаемых задач значительно расширился и включил в себя разработку, проектирование и изготовление, материально-техническое обеспечение и другие задачи предприятия. Тем не менее остались нерешенными следующие задачи: взаимодействие с заказчиком, взаимодействие партнерами-поставщиками, послепродажное сопровождение изделия и многие другие.

К середине 90-х г. XX в. появилось осознание необходимости создания интегрированной информационной системы, поддерживающей весь жизненный цикл изделия.

Жизненный цикл (ЖЦ) продукции – это совокупность процессов, выполняемых от момента выявления потребностей общества в определенной продукции до момента удовлетворения этих потребностей и утилизации продукции. К основным стадиям ЖЦ относятся: маркетинг; проектирование и разработка продукции; планирование и разработка процессов; закупки материалов и комплектующих; производство или предоставление услуг; упаковка и хранение; реализация; монтаж и ввод в эксплуатацию; техническая помощь и сервисное обслуживание; послепродажная деятельность или эксплуатация; утилизация и переработка в конце полезного срока службы.

Многообразие процессов в ходе ЖЦ и необходимость их интенсификации требуют активного информационного взаимодействия субъектов (организаций), участвующих в поддержке ЖЦ продукции. С ростом числа участников растет объем используемой и передаваемой информации.

Потребность в создании интегрированной системы поддержки ЖЦ изделия и организации информационного взаимодействия компонентов такой системы приводят к необходимости создания *интегрированной информационной среды* (ИИС), в основе которой лежит использование открытых архитектур, международных стандартов, совместное использование данных и программно-технических средств.

В сложных долговременных проектах ИИС обеспечивает взаимодействие проектных организаций и производственных предприятий, поставщиков, организаций сервиса и конечного потребителя на всех стадиях ЖЦ. В проектах, финансируемых или контролируемых государством, к необходимой информации могут иметь доступ уполномоченные государственные структуры.

Данный подход характеризуется следующими особенностями:

– в отличие от компьютерной автоматизации и интеграции отдельных процессов, например, в производстве решаются задачи информационной интеграции всех процессов ЖЦ;

– решаемые задачи выходят за границы отдельного предприятия, участники информационного взаимодействия могут быть территориально удалены друг от друга, располагаться в разных городах и даже странах;

– совместно используемая информация очень разнородна: это маркетинговые, конструкторско-технологические, производственные данные, коммерческая и юридическая информация и т. д. Для ее совместного использования должны быть стандартизованы способы, технологии представления и корректной интерпретации данных;

– основной средой передачи данных является глобальная сеть Интернет.

1.3. Технологии управления производственной информацией

В рамках технологии CALS развиваются современные технологии управления производственной информацией, часто называемые PDM-системами (Product Data Management). Они следят за большими, постоянно обновляющимися массивами данных и инженерно-технической информации. В отличие от баз данных PDM-системы интегрируют информацию любых форматов и типов, поступающую от различных источников, предоставляя ее пользователям в структурированном виде, отражающем особенности современного промышленного производства. Системы PDM отличаются также и от интегрированных систем офисного документооборота, поскольку текстовые документы являются далеко не самыми «нужными» на производстве (куда важнее геометрические модели, данные для функционирования автоматических линий, станков с ЧПУ и т. п.). Системы PDM обобщают такие широко известные технологии, как управление инженерными данными (Engineering Data Management – EDM), управление документами, управление информацией об изделии (Product Information Management – PIM), управление техническими данными (Technical Data Management – TDM), управление технической информацией (Technical Information Management – TIM), управление изображениями и пр.

Любая информация, необходимая на том или ином этапе ЖЦ изделия, может управляться системой PDM, которая предоставляет корректные данные всем пользователям и всем промышленным информационным системам. Наряду с данными PDM управляет и проектом – процессом разработки изделия, контролируя собственно информацию об изделии, о состоянии объектов данных, об утверждении вносимых изменений, осуществляя авторизацию и другие операции, которые влияют на данные об изделии и режимы доступа к ним каждого конкретного пользователя.

Системы PDM играют роль связующего звена между этапом инженерно-конструкторской подготовки нового изделия и системами MRP (Manufacturing

Resource Planning) или, другими словами, разного рода АСУ, решающими задачи автоматизации управления финансами, складским хозяйством, снабжением и сбытом, а также техническим обслуживанием. О важности такого рода систем свидетельствует хотя бы такой факт, что только 25 % рабочего времени персонала компании, начиная от проектировщика и кончая руководителем проекта, тратится на собственно творческую работу, а остальное время идет на поиск информации и стыковку потоков данных, поступающих от разных подразделений. Часто оказывается, что проще заново разработать деталь, чем найти информацию, подготовленную некоторое время назад.

Место систем PDM в общей производственной цепочке показано на рис. 1.1. Они занимают промежуточное положение между системами MPR и системами CAD/CAM/CAE, которые в отечественной практике называют одним термином – интегрированные САПР. В англоязычной литературе под указанными терминами понимают следующее:

CAD (Computer-Aided Design) – общий термин для обозначения всех аспектов проектирования с использованием средств вычислительной техники; обычно охватывает создание геометрических моделей изделия (твердотельных, трехмерных, составных), а также генерацию чертежей изделия и их сопровождений;

CAM (Computer-Aided Manufacturing) – общий термин для обозначения программных систем подготовки информации для станков с ЧПУ; традиционно исходными данными для таких систем были геометрические модели деталей, получаемые из систем CAD;

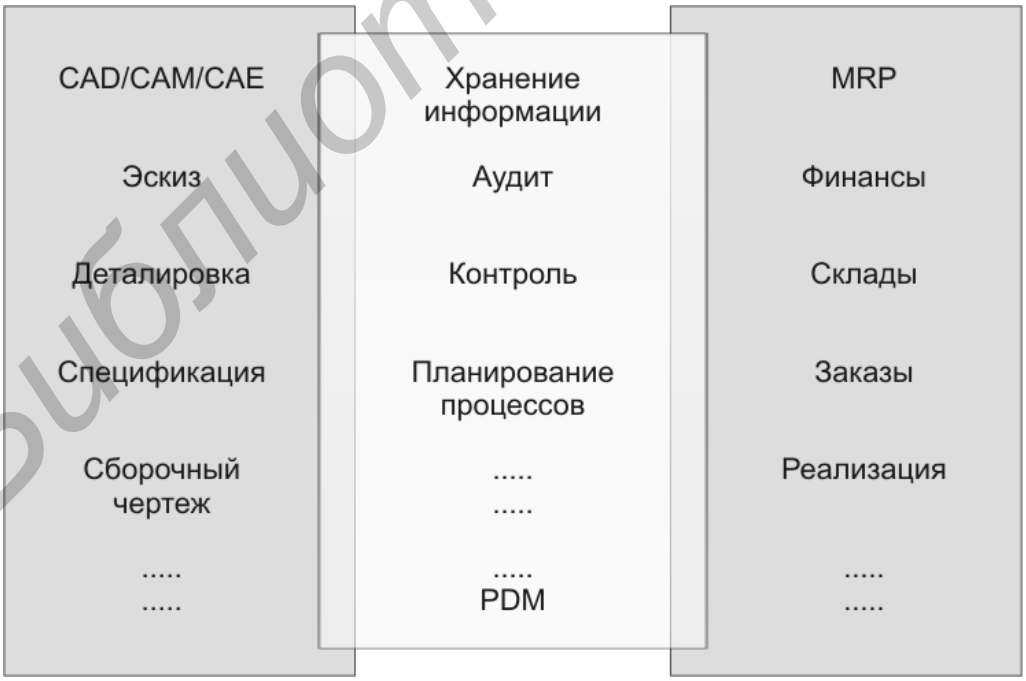


Рис. 1.1. Взаимосвязь систем автоматизации производственных процессов

CAE (Computer-Aided Engineering) – общий термин для обозначения информационного обеспечения автоматизированного анализа проекта (прочностные расчеты, коллизии кинематики и т. п.) или оптимизации производственных возможностей.

Главное направление развития современных САПР – повышение их интеллектуальных функций, т. е. способности «понимать» намерения конструкторов. В простейшем случае в системе запоминается лишь «история» или последовательность шагов, выполняемых проектировщиком. Такие системы удобны при создании библиотек стандартных деталей и элементов, но для более сложных ситуаций требуется более «интеллектуальная» реализация пользовательского интерфейса. Поэтому в САПР начинает все шире использоваться *объектная технология*, в соответствии с которой САПР должны не работать с файлами, а обрабатывать объекты. Объекты образуют собой «целостности», включающие множественные непротиворечивые представления одной и той же «сущности». Например, деталь может представлять интерес для дизайнера с позиции эстетики формы, для инженера – с позиции вычислительной сложности поверхности, для технолога – с позиции применимости процесса штамповки для ее изготовления. Объект позволяет объединить подобные представления, что открывает прямой путь к эффективной реализации идей С-технологии, т. е. параллельного проектирования и инжиниринга (concurrent design and engineering).

С-технология (конструкторско-технологическое проектирование) – это принципиально новый, интегрированный подход к проектированию. В ее основе лежит идея совмещенного проектирования изделия, а также процессов его изготовления и сопровождения, координируемых с помощью специально создаваемой для этой цели распределенной информационной среды. Подобная технология позволяет использовать проектные данные, начиная с самых ранних стадий проектирования, одновременно различными группами специалистов. Например, в трех главных конструкторских бюро компании Boeing действуют 220 групп «проектирование-производство», которые координируют параллельные разработки и состоят из специалистов таких разнообразных областей, как конструирование, технология материалов, производство и взаимодействие с клиентами. С-технология обеспечивает устранение известных недостатков последовательного проектирования, в частности, в случае, когда ошибки проекта изделия неожиданно обнаруживаются на последних его стадиях. Кроме того, появляется возможность легко и быстро вносить изменения в проект, причем таким образом, чтобы изменения не вызывали повторного проектирования созданных деталей и узлов. Сегодня «перепроектирование» продолжает оставаться существенной затратной компонентой любой разработки.

Следует обратить внимание на интересные инициативы в области САПР, возникшие в Германии, связанные с проблемой роста несовместимости решений, предлагаемых многочисленными производителями информационной

техники, включая и CAD/CAM/CAE-системы. Решение этой проблемы стало настолько насущным, что поставлен вопрос о стандартизации систем CAD/CAM/CAE и информационной техники в целом. Концерн Daimler-Benz выступил с предложением под названием «Инициатива по передовой информационной технике», которое поддержали British Aerospace, FIAT, Renault, SAAB, Volkswagen и многие другие компании. Другой проект под названием CAD2000 объединил компании Audi, BMW, Mercedes-Benz, Porsche, Volkswagen. Эти проекты пытаются решить громадную по масштабам и сложности проблему поиска стандартных решений, способных удовлетворить огромное множество прикладных требований от проектирования до изготовления, а также управления информационными данными и библиотеками стандартных компонентов.

1.4. Концепция CALS

Современное представление о процессе проектирования исходит из его «генетического» единства с процессом производства. С этой точки зрения проектирование является информационной моделью производства, а никак не процессом изготовления технической документации. Следует отметить, что ранее конструкторы не имели инструментов для проверки адекватности указанных процессов, поэтому и появилась специальность технолога, который по сути дела осуществляет «переформатирование» описания из форм, адекватных процессу проектирования, в форму, адекватную процессу производства. Но с появлением современных средств вычислительной техники стала возможна непосредственная передача информации от компьютеров к элементам производства, хотя, как правило, необходимость изготовления технической документации сохраняется.

Впервые работы по созданию интегрированных систем, поддерживающих ЖЦ продукции, были начаты в 80-х гг. XX в. в оборонном комплексе США. Новая концепция была востребована жизнью как инструмент совершенствования управления материально-техническим обеспечением армии США. Предполагалось, что реализация новой концепции, получившей обозначение CALS (Computer Aided Logistic Support – компьютерная поддержка процесса поставок), позволит сократить затраты на организацию информационного взаимодействия государственных учреждений с частными фирмами в процессах формализации требований, заказа, поставок и эксплуатации военной техники. Появилась реальная потребность в организации ИИС, обеспечивающей обмен данными между заказчиком, производителями и потребителями, а также повышение управляемости, сокращение бумажного документооборота и связанных с ним затрат. Доказав свою эффективность, концепция последовательно совершенствовалась, дополнялась и, сохранив существующую аббревиатуру (CALS), получила более широкую трактовку – Continuous Acquisition and Life cycle Support – непрерывная информационная поддержка поставки и ЖЦ продукции.

Первая часть – Continuous Acquisition [Support] (непрерывная поддержка поставки) означает непрерывность информационного взаимодействия с заказчиком в ходе формализации его потребностей, формирования заказа, процесса поставки и т. д. Вторая часть – Life Cycle Support (поддержка ЖЦ изделия) – означает системность подхода к информационной поддержке всех процессов ЖЦ изделия, в первую очередь процессов эксплуатации, обслуживания, ремонта и утилизации и т. д.

В период 1990–2000 гг. в мире был выполнен ряд проектов, направленных на апробацию и внедрение принципов CALS в различных отраслях промышленности. Поскольку термин CALS всегда носил военный оттенок, в гражданской сфере широкое распространение получили термины Product Life Cycle Support (PLCS) или Product Life Management (PLM) – «поддержка жизненного цикла изделия». К PLM относятся практически все средства и системы автоматизации: конструкторские и технологические САПР (CAD/CAM/CAE, CAPP), системы ERP (MRP), средства управления взаимодействием с клиентами (CRM), цепочками поставок (SCM), техническим обслуживанием (сервисом) и т. д. Сюда же относятся PDM-системы, которым отводится ключевая роль в организации информационного взаимодействия всех участников ЖЦ изделия через ИИС.

2. Графические стандарты

2.1. Форматы графических файлов

При автоматизированном проектировании возникает необходимость обмениваться графической информацией между различными подсистемами, которые в общем случае могут быть реализованы не только в различной программной среде, но и на различных аппаратных средствах. При этом важно правильно выбрать формат записи данных, который, с одной стороны, должен обеспечивать минимальный размер файлов, а с другой – сохранение точности графической модели изделия. Поэтому рассмотрим подробнее вопросы кодирования графической информации или, сокращенно, графические форматы. Для записи графической информации используются два принципиально различных формата – растровый и векторный. Первый применяется при обработке изображений, полученных с помощью сканера, а также при редактировании фотоизображений; второй – в системах автоматизированного проектирования и графических пакетах.

Растровый формат (Adobe Photoshop, Paint) описывает изображение как совокупность точек (dots), число которых определяется разрешением, измеряемым в специальных единицах – dpi или dpc (число точек на 1 дюйм или на 1 см соответственно). Для цветных и полутоновых изображений определяется также «глубина цвета» – число двоичных разрядов, отводимых

для хранения информации о цвете. Например, для изображений типа RGB глубина 24 бита означает, что на каждый основной цвет (красный – R, зеленый – G, синий – B) отводится по 8 бит, поэтому цветовая палитра состоит из $224 \approx 16$ млн цветов.

Основной недостаток растрового формата – большой объем графического файла. Так, даже для растрового изображения со сравнительно невысоким разрешением 1024×768 точек и 256 цветами требуется 768 кб. Поэтому в графических пакетах используются различные алгоритмы сжатия, что затрудняет преобразование растрового формата и создает множество проблем при использовании его в САПР.

Векторный формат (AutoCAD, CorelDRAW, Visio и др.) описывает изображение как совокупность простейших примитивов (линия, ломаная, кривая Безье, эллипс, прямоугольник и т. д.), для каждого из которых задаются соответствующие атрибуты: координаты вершин или других характерных точек, толщина и цвет контурной линии, тип и цвет заливки и т. д. Кроме того, задается расположение элементов относительно друг друга (какой из них расположен сверху, а какой – снизу). Главное достоинство векторных файлов по сравнению с растровыми – меньший размер и удобство редактирования, но при их выводе на экран производится множество математических операций. Поэтому скорость вывода векторных изображений обычно меньше, чем растровых, хотя этот недостаток довольно эффективно устраняется при помощи специальных процессоров – графических ускорителей.

Существует целый ряд программ, переводящих графические данные из векторного формата в растровый. Но обратная задача (перевод растровых изображений в векторные) является довольно сложной и решается только в наиболее совершенных графических пакетах. Не менее сложные проблемы возникают и при преобразованиях одного векторного формата в другой, так как многие графические пакеты используют уникальные математические модели для элементов изображения. В настоящее время применяют несколько десятков форматов представления графических данных. Рассмотрим наиболее распространенные из них (табл. 2.1).

Таблица 2.1

Форматы графических файлов

Тип	**Название**	**Описание**
BMP	Bytemap	Растровый. Поддерживается практически любым Windows-приложением. Независим от типов адаптера и монитора при кодировании цвета. Для сжатия данных используется алгоритм RLE (Run-Length Encoded). Размер файлов довольно большой. Использует индексированные цвета либо цветовую модель RGB
3DS	3D Studio	Универсальный формат хранения полигональной графики

Продолжение табл. 2.1

Тип	Название	Описание
TIFF	Tagged Image File Format	Растровый. Универсальный формат хранения сканированных изображений с цветовыми каналами. Импортируется в программы настольных издательских систем. Позволяет хранить изображения с любой глубиной цвета и цветовой моделью. Поддерживаются дополнительные каналы масок, калибровочная информация, алгоритмы сжатия без потерь информации. Начиная с шестой спецификации позволяет хранить простейшие объектные контуры. Ориентирован на типографскую печать. Приводит к значительным размерам файлов (цветное изображение А4 с разрешением 300 dpi занимает около 40 Мб)
GIF	Graphics Interchange Format	Растровый. Создан для передачи изображений в Internet. Использует сжатие без потерь информации. Хранит индексированные изображения в 256-цветной палитре с одним прозрачным цветом. Возможно хранение анимации
PNG	Portable Network Graphics	Растровый. Предназначен для передачи изображений в сетях. Использует эффективный алгоритм сжатия без потерь информации. Поддерживает полноцветные и индексированные изображения
JPEG	Joint Photographic Experts Group	Растровый. Использует сжатие с потерями информации. Работает с полноцветными изображениями в моделях RGB и CMYK, а также полутоновыми. Не поддерживаются дополнительные каналы. Используется для верстки Web-страниц и хранения фотографий
JPEG 2000	Joint Photographi c Experts Group	Новый вариант JPEG. Имеет усовершенствованный алгоритм с высокой степенью сжатия, что позволяет добиваться большей эффективности, меньше жертвуя качеством. Исключает характерный для JPEG эффект, связанный с блочной обработкой изображений (видимые квадраты 8×8 пикселов). Существует в двух вариантах: стандартном (Standard, расширение jp2) и оптимизированном для Web (Codestream, расширение jpc)
CGM	Computer Graphics Metafile	Метафайл. Обеспечивает кодирование как векторных, так и растровых изображений. Независим от программных средств, что позволяет эффективно осуществлять обмен данными между различными программами и платформами. Но для хранения чертежей и рисунков этот формат практически не применяется
DWG	Drawing	Векторный. формат САПР AutoCAD
IGES	Iintial Graphical Exchange Standard	Представляет собой набор протоколов для передачи графических данных и вывода их на экран монитора. В настоящее время используется в ряде CAD-приложений, оперирующих с трехмерными изображениями

Окончание табл. 2.1

Тип	Название	Описание
DXF	Drawing eXchange Format	Векторный. Поддерживается практически всеми САПР и графическими пакетами. Позволяет хранить трёхмерные объекты, однако из-за его сложности многие приложения читают DXF-файлы, но не сохраняют в нём данные
CDR	CorelDraw	Векторный. Формат программы CorelDraw позволяет сохранять векторные изображения и растровую графику и текст
EMF	Enhanced metafile	Аппаратно-независимый формат хранения векторной графики
WMF	Windows MetaFile	Обеспечивает кодирование как векторных, так и растровых данных и является аналогом формата PICT для оболочки Windows. Служит для передачи векторов через буфер обмена (Clipboard). Понимается практически всеми программами Windows, связанными с векторной графикой. Искажает цвет, не может хранить ряд параметров, которые могут быть присвоены объектам в различных векторных редакторах

2.2. Обмен графической информацией

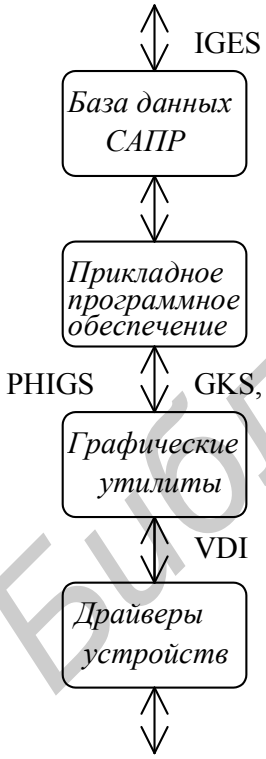


Рис. 2.1. Графические стандарты

При организации обмена графической информацией в компьютерной графике различают несколько уровней графических стандартов (рис. 2.1). Эти стандарты обеспечивают связь между:

а) графическими утилитами и устройствами графического вывода;

б) прикладными программами и графическими утилитами;

в) различными САПР.

Для обеспечения связи между графическими утилитами и устройствами вывода наиболее часто используется стандарт VDI (Virtual Device Interface – интерфейс виртуального устройства), который в настоящее время переименован в CGI (Computer Graphics Interface – интерфейс компьютерной графики).

Наиболее распространенным стандартом, обеспечивающим связь между прикладными программами и графическими утилитами, является GKS (Graphical Kernel System – графическая корневая система). Иногда используется и более ранний стандарт CORE, основные функции

которого реализованы в GKS. А наиболее совершенным из стандартов этого класса является PHIGS (Programmers Hierarchical Interface for Graphics – программистский иерархический графический интерфейс), описывающий сложные иерархические структуры графических данных, в том числе и трехмерные.

Для обеспечения связи между различными САПР используется ряд стандартов, наиболее распространенным из которых является IGES (стандартный протокол обмена графической информацией). В этом стандарте различные данные классифицируются в терминах сущностей, которые могут принадлежать к одной из трех категорий: геометрии (точки, отрезки, дуги, плоскости и т. п.), аннотации (размеры, осевые линии, стрелки и т. п.), структуры (геометрические группы, макроопределения и т. д.). Чтобы использовать IGES, любая САПР снабжается двумя программами – препроцессором и постпроцессором (рис. 2.2).

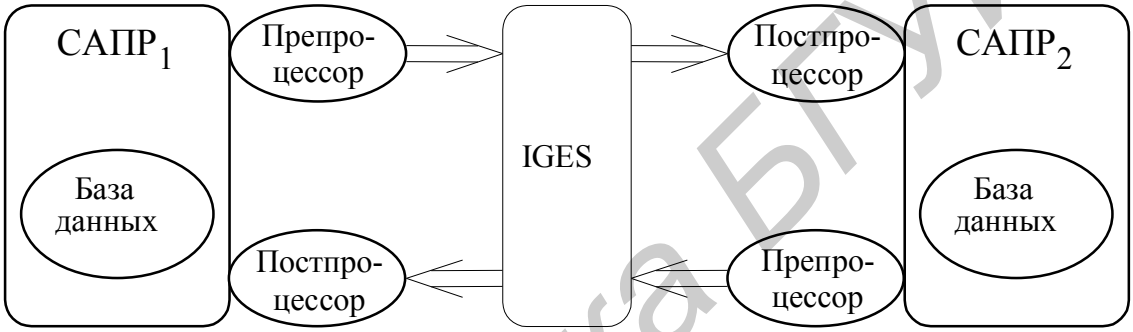


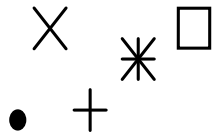
Рис. 2.2. Связь между двумя различными САПР через формат IGES

2.3. Стандартные графические примитивы

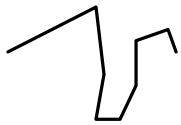
Международный стандарт GKS (Grapfical Kernel System) принят в 1985 г. и предназначен для обеспечения переносимости и совместимости программных средств машинной графики. Согласно этому стандарту, любое изображение должно строиться из типовых базовых элементов – примитивов вывода (рис. 2.3). В GKS определено шесть основных *примитивов вывода:* полимаркер, полилиния, текст, заполнение области, массив пикселей, обобщенный примитив вывода.

Полимаркер используется для указания характерных точек на экране, которые отображаются в виде ярких точек, крестов, квадратов и т. д. *Полилиния* представляет собой набор отрезков прямых (ломаную), соединяющих заданную последовательность точек. *Текст* – это строка символов, располагающаяся в указанной позиции. *Заполнение области* представляет собой многоугольник, заполненный штриховкой, узором или фоновой окраской. *Массив пикселей* позволяет задать цвет индивидуально для каждой точки некоторой области (пикселя). И, наконец, *обобщенный примитив вывода* представляет собой стандартное средство определения более сложных элементов (прямоугольник, эллипс и т. д.), вид и количество которых зависят от специфики конкретных графических пакетов.

Полимаркер
(PolyMarker)



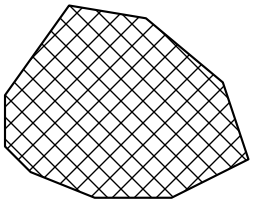
Полилиния
(PolyLine)



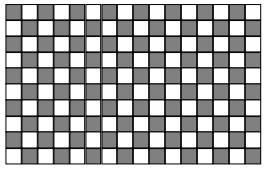
Текст
(Text)

Это текст
Это текст
Это текст

Заполнение области
(Fill Area)



Массив пикселов
(Cell Array)



Обобщенный примитив вывода
(Generalised Drawing Primitive)

Рис. 2.3. Основные графические примитивы вывода GKS

С каждым из примитивов в GKS связан набор параметров – атрибутов, определяющих его геометрические и качественные свойства. Для основных примитивов вывода используются следующие атрибуты:

- полимаркер тип маркера, его цвет и масштаб;
- полилиния тип, цвет и толщина линии;
- текст шрифт, размеры, цвет и ориентация символов;
- заполнение области вид штриховки, цвет;
- массив пикселов цвет пикселов.

GKS позволяет разделить изображение на отдельные сегменты, которые могут обрабатываться и отображаться независимо друг от друга. Предусмотрены также средства для включения одного сегмента в другой. При создании графической модели объекта и его изображения используются три типа систем координат: глобальная, нормализованная приборная и собственно приборная.

Ввод в GKS определяется как связь с одним из пяти допустимых логических устройств ввода:

- локатор выдает положение в глобальной системе координат;
- значение выдает значение числа;
- выбор выдает число, определяющее возможный вариант ответа;
- указание выдает имя сегмента и идентификатор примитива;
- строка обеспечивает ввод строки символов.

Ввод может происходить в одном из трех режимов: *запрос, опрос, событие.* Первый из указанных режимов аналогичен операции чтения обычных языков программирования: система ожидает события ввода, после чего передает в программу соответствующее значение. При этом в любой момент допустимо наличие только одного запроса на ввод. Второй режим применяется для ввода от таких устройств, у которых на выходе постоянно существует какое-либо значение (например положение указателя мыши). А третий режим используется для ввода от устройств, инициирующих прерывания. Эти прерывания запоминаются в очереди и обрабатываются в соответствии с принятой дисциплиной обслуживания. Появление GKS в качестве первого международного стандарта оказало существенное влияние на развитие машинной графики и ее применение в САПР.

3. Принципы построения систем графического моделирования

3.1. Основные этапы процесса проектирования

Можно выделить два подхода к конструированию.

Первый подход базируется на двухмерной геометрической модели – *графическом изображении* (ГИ) и использовании компьютера как электронного кульмана, позволяющего значительно ускорить процесс конструирования и улучшить качество оформления конструкторской документации. Центральное место в этом подходе к конструированию занимает чертеж, который служит средством графического представления изделия, содержащего информацию для решения графических задач, а также для изготовления изделия (рис. 3.1).

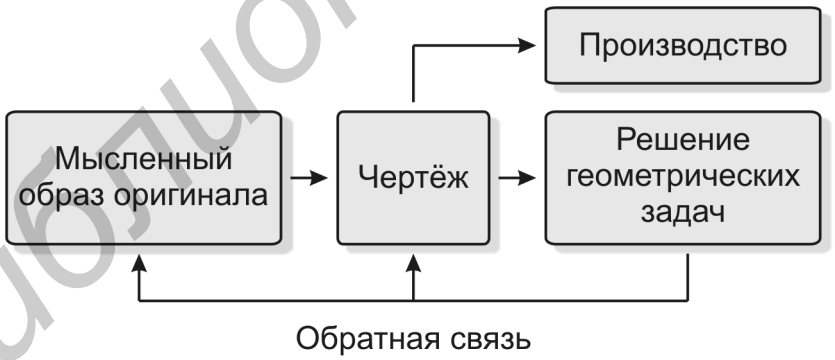


Рис. 3.1. Схема традиционной технологии конструирования

При таком подходе использование компьютера при решении графических задач должно быть рациональным и достаточно эффективным. Это может быть реализовано в том случае, когда созданное ГИ используется многократно или в различных вариациях, а формирование текстовых документов происходит автоматически в результате созданных чертежей и схем. Так, с помощью вычислительной техники облегчаются: оформление конструкторских документов, насыщенных изображениями стандартных, типовых,

унифицированных составных частей, например, электрических и других принципиальных, функциональных схем, печатных плат, модулей, приборов, электронных блоков, стоек, шкафов, пультов и т. д.; модернизация существующих конструкций (частичное изменение, а не создание принципиально нового); разработка текстовых документов (спецификаций, перечней элементов и др.).

В основе второго подхода лежит компьютерная пространственная геометрическая модель изделия (рис. 3.2), которая является более наглядным способом представления оригинала. При этом чертеж играет вспомогательную роль, а методы его создания основаны на методах компьютерной графики, методах отображения пространственной модели.

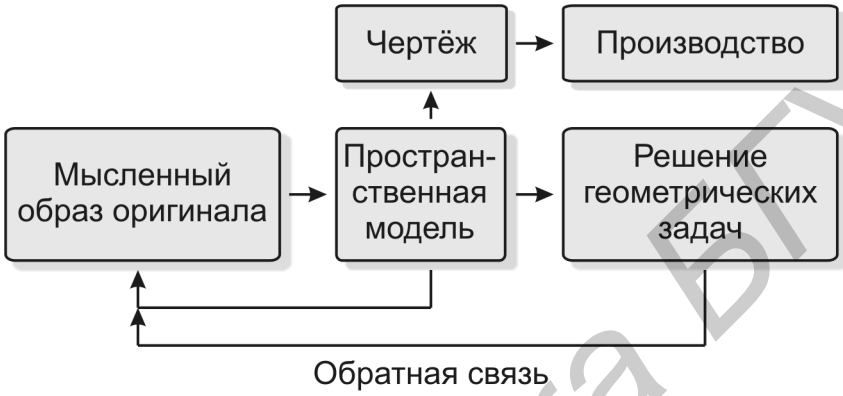


Рис. 3.2. Схема современной технологии конструирования

Процесс формирования моделей изделий можно разбить на несколько этапов.

На *первом* этапе (рис. 3.3) реальный объект (в примере – деталь) подвергается абстракции, в результате которой определяется информационная модель.

На *втором* этапе в информационной модели выделяют уровни структуризации данных и их взаимосвязи, чаще всего с учетом процессов обработки информации. Таким образом, осуществляются уточнение и структурирование информации с логической точки зрения. Существенным моментом в этом представлении является то, что оно должно отражать характеристики не одной детали, а целого класса деталей на различных стадиях проектирования, фиксируемых в технической документации.

При формировании информационной модели предполагается использование множества конструктивных элементов для получения деталей произвольной формы, геометрических элементов – точек, контуров, поверхностей, элементарных и сложных объектов, которые обеспечивают обработку геометрической информации для всех процессов автоматизированного проектирования. Таким образом строится модель данных, которая отражает их логическую структуру.

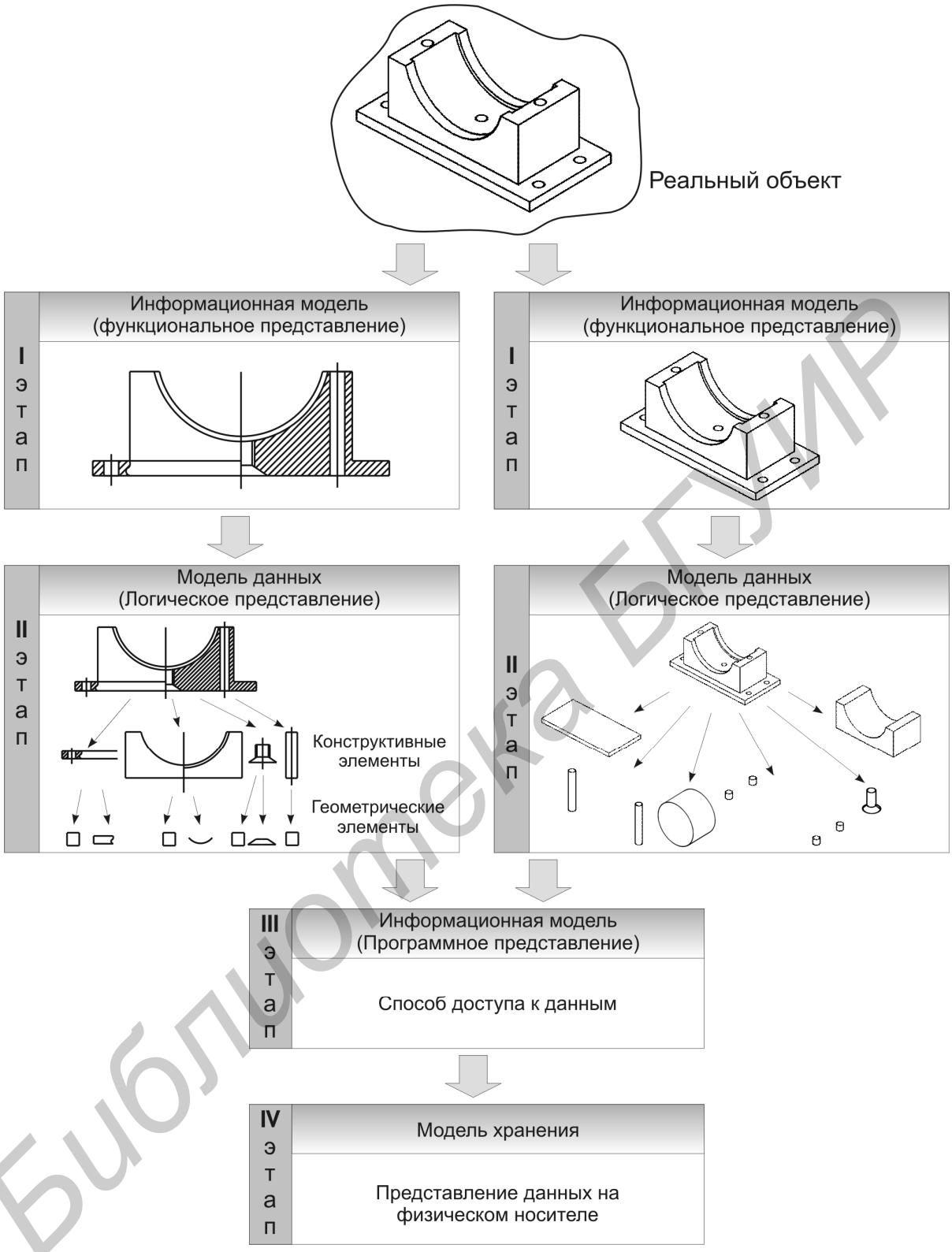


Рис. 3.3. Внутримашинное представление о реальном объекте

На *третьем* этапе осуществляется процесс преобразования модели данных во внутримашинное представление – формирование модели доступа. Модель доступа (или размещения) ориентирована на физическое размещение данных в памяти компьютера, в модели хранения.

Таким образом, на *четвертом* этапе определяется модель хранения, которая описывает способы организации данных, сформированных в модели доступа, в физической памяти и управления ими. Существуют три способа записи данных на физические носители: последовательный, списковый и прямой. В AutoCAD, например, используется списковый способ записи геометрических данных, что дает возможность пользователю хранить данные на физических носителях независимо от их логической последовательности.

3.2. Двухмерные графические системы

Двухмерные графические системы широко применяются при автоматизации чертежных работ. Чертежи любой сложности строятся из базовых графических элементов: точек, прямых, окружностей и других кривых. Каждый из этих элементов задается группой характерных точек, координаты которых могут определяться в абсолютной (мировой) системе координат или относительно предыдущей введенной точки (инкрементный ввод). При этом используют несколько способов задания точек:

– указанием на экране с помощью мыши;

– введением чисел с клавиатуры;

– «привязкой» к некоторому элементу чертежа, в окрестности которого располагается указатель.

Первый способ используют в основном для создания эскизов, а второй и третий – для построения точных изображений. Особенно удобным является третий способ, который позволяет «захватить» ближайший к курсору уже построенный элемент и ввести точные координаты конца или середины отрезка, центра окружности, точки пересечения прямых и т. д.

В средствах двухмерной графики обычно имеется несколько способов построения одного и того же элемента. Например, отрезок можно построить по двум точкам либо по начальной точке, длине и углу наклона, а окружность – по центру и радиусу, по трем точкам, по двум точкам и радиусу и т. д. Кроме того, в таких системах имеется ряд средств, автоматизирующих процесс черчения. Рассмотрим их подробнее.

Автоматическое построение скругления и фаски. Для получения скругления или фаски необходимо указать мышью на стороны угла и ввести значение радиуса скругления или размер фаски. Отрезки на стыке угла и скругления (фаски) автоматически «срезаются» (рис. 3.4).

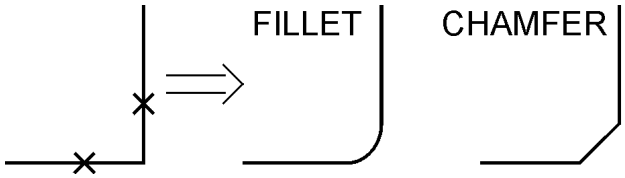


Рис. 3.4. Автоматическое построение скругления и фаски

Автоматическая штриховка и закраска. Для получения штриховки надо ввести угол и шаг штриховки, а затем указать мышью на внутреннюю часть области, которую надо заштриховать. Аналогично выполняется и закраска (рис. 3.5).

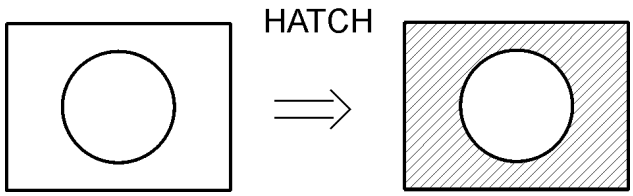


Рис. 3.5. Автоматическая штриховка

Автоматическая простановка размеров. Для простановки размера необходимо указать образмериваемые элементы и точку уровня размерной линии. После этого система автоматически вычислит числовое значение размера, выведет его на экране и нарисует выносные и размерные линии (рис. 3.6).

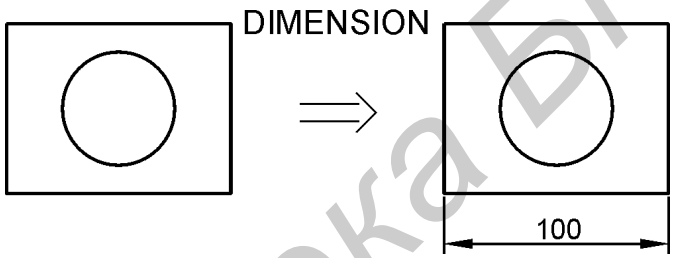


Рис. 3.6. Автоматическая простановка размеров

На любом этапе выполнения компьютерного чертежа можно удалить и модифицировать графические элементы изображения. Обычно двухмерные САПР позволяют выполнять следующие операции редактирования.

Отсечение. Эта процедура позволяет удалить лишние вспомогательные линии на чертеже. Например, после указания мышью на секущую прямую и окружность из чертежа удаляется сегмент окружности. Такая же процедура может быть выполнена для любой комбинации графических элементов (рис. 3.7).

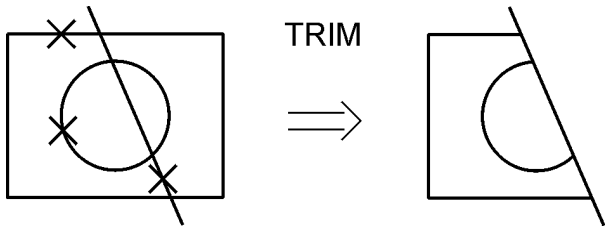


Рис. 3.7. Отсечение

«Резиновое» растяжение. Процедура растяжения позволяет пользователю растягивать или сжимать изображенные на экране формы. Можно, в частности, с помощью мыши переместить любую из вершин ломаной. При этом в процессе

редактирования перемещаемая вершина соединяется с соседними при помощи пунктирных («резиновых») линий, и сразу же после отпускания кнопки мыши пунктирные линии заменяются сплошными (рис. 3.8).

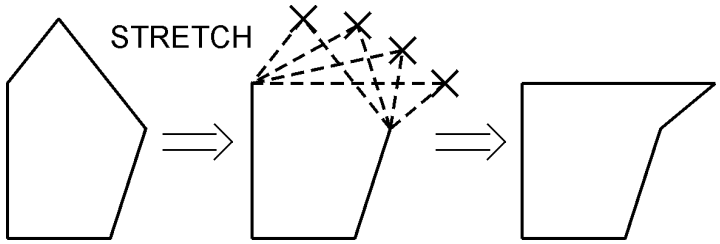


Рис. 3.8. «Резиновое» растяжение

Вспомогательная сетка. Для получения изображения с регулярной структурой удобно использовать сетку, которая не является частью чертежа и предназначена для визуальной координации. В таком режиме осуществляется автоматический «захват» ближайшего узла и от конструктора не требуется очень точного указания точки. В результате можно легко ввести точки с заданным шагом. На печать изображение сетки не выводится (рис. 3.9).

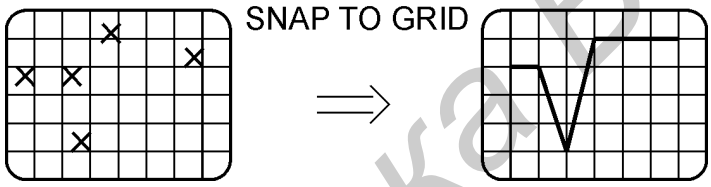


Рис. 3.9. Привязка к узлам сетки

Создание сплайнов. В большинстве двухмерных пакетов имеются средства для автоматической генерации гладких кривых (сплайнов), проходящих через заданные точки. Такие процедуры очень удобны при создании нестандартных геометрических форм и позволяют дополнить автоматизированное черчение элементами автоматизированного проектирования (рис. 3.10).

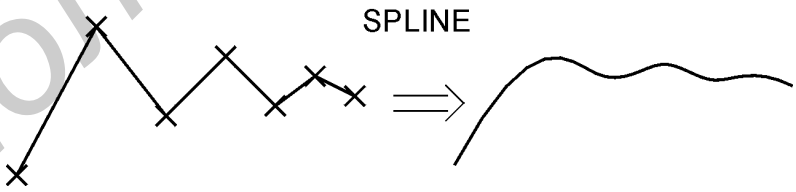


Рис. 3.10. Построение сплайна

Увеличение и панорамирование. Средства увеличения позволяют увеличить или уменьшить любую область чертежа для более детального просмотра или редактирования. Чтобы проанализировать другие элементы чертежа, область просмотра может быть панорамирована (сдвинута) в любом направлении, что позволяет использовать один сборочный чертеж для изделий любого размера и сложности, отказавшись от множества небольших чертежей,

как принято в традиционном черчении. При необходимости можно получить твердую копию любой области чертежа (рис. 3.11).

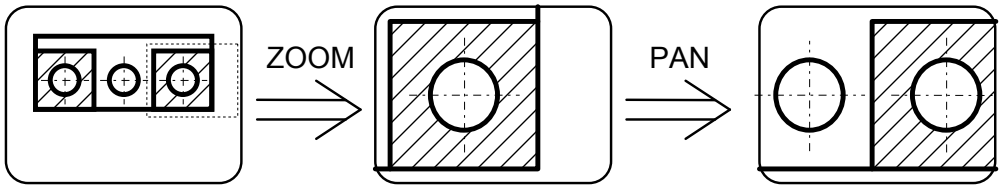


Рис. 3.11. Увеличение и панорамирование

Копирование, поворот и перенос. Любой элемент изображения или группу элементов можно скопировать, переместить и/или повернуть. Для двухмерного поворота достаточно задать положение центра вращения и угол поворота. При переносе обычно вводятся две точки, определяющие вектор смещения. Перенос и поворот, а также копирование широко применяются при вставке в чертеж стандартных элементов, вызываемых из стандартных библиотек (рис. 3.12).

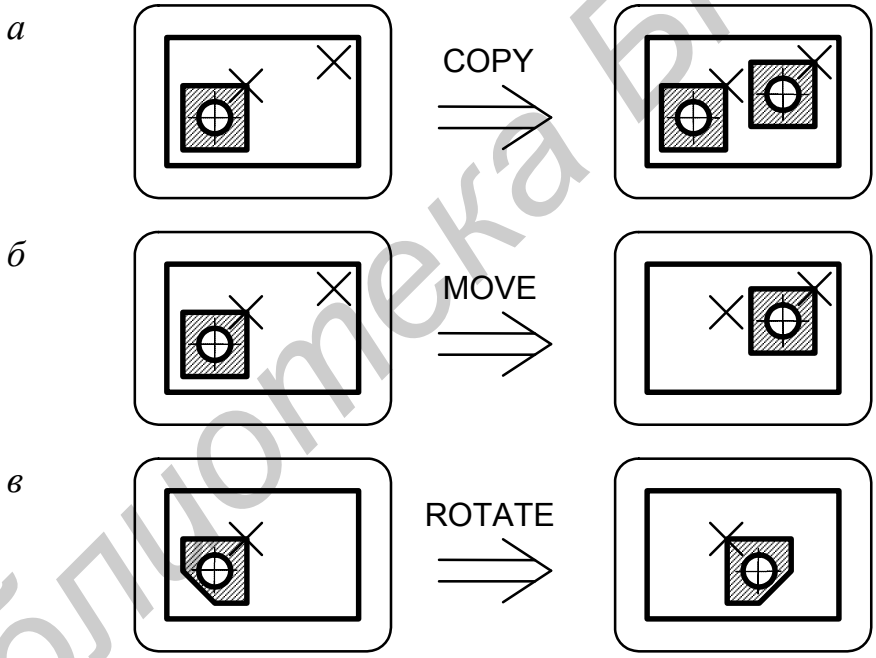


Рис. 3.12. Копирование (*а*), параллельный перенос (*б*) и поворот (*в*)

Преобразования. Кроме поворота и переноса многие графические пакеты имеют и более сложные средства манипулирования изображением: *зеркальное отражение* и *пошаговое размножение.* Используя эти средства, очень удобно строить чертежи регулярных структур. Например, для создания чертежа зубчатой рейки достаточно построить половину профиля одного из зубьев, далее применить операцию зеркального отражения, а затем размножить полученный образ. Можно также провести *масштабирование* изображения (рис. 3.13).

Выбор элементов и объединение их в группы. В большинстве графических пакетов операции редактирования могут выполняться как над

отдельным элементом, так и над группой элементов. Например, для сдвига части чертежа нет необходимости перемещать все элементы по отдельности. Вместо этого достаточно *«выбрать»* их путем указания на каждый из элементов либо осуществить *«окнирование»* (выделение на чертеже прямоугольной области, все элементы которой выбираются автоматически). В дальнейшем операции редактирования производятся одновременно над всеми элементами полученного набора. Можно, например, изменить цвет и тип линий всех элементов, произвести масштабирование и т. д.

Иногда удобнее объединить выбранные элементы в группы (блоки). В результате получается составной графический элемент, который при редактировании рассматривается как единое целое. В группы можно объединять и составные элементы, образуя многоуровневые иерархические структуры. При необходимости можно выполнить и обратную операцию – разделить составной элемент на отдельные составляющие.

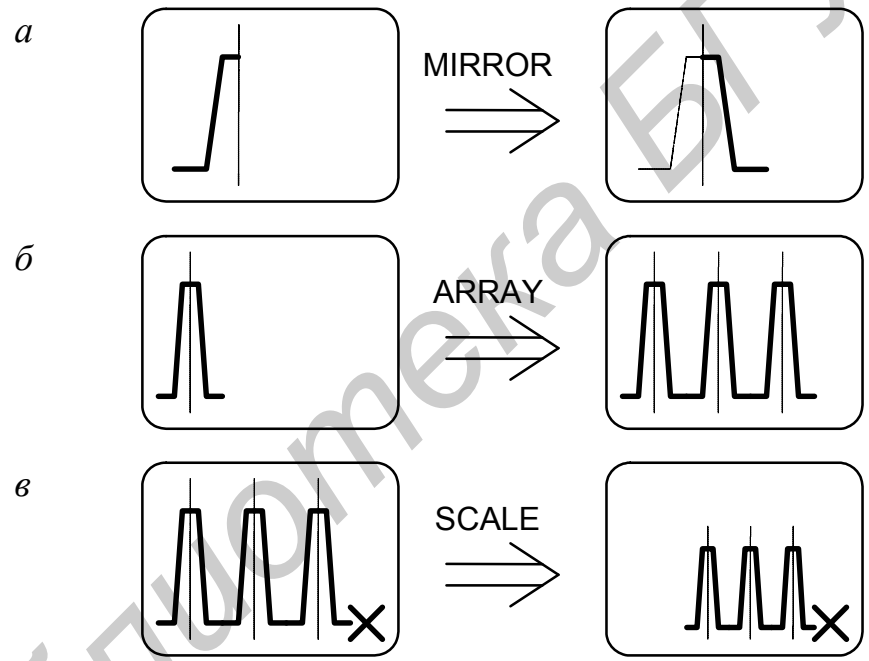


Рис. 3.13. Зеркальное отражение (*а*), размножение (*б*) и масштабирование (*в*)

Расслоение. Во многих двухмерных САПР реализован принцип «расслоения», позволяющий разделить чертеж на несколько частей, наложенных друг на друга. С точки зрения традиционного черчения это эквивалентно созданию нескольких чертежей, каждый из которых выполнен на прозрачной пластине. Причем можно рассматривать каждую пластину по отдельности либо, складывая их, получать совместное изображение. При выполнении чертежей механических конструкций можно, например, разместить все геометрические элементы в одном слое, а все размеры и пояснительные надписи – в другом.

Двухмерные системы позволяют построить упрощенные геометрические модели реальных физических объектов, состоящие из трех независимых

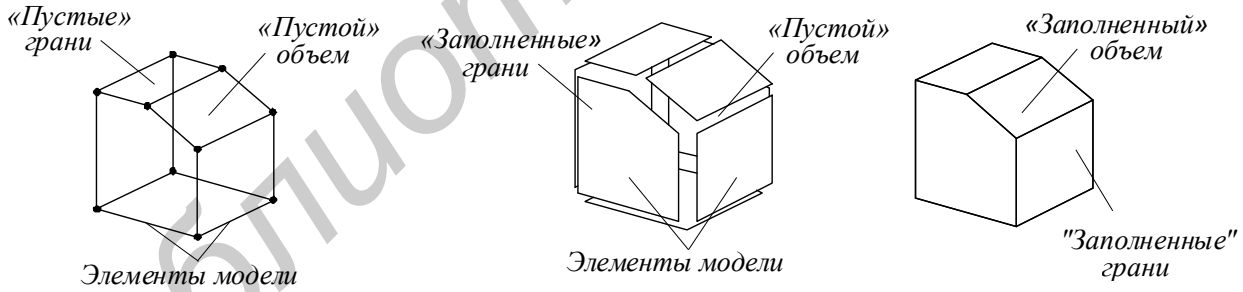
проекций (видов). При их использовании модель куба, например, задается 12 двухмерными точками с координатами XY, поэтому при внесении изменений конструктор должен редактировать отдельно каждую из проекций.

3.3. Трехмерные графические системы

В трехмерных системах используются точки с тремя координатами, что позволяет автоматически устанавливать проекционные связи. Так, в этом случае куб описывается восемью трехмерными точками XYZ, по которым находятся проекции XY, YZ и XZ. При использовании таких систем обычно начинают с построения трехмерного изображения, а двухмерные виды формируются на последнем этапе, при выводе чертежей. А в некоторых случаях двухмерные чертежи полностью заменяются трехмерной компьютерной моделью, по которой генерируются программы для станков с ЧПУ.

Системы трехмерного моделирования широко применяются в интегрированных САПР/АСТПП. Они часто дополняются средствами автоматического анализа физических характеристик (вычисление массы, центра масс, моментов и тензоров инерции и др.), а также модулями, обеспечивающими оценку прочности и технологичности. Использование трехмерных систем в настоящее время уже не сдерживается стоимостью программных средств и оборудования.

Методы трехмерного моделирования, используемые в САПР, делятся на три группы: каркасное, поверхностное и твердотельное (сплошное) моделирование (рис. 3.14).



а *б* *а*

Рис. 3.14. Геометрические модели трехмерных объектов:
а – каркасная (Wire-frame); *б* – поверхностная (Surface);
в – твердотельная (Solid)

Каркасное моделирование. Модель каркасного типа (рис. 3.14, *а*) полностью описывается в терминах точек и линий. Ее главными достоинствами являются простота и невысокие требования к компьютерной памяти, а недостатки связаны с отсутствием информации о гранях, заключенных между линиями, и с невозможностью различить внешнюю (незаполненную) и внутреннюю (заполненную) области. Наиболее широко каркасное

моделирование применяется при имитации несложного пространственного движения инструмента (например при фрезеровании по трем осям).

При использовании каркасных моделей в САПР необходимо учитывать следующие ограничения:

– *неоднозначность* – отсутствие возможности однозначно оценить ориентацию и видимость граней, что не позволяет различать виды сверху и снизу, а также автоматизировать удаление скрытых линий;

– *приближенное представление криволинейных граней* – невозможность точно описать криволинейные поверхности (цилиндры, конусы и др.), которые реально не имеют ребер; иногда для таких поверхностей вводят фиктивные ребра, располагаемые через регулярные интервалы (рис. 3.15);

– *невозможность обнаружить столкновения* – отсутствие информации о поверхностях, ограничивающих форму, не позволяет обнаружить столкновения между объектами, что важно при моделировании роботов, проектировании планов размещения оборудования и т. д.;

– *погрешности оценки физических характеристик* – возможность некорректного вычисления массы, центра тяжести, момента инерции и т. д., обусловленного недостатком информации об ограничивающих поверхностях;

– *отсутствие средств «затенения» поверхностей* – у модели, состоящей только из ребер, невозможно произвести закраску поверхностей различными цветами.

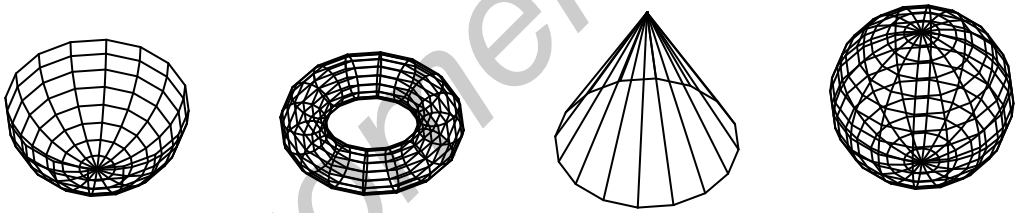


Рис. 3.15. Приближенное представление криволинейных поверхностей в каркасных моделях (вводятся фиктивные ребра)

Поверхностное моделирование. Модель поверхностного типа (рис. 3.14, *б*) задается в терминах точек, линий и поверхностей. В отличие от каркасной модели она обеспечивает:

– точное представление криволинейных граней;
– автоматическое распознавание граней и их закраску;
– автоматическое удаление невидимых линий (рис. 3.16);
– распознавание особых линий на гранях (отверстий и т. д.);
– обнаружение столкновений между объектами.

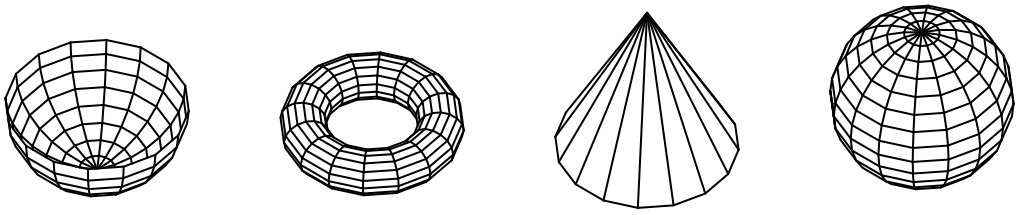


Рис. 3.16. Удаление невидимых линий при поверхностном моделировании

Метод поверхностного моделирования наиболее эффективен при проектировании и изготовлении сложных криволинейных поверхностей (корпусов автомобилей и др.). При этом можно использовать:

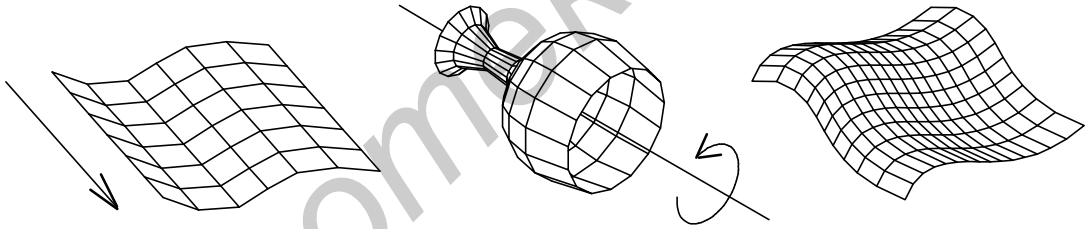
– *базовые геометрические поверхности* (плоскости, цилиндры, кубы, результат перемещения образующей кривой в заданном направлении и т. д. (рис. 3.17, *а*));

– *поверхности вращения* (результат вращения линии вокруг оси (рис. 3.17, *б*));

– *пересечения и сопряжения поверхностей;*

– *аналитические поверхности* (задаются математическим уравнением);

– *скульптурные поверхности или поверхности «свободных форм»,* которые не могут быть описаны одним математическим уравнением, а задаются при помощи методов сплайн-интерполяции образующих кривых (корпуса автомобилей, фюзеляжи самолетов, лопатки турбин) (рис. 3.17, *в*; 3.18)).



а *б* *в*

Рис. 3.17. Поверхностные модели, полученные путем перемещения кривой в заданном направлении (*а*), вращения кривой (*б*) и при помощи сплайн-интерполяции (*в*)

В современных трехмерных системах широко используются *составные поверхности,* составленные из криволинейных четырехугольников, ограниченных гладкими кривыми. Внутренняя область каждого такого участка определяется путем интерполяции. При изображении составных поверхностей на экране создается сетка, натянутая на многогранный каркас.

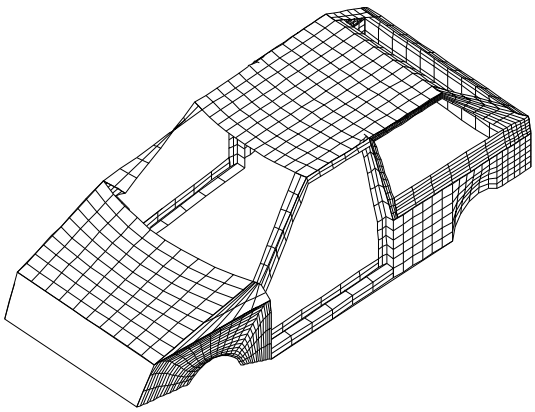


Рис. 3.18. Поверхностная модель кузова автомобиля

На базе методов поверхностного моделирования построен ряд мощных графических систем, широко применяемых в промышленности. Однако поверхностные модели имеют ряд недостатков, которые могут быть устранены только в рамках твердотельного моделирования. К ним относятся:

– *неоднозначность* при моделировании реальных твердых тел;

– *сложность* процедур удаления невидимых линий и отображения внутренних областей.

Твердотельное моделирование. Модель твердотельного типа (рис. 3.14, *в*) описывает трехмерный объем, который занимает рассматриваемое физическое тело. Внутреннее представление твердотельной геометрической модели выражается его границами (например гранями, ребрами и вершинами). В этом случае используются данные трех типов: *геометрические* (например, координаты вершин, описанные аналитически, ребра и поверхности, описанные уравнениями); *топологические,* позволяющие с помощью понятий «*внутри / вне»* определить топологию объекта; *вспомогательные* – например, цвет, степень прозрачности.

В отличие от каркасных и поверхностных моделей твердотельная модель обеспечивает:

– *полное описание заполненного объема* и возможность разграничения внешних и внутренних областей, на основе чего автоматизируется процесс обнаружения столкновений;

– автоматизацию процесса *удаления скрытых линий;*

– автоматизацию процесса *построения разрезов и сечений,* что требуется при создании сборочных чертежей сложных изделий;

– применение *современных методов анализа* конструкций (точное вычисление массы и габаритов, расчет прочности и деформаций методом конечных элементов и т. д.);

– эффективное *управление цветами и источником освещения,* получение тоновых изображений;

– более точное моделирование кинематики и динамики многозвенных механизмов (роботов, станков и т. д.).

Объемные тела, образованные из более простых объектов, называются *составными* графическими объектами. При этом используются логические операции объединения, пересечения, вычитания. Операция сборки составных

графических объектов осуществляется с использованием их представления в виде иерархической структуры в форме *дерева построения*.

Существует несколько методов построения и хранения данных о твердотельных моделях. Однако в САПР наибольшее распространение получили два из них: метод конструктивного представления (C-Rep) и метод граничного представления (B-Rep). Рассмотрим их подробнее.

Метод конструктивного представления (C-Rep) основан на создании моделей из типовых твердотельных примитивов с заданными размерами, ориентацией и точкой привязки. При определении взаимоотношений между соседними примитивами используются булевы операции: «объединение», «разность» и «пересечение». Твердотельные примитивы могут выбираться из библиотеки или генерироваться путем движения произвольной поверхности вдоль некоторой кривой. В результате происходит «захват» (sweeping) части трехмерного пространства, принадлежащей примитиву.

Метод граничного представления (B-Rep) также оперирует с примитивами, связанными при помощи булевых операций. При этом модель описывается совокупностью ребер и граней, определяющих граничную поверхность твердого тела. Эти данные дополняются информацией о топологии примитива и особенностях его геометрии. Метод B-Rep более удобен при модификации примитивов, но требует большего объема компьютерной памяти.

Методы твердотельного моделирования, основанные на булевых операциях, особенно удобны при вычислении поверхностных и весовых параметров тел, расчете напряжений, имитации операций механической обработки. В последнем случае операции резания металла (точение, фрезерование, сверление и т. п.) могут быть легко описаны при помощи булевой разности. Естественным приложением булевой алгебры является также анализ столкновений (коллизий), которые обнаруживаются при помощи операции пересечения.

4. Системы геометрического моделирования

4.1. Система DUCT

Концепция этой системы была разработана в конце 60-х гг. XX в. на инженерном факультете Кембриджского университета. Ее промышленный вариант распространяется с 1983 г. фирмой DELCAM (Delta Computer-Aided Manufacturing). Система DUCT позволяет конструктору создавать трехмерные каркасные и поверхностные модели объектов, визуализировать их на экране монитора, а также вычислять площадь поверхности, объем, координаты центра тяжести объекта и генерировать сетку конечных элементов для анализа потоков и расчетов на прочность.

При помощи системы DUCT автоматизированы процессы создания литейных форм и штампов, подготовки чертежей изделий и программ для станков с ЧПУ. При этом учитываются припуски на толщину стенок, усадки, зазоры при механической обработке.

Встроенный в DUCT каркасный редактор позволяет быстро и просто задать форму объекта. Поверхности создаются на основе поперечных сечений, определяемых набором точек. Эти сечения могут размещаться и ориентироваться относительно заранее построенной кривой (спина), играющей роль «хребта», вдоль которого вытянута поверхность. Для интерполяции поперечных сечений по заданным точкам используются кривые Безье.

При генерации поверхности, состоящей из отдельных элементов (лоскутов), имеется возможность задавать границы в виде сложных кривых. При этом автоматически обеспечивается гладкость в местах сопряжений лоскутов. Можно выполнить скругление угла, образованного двумя или тремя поверхностями, причем радиус скругления может быть как постоянным, так и переменным. Из набора поверхностей можно создать единый объект в виде оболочки, соединяющей в себе преимущества твердотельного и поверхностного моделирования. При визуализации объекта, используя несколько цветов и различные источники света, генерируется цветное изображение. Если позволяет производительность компьютера, формируются цветные динамические модели.

Ядро системы DUCT составляет реляционная база данных, в которой хранится вся информация о деталях и чертежах. При этом информация может располагаться в любой из объединенных в сеть рабочих станций. Имеется возможность создавать параметрические чертежи. Система также имеет библиотеку типовых деталей и интерфейсы с языком Си.

Графический интерфейс пользователя включает в себя набор многоуровневых меню, отображающих на экране списки параметров выбранной команды. Пользователь может также создавать свои собственные меню и прикладные программы. Для этого в системе имеется специальный командный язык.

Система DUCT предоставляет пользователю широкий набор методов механической обработки. При фрезеровании можно задать форму инструмента или выбрать из библиотеки различные фрезы: шаровую, торцевую, либо торцевую со скругленными кромками. При этом осуществляется проверка на минимальный радиус кривизны поверхности. Недоступные области высвечиваются на экране и затем могут быть обработаны инструментом меньшего радиуса. Для этого следует задать направление резания, скорость подачи и скорость вращения шпинделя, припуск для черновой обработки. Траектория движения инструмента генерируется, визуализируется на экране вместе с изображением инструмента и при необходимости корректируется. Система DUCT позволяет осуществлять 2,5-, 3- и 5-координатное

фрезерование, токарную обработку и 4-координатную электроэрозионную обработку.

DUCT может функционировать на стандартных рабочих станциях с операционной системой UNIX. Программное обеспечение построено по модульному принципу. К числу основных модулей относятся DUCTmodel, Advanced model, DUCTshade, DUCTnc, DUCTdraft и др. Для передачи данных используются стандарты IGES, VDA-FS и DXF. Имеются также прямые интерфейсы с системами PDGS и CADDS.

4.2. Система CATIA

Система CATIA, разработанная французской фирмой Dassault Systemes, появилась на мировом рынке САПР в 1981 г. Первоначально она предназначалась для самолетостроения, но постепенно нашла применение в автомобильной и других отраслях промышленности. В настоящее время система CATIA распространяется фирмой IBM и активно используется такими фирмами, как Boeing, Chrysler, ГАЗ и ВАЗ. Большинство пользователей системы CATIA использует рабочие станции IBM, но систему можно также установить на мэйнфреймах, UNIX-станциях RISC/6000 и рабочих станциях фирм Hewlett-Packard и Silicon Graphics.

Система имеет программные модули для автоматизированного проектирования в следующих областях:

- конструирование механизмов и узлов (Mechanical Design);
- формообразование поверхностей и дизайна (Shape Design and Styling);
- анализ и моделирование (Analysis and Simulation);
- подготовка производства (Manufacturing);
- инженерные расчеты (Equipment and System Engineering).

Система CATIA обладает одной из самых совершенных методик проектирования поверхностей (модули Surface Design, Advanced Surface Design, FreeForm Design) и анализа их качества по кривизне, гладкости, непрерывности в областях сопряжения и т. д. При необходимости поверхностные модели могут быть преобразованы в твердотельные. Сложные кривые строятся на базе кривых Безье. Поверхности, которые нельзя описать стандартными типами элементов, могут быть аппроксимированы точками с заданными координатами, по которым строятся бипараметрические полиномиальные функции.

В рамках системы CATIA можно использовать целый ряд приложений, входящих в «архитектуру прикладных программ» (CAA – CATIA Application Architecture). Они позволяют выполнить динамический анализ механизмов (CATDADS), моделирование роботизированных комплексов (Robuse), подготовить постпроцессоры и программы для станков с ЧПУ, создать проекты трубопроводов и пневмосистем (3D-Tubing), а также создать фотореалистические изображения (Visualisation Studio). В результате

сотрудничества IBM с фирмой BMW создан уникальный модуль проверки собираемости изделия (Fitting Simulation), который моделирует весь процесс сборки изделия с учетом доступности, способов установки и используемой оснастки. Этот же модуль позволяет оценить возможность съёма узла для ремонта или обслуживания.

К числу существенных достоинств системы CATIA относятся совершенная схема построения параметрически связанных моделей и возможность постпараметризации. Конструктор может начать построение модели, руководствуясь лишь функциональностью и технологичностью, не занимаясь на первом этапе параметризацией. Далее, на любом этапе, можно определить параметры и связи между ними, при этом допускается задать лишь наиболее существенные параметры, а остальные оставить по умолчанию. При изменении параметров конструктор может позволить ассоциативно распространить изменения на все объекты либо ограничить изменения некоторой зоной (т. е. определить некоторые параметры как локальные). На любом этапе можно принять решение о разрыве связи. Система CATIA также позволяет рассчитать трехмерные допуски (Functional DIMENSIONING and 3D-Tolerancing), размерные цепи любой сложности и оптимальные допуски, промоделировать поведение механизмов при варьировании размеров деталей.

Новым нетрадиционным средством системы CATIA, не характерным для других САПР, является «проектирование при помощи правил», реализующее по существу, концепцию конструктивно-технологической базы знаний. Оно позволяет конструктору определить свои собственные правила проектирования и использовать их в повседневной работе. Например, при проектировании гидро- и пневмосистем можно определить шаг расстановки крепежа, допустимые и запретные зоны расположения, зоны безопасного расстояния от элементов конструкции, материалы и т. д.

При помощи системы CATIA можно автоматизировать и некоторые процессы управления предприятием (финансами, складским хозяйством, кадрами и т. д.). Для этого используются коммерческие сетевые реляционные базы данных (Oracle, DB2) и интерфейсы с наиболее распространенными системами управления предприятиями (R3, Triton).

4.3. Системы фирмы Autodesk

Наиболее распространенным и известным программным продуктом фирмы Autodesk является пакет AutoCAD, первые версии которого были ориентированы на двумерное черчение и выпуск конструкторской документации. В процессе совершенствования этот пакет превратился в развитую среду трехмерного моделирования проектов в архитектуре, строительстве, машиностроении, картографии и других отраслях. Последние версии пакета AutoCAD могут выполняться как на ПК под управлением Windows, так и на UNIX-платформах (Silicon Graphics, Hewlett-Packard, Sun

SPARC-Station, IBM R/6000). Ядро системы написано на языке C++ и представляет собой объектно-ориентированную среду, являющуюся основой для множества прикладных программ, создаваемых как самой фирмой Autodesk, так и ее партнерами. Их число уже превысило 5000.

Для машиностроения фирмой Autodesk разработан интегрированный пакет Autodesk Mechanical Desktop (AMD), в который входят практически все необходимые инженеру-конструктору средства моделирования геометрических объектов. Он объединяет в себе возможности известных программных продуктов Autodesk:

– AutoCAD – в качестве графической среды;

– AutoCAD Designer – для конструирования деталей и сборочных узлов;

– AutoSurf – для моделирования сложных NURBS-поверхностей;

– IGES Translator – для обмена файлами с другими системами САПР;

– MCAD – система меню для организации взаимодействия с другими машиностроительными приложениями.

Модуль AutoCAD Designer позволяет автоматизировать процесс создания рабочих чертежей деталей и сборочных единиц. Он дает возможность пользователю оперировать с привычными конструкторско-технологическими элементами (сопряжение, фаска, отверстие и т. д.). В нем реализован принцип параметрического моделирования, что позволяет довольно гибко вносить изменения на любой стадии проектирования.

Процесс создания трехмерных моделей в AutoCAD Designer происходит в два этапа: сначала задается плоский эскиз детали, а затем ему придается третье измерение. При конструировании сборочной единицы пользователю достаточно задать параметрические связи между существующими объектами, ограничивая число степеней свободы проектируемой механической системы. Для разработанных моделей автоматически генерируются двухмерные проекции (виды), причем постоянно действует двунаправленная параметрическая связь модель-чертеж. Возможности параметрического черчения позволяют вносить изменения как на самой трехмерной модели, так и на ее двумерных видах путем корректировки отдельных размеров. В модуле также имеются встроенные функции, обнаруживающие взаимное пересечение деталей в сборочных единицах.

Модуль AutoSurf обеспечивает построение и редактирование формообразующих кривых и сложных поверхностей, построение на их основе новых объектов, создание каркасных (проволочных) моделей, проецирование различных контуров на плоскость или поверхность. В этом модуле реализована NURBS-технология, основанная на наиболее эффективных методах работы с произвольными поверхностями, которые описываются неоднородными рациональными B-сплайнами. Поэтому AutoSurf получил широкое распространение в автомобильной и аэрокосмической промышленности.

IGES-транслятор используется для корректного обмена графическими моделями с другими системами CAD/CAM/CAE. Например, dwg-файлы формата AutoCAD после обработки в AutoSurf могут передаваться в другие системы, осуществляющие расчеты прочности, генерации программ для ЧПУ и т. д.

Как показывает опыт, наибольший эффект при проектировании машиностроительных конструкций может быть достигнут только при совместном использовании модулей AutoCAD Designer и AutoSurf, когда некоторые трудности моделирования поверхностей произвольной формы в AutoCAD Designer устраняются средствами AutoSurf, а проблемы построения проекционных видов в AutoSurf решаются за счет средств AutoCAD Designer. Объединение этих двух программ не только обеспечивает двусторонний обмен данными на основе формата AutoCAD (dwg-файлы), но и позволяет по-новому организовать доступ к основным функциям AMD. В этом пакете выделены четыре основных функциональных модуля:

1) параметрического твердотельного моделирования;

2) параметрического моделирования сборочных единиц;

3) моделирования поверхностей произвольной формы;

4) генерирования двухмерных чертежей.

Первые два модуля являются составными частями программы AutoCAD Designer, третий модуль включает основные функции AutoSurf, а четвертый представляет собой универсальный инструмент, применимый как для стандартных трехмерных объектов AutoCAD, так и для комбинаций разнородных трехмерных объектов.

При создании плоского эскиза изделия используются стандартные для AutoCAD команды рисования и редактирования двухмерных объектов. Затем устанавливаются зависимости на горизонтальность, вертикальность, параллельность, перпендикулярность, коллинеарность, концентричность, проекции, касания, равенство радиусов и координаты X и Y. При этом от конструктора не требуется соблюдения большой точности в отношении размеров, параллельности или перпендикулярности. Поэтому в отличие от стандартных приемов AutoCAD здесь не используются режимы «шаг», «сетка», а также функции объектной привязки. Эскиз создается концептуально.

Далее выполняется профилирование эскиза, в процессе которого на модель накладываются геометрические связи и почти горизонтальные линии становятся строго горизонтальными, почти замкнутые – замкнутыми и т. д. При этом программа выдает сообщение о том, сколько связей или размеров требуется, чтобы однозначно определить профиль.

Для простановки параметрических размеров используется единая команда, которая в зависимости от заданных примитивов автоматически определяет тип размера (линейный, угловой, радиальный и т. д.). Этим размерам соответствуют свои переменные, которые могут использоваться в математических формулах,

что обеспечивает формирование изображения детали при помощи таблицы. Если требуется указать размер, зависящий от размера других деталей, используются глобальные переменные.

Геометрия эскиза может быть любой сложности, но в нем должен быть один замкнутый контур. Он используется в дальнейшем при построении третьего измерения или создании так называемой базовой формы одним из следующих способов: выдавливание, вращение, перемещение вдоль двухмерной криволинейной направляющей, сдвиг плоского эскизного контура, а также путем отсечения фрагментов от твердотельных объектов произвольными поверхностями. Далее к базовой форме добавляются стандартные конструкторско-технологические элементы: отверстия (в том числе с зенковкой, разверткой и резьбовые), фаски, сопряжения (галтели) или элементы произвольной формы.

Следует учитывать, что базовая форма представляет собой твердое тело и формообразование в AutoCAD Designer осуществляется при помощи булевых операций над пространственными множествами (объединение, вычитание и пересечение). Например, добавление отверстия к модели сводится к вычитанию объема, а при задании фасок может выполняться как вычитание, так и объединение.

AutoCAD Designer реализует функцию параметризации как на уровне отдельной модели, так и сборочной единицы. Поэтому процесс сборки почти полностью автоматизирован: пользователь должен только задать связи, ограничивающие число степеней подвижности, а программа сама генерирует сборочные чертежи и спецификации. При этом сборочная единица представляется в виде иерархической структуры, в которой строго заданы как взаимосвязи отдельных компонент, так и последовательность процесса сборки.

Процесс моделирования сборочных единиц в AutoCAD Designer состоит из следующих этапов:

- определение компонентов сборочной единицы;
- вставка компонентов в сборочную единицу;
- наложение и редактирование связей между компонентами;
- сборка компонентов и анализ сборочной единицы;
- создание сборочного чертежа.

Чтобы начать процесс сборки, все компоненты необходимо явно определить как доступные для сборки («материализовать»). Вставка компонентов в рабочее пространство AMD подобна вставке блоков в AutoCAD, причем один и тот же компонент (деталь) может быть использован неоднократно. Детали и подузлы представлены в виде иерархической структуры сборочного узла. Конструктор должен строго придерживаться определенной последовательности сборки, вводя сначала «базовые» компоненты, а затем «присоединяемые» к ним. При этом не требуется

внимательно следить за взаимной ориентацией и положением собираемых деталей, так как в дальнейшем они скорректируются автоматически за счет введения параметрических связей. Возможно свободно-координатное расположение деталей, которое задается относительно друг друга по их ребрам, осям или граням, а степени свободы компонентов отображаются графически.

Окончательная сборка изделия осуществляется путем определения связей, определяющих взаимное расположение компонентов:

– *Insert* (вставить соосно) – задаются цилиндрические поверхности, которые получат общую ось, расстояние между ними и направление соединения;

– *Mate* (встык) – задаются сопрягаемые поверхности, линии или точки двух компонентов, а также расстояние между ними (отступ);

– *Flush* (заподлицо) – задаются грани двух компонентов, нормали к которым ориентируются параллельно в одном направлении;

– *Angle* (угловая зависимость) – нормали указанных граней пары компонентов ориентируются под заданным углом;

– *Oppose* (под углом в противоположных направлениях) – нормали указанных граней пары компонентов ориентируются в противоположных направлениях под заданным углом.

После задания типа связи необходимо указать компоненты, к которым применяется заданная связь; затем компоненты перестраиваются на экране автоматически с учетом введенных связей, имитируя процесс сборки.

После сборки изделия производится расчет массоинерционных характеристик, площади поверхности, массы и объема деталей и сборочных узлов, моментов инерции и анализ взаимодействия деталей в сборочных узлах.

При генерации сборочных чертежей пользователь может задать любую совокупность проекционных видов и разрезов сборочной единицы, позволяющих уяснить взаимное расположение компонентов. При этом устанавливается двунаправленная ассоциативная связь между моделью и ее чертежом, автоматически удаляются штриховые и невидимые линии, наносятся размеры и выноски. Кроме того, возможно построение изометрических проекций в так называемом разобранном виде (exploded view), что удобно при подготовке руководств по сборке и эксплуатации. В AMD имеются также средства простановки номеров позиций на сборочных чертежах и автоматического заполнения спецификаций, форма которых определяется пользователем. Спецификация, соответствующая стандартам ANSI, ISO, DIN, JIS и ЕСКД, может размещаться в поле чертежа или выводиться во внешний файл.

Средствами AutoSurf осуществляется моделирование как примитивных поверхностей (конус, шар, цилиндр), так и сложных поверхностей произвольной формы (трубчатых, поверхностей натяжения, изгиба, перехода), а также плавное сопряжение произвольных поверхностей. Во внутреннем

формате AutoSurf контуры и оболочки точно описываются математическими уравнениями, однако при выводе на экран поверхности преобразуются в каркасы, что существенно сокращает время регенерации изображения. Кроме того, в AutoSurf каркасы используются как исходные данные при построении поверхностей сложной формы. Имеются функции вычисления площади поверхности и объема. Всего в AutoSurf существуют четыре типа поверхностей:

1) элементарные;

2) вращения (получаемые перемещением элементов каркаса);

3) движения (получаемые натяжением оболочки на каркас);

4) производные (получаемые из существующих).

Для их построения можно использовать 16 различных способов, применяя разные типы каркасных элементов: сплайны, полилинии, дуги, окружности, эллипсы, линии с векторами вращения. Но несмотря на разнообразие методов построения, внутренний формат представления всех без исключения поверхностей основан на одном и том же математическом аппарате – аппарате неоднородных рациональных B-сплайнов (NURBS). Поверхности AutoSurf могут быть представлены на экране либо в тонированном виде, либо в виде каркаса, который является лишь вспомогательным средством отображения. Для тонирования трехмерных моделей и создания фотореалистических изображений в стандартную поставку пакета AutoCAD включен модуль AutoVision.

4.4. Система I-DEAS

Торговая марка I-DEAS объединяет целый ряд программных продуктов, которые можно рассматривать как интегрированный комплекс CAD/CAM/CAE. Эти программные средства созданы фирмой SDRC (Structural Dynamics Research Corporation) и предназначены для автоматизации разработки металлических конструкций в аэрокосмической, автомобильной и других отраслях промышленности. Отличительными особенностями I-DEAS являются возможность распределения пользовательских лицензий по неоднородным локальным сетям, которые содержат рабочие станции разных производителей (HP, IBM, SUN и др.), а также наличие встроенных средств конечно-элементного моделирования, оптимизации и автоматизации испытаний.

Типовой набор модулей I-DEAS для решения задач проектирования металлоконструкций (Product Design Package) включает в себя следующие модули:

– I-DEAS Master Modeler – базовый модуль трехмерного моделирования (проволочного, поверхностного, твердотельного);

– I-DEAS Master Surfacing – модуль трехмерного моделирования деталей со сложными «скульптурными» поверхностями;

– I-DEAS Master Assembly – модуль для трехмерного моделирования сборочных узлов и простейших механизмов;

– I-DEAS Drafting – модуль создания чертежей изделия;

– I-DEAS Data Translation – модуль преобразования форматов графических данных.

При помощи модуля I-DEAS Master Modeler создается твердотельная геометрическая модель, которая используется в качестве исходной в большинстве других задач (при прочностном анализе, черчении, подготовке программ для ЧПУ и т. д.). В нем реализована NURBS-геометрия, а также «параллельная ассоциативность», поддерживающая групповую работу нескольких конструкторов. Кроме того, «история» процесса проектирования запоминается в виде дерева, любую ветвь которого можно редактировать.

Модуль I-DEAS Master Surfacing позволяет создавать модели деталей со сложными «скульптурными» поверхностями. При этом предполагается, что первичный эскиз подготовлен при помощи модуля I-DEAS Master Modeler, в котором реализована NURBS-геометрия с двойной точностью, а также моделирование при помощи кривых Безье высокого порядка. Для контроля гладкости полученной поверхности используются специальные средства визуализации, выявляющие участки с резкими изломами.

Модуль I-DEAS Master Assembly позволяет создавать трехмерные модели сборочных узлов, состоящих из большого числа мелких элементов, каждый из которых также может быть сборочным узлом. При вставке элемента в механизм конструктору не требуется точного позиционирования, достаточно лишь задать связи между элементами, по которым далее находится точное взаимное расположение деталей. Для созданной сборки можно автоматически сформировать спецификацию, а для группы сборок – построить таблицы входимости, содержащие списки сборочных узлов, в которые входят заданные детали. Кроме того, имеются функции анализа собираемости созданного узла и оценки допусков при помощи I-DEAS Tolerance Analysis. Модель простейшего механизма также можно построить с помощью модуля Master Assembly, но при создании моделей сложных пространственных механизмов применяется специальный модуль I-DEAS Mechanism Design, который также позволяет провести полный динамический анализ (вычислить силы, моменты, перемещения, скорости, ускорения и т. д.).

Модуль I-DEAS Drafting предназначен для получения чертежей изделия, созданного в модулях I-DEAS Master Modeler, I-DEAS Master Surfacing, I-DEAS Master Assembly, либо применяется как самостоятельная система двухмерного черчения. Для взаимодействия с пользователем I-DEAS Drafting использует «Динамический навигатор» – специальное средство, поддерживающее как однопользовательскую, так и многопользовательскую работу. По геометрии главной модели (мастер-модели) легко создаются проекции, сечения, разрезы, проставляются размеры. Чертежи взаимосвязаны с главной моделью в обоих

направлениях, т. е. изменение любого размера на чертеже сразу же отражается в мастер-модели и наоборот. В процессе черчения также автоматически создается спецификация, которая обновляется динамически при изменении числа объектов, их атрибутов и т. д.

I-DEAS Data Translation представляет собой набор модулей для чтения и записи графических файлов в форматах IGES, STEP, VDA-FS, DXF, а также для обмена данными с пакетами Pro/ENGINEER, CATIA, CAD AM, CADDS и другими.

4.5. Система Unigraphics

Система Unigraphics фирмы EDS широко распространена в аэрокосмической и автомобильной промышленности, а также в машиностроении. Ее отличительными особенностями являются наличие средств гибридного трехмерного моделирования, ассоциативной базы данных, развитых средств моделирования сборочных узлов и создания чертежей. В состав системы Unigraphics входит несколько десятков модулей, основными из которых являются:

– UG/Gateway – поддерживает интерфейс системы с пользователем и взаимодействие между ее отдельными компонентами;

– UG/Solid Modeling – базовый модуль трехмерного гибридного моделирования (проволочного, поверхностного, твердотельного и их модификаций);

– UG/Features Modeling – позволяет редактировать и параметрически задавать стандартные элементы изделий, такие, как отверстия, щели, выступы, прокладки, стержни, трубы, желоба и т. д.;

– UG/Freeform Modeling – модуль трехмерного моделирования сложных «скульптурных» поверхностей;

– UG/User-Defmed Features – позволяет представлять произвольные группы деталей в виде одного параметризованного стандартного объекта, который может использоваться всеми конструкторами;

– UG/Drafting – модуль автоматизированного черчения, поддерживающий все основные промышленные стандарты (ANSI, ISO, DIN, JIS) и включающий в себя средства формирования ортогональных и изометрических проекций, разрезов, сечений и т. д.;

– UG/Assembly Modeling – позволяет создавать ассоциативные параметрические модели сложных сборочных узлов в режиме групповой работы конструкторов;

– UG/Mechanisms – модуль проектирования и моделирования двухмерных и трехмерных механических систем непосредственно в среде пакета Unigraphics, позволяющий осуществить полный кинематический анализ, оценить зазоры между элементами, выявить столкновения, вычислить силы, моменты и т. д.

Кроме этого, в состав Unigraphics входит целый ряд модулей для подготовки автоматизированного производства, базовыми из которых являются UG/CAM Base и UG/Postprocessor. Имеются также модули, ориентированые на конкретные технологические процессы: UG/MF-Flowcheck – для литья; UG/Lathe – для токарных работ; UG/Planar Milling, UG/Fixed-Axis Milling, UG/Variable-Axis Milling – для фрезерования; UG/Sheet Metal Design, UG/Sheet Metal Fabrication, UG/Sheet Metal Nesting – для изготовления изделий из листового металла. Используя средства пакета Unigraphics, можно создавать фотореалистические изображения разрабатываемых изделий (UG/Photo) и их трехмерные прототипы (UG/Rapid Prototyping).

5. Системы координат

Для вывода изображения объекта на экран графического устройства необходимо решить две основные задачи:

– указать положение всех точек объекта в пространстве;
– определить положение их образов на мониторе.

Для задания положения точек в пространстве и на мониторе используются системы координат. Необходимо знать, как связаны между собой различные системы координат, особенно это важно для проектирования трёхмерного объекта на плоский экран. Проекция на экране строится по тем же законам, что и проекция реального объекта на сетчатке человеческого глаза.

Первой среди систем координат рассмотрим *систему координат устройства* (*device coordinate system*), определяющую положение точки на экране. Она состоит из горизонтальной оси *u* и вертикальной оси *v* (рис. 5.1). Начало отсчёта выбирается произвольно. Осей *u* и *v* достаточно для задания положения любой точки экрана, поэтому третья ось, перпендикулярная первым двум, не определяется. Положение любой точки задаётся двумя целыми числами *u* и *v*, равными числу пикселов между началом координат и точкой по осям *u* и *v*. Однако одна и та же точка может задаваться разными парами *u* и *v* в зависимости от положения начала координат, направления осей и масштаба. Эти параметры для разных графических устройств устанавливаются произвольно (рис. 5.1).

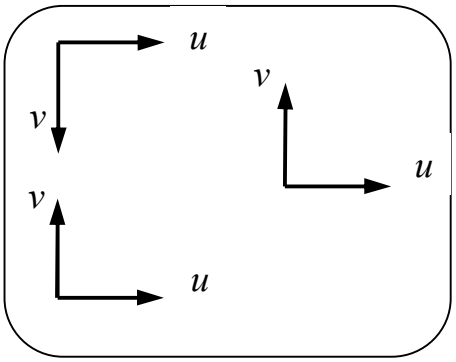


Рис. 5.1. Системы координат устройства

Виртуальная система координат устройства (virtual device coordinate system) позволяет избежать описанной выше проблемы. Она фиксирует точку отсчета, направление и масштаб осей для всех рабочих станций. «Виртуальный» означает, что данная система отсчёта существует только в воображении программиста. Обычно начало отсчета располагается в левом нижнем углу экрана, ось *u* откладывается вправо, а ось *v* – вверх. Точка в виртуальной системе координат на любом экране попадает в одно и то же место. Графическая программа передает виртуальные координаты подпрограмме драйвера устройства, которая преобразует их в координаты конкретного устройства.

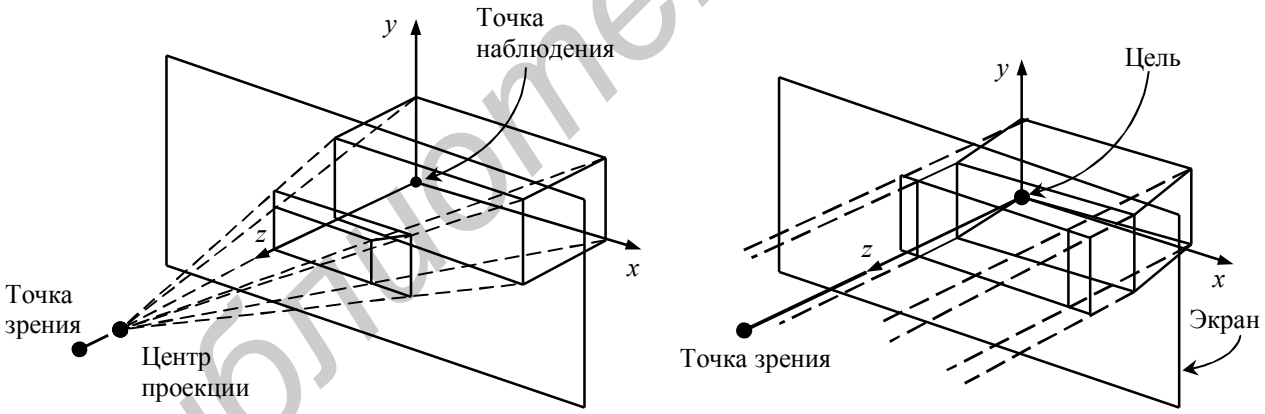
Виртуальная и обычная системы координат устройства позволяют задавать положение точки на плоском экране. Займемся теперь системами координат для работы с трехмерным пространством. Основных трехмерных систем координат всего три: *внешняя система координат (world coordinate system,*

WCS), система координат модели (model coordinate system) и *система координат наблюдателя (viewing coordinate system).*

WCS – это внешняя по отношению к объектам мира, опорная система координат. Такая система может использоваться для описания расположения и ориентации парт, стульев и доски, если интересующий нас мир представляет собой класс.

Следующий шаг – описание формы объекта. Она определяется координатами его характеристических точек по отношению к системе координат, связанной с ним, – *системой координат модели* (*model coordinate system*). Координаты точек объекта, определённые таким образом, не изменяются, когда объект перемещается или вращается в пространстве. Система координат модели перемещается вместе с тем объектом, к которому она привязана. Расположение и ориентация любого объекта задаются относительным положением и ориентацией модельной системы координат данного объекта по отношению к внешней системе координат. Относительное расположение и ориентация систем координат определяются матрицей преобразования, которая позволяет получить координаты любой точки любого объекта во внешней системе.

Следующий шаг – проецирование трехмерных объектов или их точек на монитор подобно тому, как они проецируются на сетчатку человеческого глаза. В компьютерной графике используются два вида проекций: перспективная и параллельная (рис. 5.2).



а *б*

Рис. 5.2. Два вида проекций: *а* – перспективная; *б* – параллельная

Оба вида требуют задания двух точек: точки зрения и точки наблюдения. *Точка зрения (viewpoint)* – это глаз наблюдателя. *Точка наблюдения* (*viewsite*) – это точка объекта, определяющая направление «луча зрения». Вектор, проведенный от точки зрения к цели, задает направление наблюдения.

В *перспективной проекции (perspective projection)* все точки объекта соединяются с центром проекции, лежащем на линии, соединяющей точку зрения и цель (в противном случае проекция называется косоугольной). Точки

пересечения этих линий с экраном образуют проекцию. Экран располагается между точкой зрения и целью. В *параллельной проекции (parallel projection)* линии от всех точек объекта проводятся в направлении наблюдателя параллельно направлению наблюдения, а точки пересечения этих линий с экраном формируют проекцию. Экран, как и в перспективной проекции, располагается перпендикулярно направлению проектирования. Такая проекция называется ортогональной.

В AutoCAD существуют две системы координат: мировая система координат World Coordinate System (WCS) и пользовательская система координат User Coordinate System (UCS). Ось *x* мировой системы координат направлена горизонтально, *y* – вертикально, а ось *z* проходит перпендикулярно плоскости *xy*. Началом координат является точка пересечения осей *x* и *y*. Первоначально она совмещается с левым нижним углом чертежа. В любой момент активна только одна система координат, которую принято называть текущей. Она управляется командой UCS, в ответ на которую выдается список возможных опций:

Origin/ZAxis/3point/OBject/View/X/Y/Z/Prev/Restore/Save/Del/?/ <World>:

Рассмотрим основные из них:

– World UCS – переход в мировую систему координат.

– Object UCS – выравнивание системы координат по существующему объекту.

– View UCS – выравнивание системы координат в направлении текущего вида, то есть определение новой системы координат с плоскостью XY, расположенной перпендикулярно направлению вида, иначе говоря, параллельно экрану.

– Origin UCS – смещение начала координат.

– 3Point UCS – определение положения системы координат по трем точкам.

– XAxis Rotate UCS – поворот системы координат вокруг оси X.

– YAxis Rotate UCS – поворот системы координат вокруг оси Y.

– ZAxis Rotate UCS – поворот системы координат вокруг оси Z.

В САПР AutoCAD ввод координат с клавиатуры возможен в абсолютных или относительных координатах. Ввод **абсолютных координат** производится в следующих форматах:

– *декартовы (прямоугольные) координаты*. Ввод координат заключается в указании расстояния от начала координат до точки по каждой из осей *x*, *y*, *z*, а также направления (положительного или отрицательного).

– *полярные координаты*. Задаются в формате $r < A$, где *r* – радиус, *A* – угол от предыдущей точки. Угол задается в градусах против часовой стрелки. Значение 0 соответствует положительному направлению оси OX.

Относительные координаты задают смещение от последней введенной точки. При вводе точек в относительных координатах можно использовать любой формат записи в абсолютных координатах: @dx,dy – для декартовых, *@r < A* – для полярных.

Пример построения в абсолютных координатах:

From point: 40,20 – из точки 1

To point: 190,20 – в точку 2

Пример построения в относительных координатах:

From point: 40,20 – из точки 1

To point: @150,0 – в точку 2

Пример построения в полярных координатах:

From point: 40,20 – из точки 1

To point: @150<0 – в точку 2

6. Матрицы преобразования

Проецирование точек на объект в трёхмерном пространстве требует преобразования координат из одной системы в другую [2]. При этом сначала необходимо перевести координаты точек объекта из модельной системы в мировую. Текущее положение объекта обычно задаётся через повороты и смещения относительно исходного положения, в котором модельная система координат совпадала с мировой. Следовательно, мировые координаты точек объекта можно получить трансляцией и поворотом соответствующих точек из их исходного положения, в котором их модельные координаты совпадали с мировыми.

Большинство графических библиотек выполняют эти преобразования самостоятельно, а программисту остаётся задать только смещение и поворот для интересующего его объекта. Однако проектировщику всё равно нужно знать законы преобразований, чтобы рисовать объекты в нужных местах без проб и ошибок, в особенности если эти объекты перемещаются достаточно сложным образом.

Получив мировые координаты всех точек объекта в его текущем положении, необходимо вычислить координаты этих точек в наблюдательской системе. Перевод координат из одной системы в другую называется *отображением (mapping).*

Отображение между мировой и наблюдательской системами координат обычно также осуществляется графической библиотекой самостоятельно по заданным координатам точки зрения, точки наблюдения и направлению вектора вертикали (в мировых координатах).

6.1. Трансляция

При трансляции объекта на величины a, b и с в направлениях X, Y и Z соответственно по отношению в начальному положению, в котором модельная система координат совпадала с мировой (рис. 6.1), мировые координаты точек объекта в новом положении (X_w, Y_w, Z_w) вычисляются следующим образом:

$$\begin{aligned} X_w &= X_m + a; \\ Y_w &= Y_m + b; \\ Z_w &= Z_m + c. \end{aligned} \qquad (6.1)$$

В этой формуле числа X_m, Y_m, Z_m являются также модельными координатами точки. В матричной форме

$$\begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_m \\ Y_m \\ Z_m \\ 1 \end{bmatrix} \qquad (6.2)$$

$$\Downarrow$$

$$Trans\,(a, b, c)$$

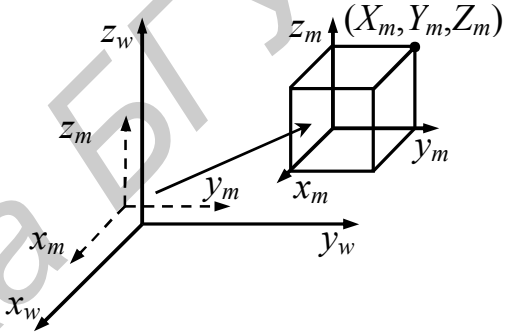


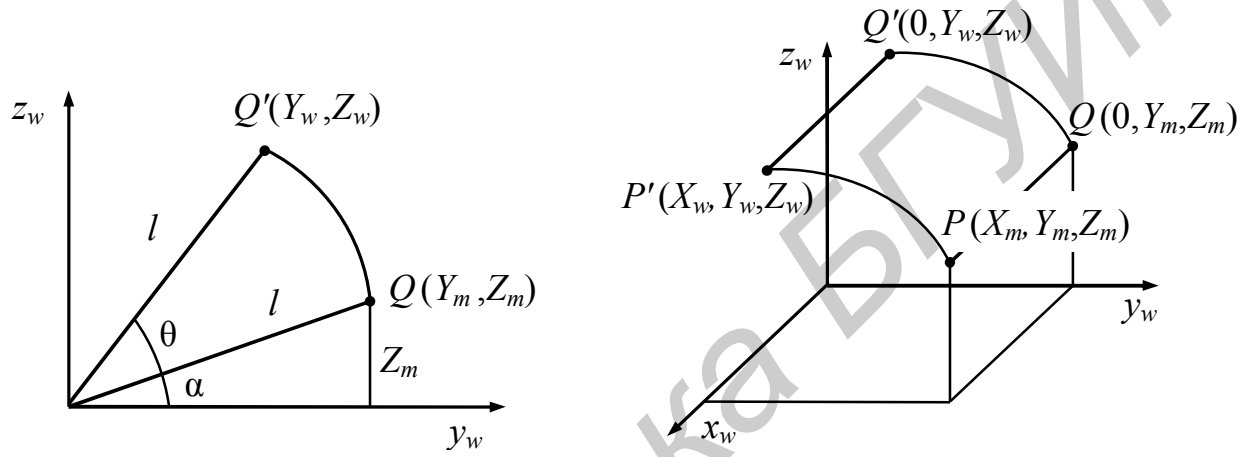
Рис. 6.1. Трансляция объекта

Формулы (6.2) и (6.1) эквивалентны друг другу: для этого достаточно записать (6.2) в развёрнутом виде. Операцию сложения в (6.1) удалось записать через умножение в (6.2) благодаря использованию однородных координат, в которых трехмерный вектор записывается через четыре скаляра вместо трёх (Любой вектор $(x, y, z)^T$ трёхмерного пространства может быть записан в соответствующих однородных координатах в виде $(xw, yw, zw, w)^T$, где верхний индекс Т обозначает операцию транспонирования. Поскольку значение w может быть произвольным, для каждого вектора существует множество вариантов записи в однородных координатах. В формуле (6.2) используется значение $w = 1$. Полученная матрица, называется *матрицей однородного преобразования (homogeneous transformation matrix).* Если бы преобразование (в частности, трансляцию) нужно было применить к точке в двухмерном пространстве, однородная матрица преобразования редуцировалась бы до матрицы размерностью 3×3 удалением третьей строки и третьего столбца из матрицы размерностью 4×4. Новая матрица действовала бы на вектор координат размерностью 3×1, полученный из вектора 4×1 удалением z-координаты.

6.2. Вращение

Пусть объект поворачивается на угол θ вокруг оси x мировой системы координат вместе со своей модельной системой, которая, как и в предыдущем случае, изначально совпадает с мировой (рис. 6.2). Мировые координаты точки объекта в новом положении (X_w, Y_w, Z_w) могут быть получены из исходных мировых координат этой точки (X_m, Y_m, Z_m), совпадающих с её текущими координатами в модельной системе.

Соотношение между (X_w, Y_w, Z_w) и (X_m, Y_m, Z_m) становится очевидным после проецирования рис. 6.2, а на плоскость yz. Результат проецирования показан на рис. 6.2, *б*.



а *б*

Рис. 6.2. Вращение вокруг оси x: *а* – пространственное положение объекта; *б* – проекция на плоскость yz

Из рис. 6.2 можно легко получить следующие равенства:

$$X_w = X_m; \tag{6.3}$$

$$\begin{aligned} Y_w &= l\cos(\theta+\alpha) = l(\cos\theta\cos\alpha - \sin\theta\sin\alpha) = \\ &= l\cos\alpha\cos\theta - l\sin\alpha\sin\theta = Y_m\cos\theta - Z_m\sin\theta; \end{aligned} \tag{6.4}$$

$$\begin{aligned} Z_w &= l\sin(\theta+\alpha) = l(\sin\theta\cos\alpha + \cos\theta\sin\alpha) = \\ &= l\cos\alpha\sin\theta + l\sin\alpha\cos\theta = Y_m\sin\theta + Z_m\cos\theta. \end{aligned} \tag{6.5}$$

Равенства (6.3), (6.4) и (6.5) могут быть записаны в матричной форме:

$$\begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_m \\ Y_m \\ Z_m \\ 1 \end{bmatrix}. \tag{6.6}$$

Матрица в правой части формулы (6.6) – это однородная матрица преобразования вращения вокруг оси x, которая кратко обозначается $Rot(x, \theta)$.

Подобно матрице трансляции, для двумерного объекта однородная матрица вращения редуцируется до размера 3×3.

Однородные матрицы вращения вокруг осей y и z получаются аналогичным образом и записываются так:

$$Rot(y,\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}; \qquad (6.7)$$

$$Rot(z,\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \qquad (6.8)$$

Полученные матрицы преобразования описывают поворот вокруг мировых осей координат. Поворот вокруг любой произвольной оси раскладывается на повороты вокруг осей x, y и z. Таким образом, матрица преобразования для произвольной оси получается перемножением матриц (6.6)–(6.8).

6.3. Отображение

Отображение (mapping) – вычисление координат точки в некоторой системе координат по известным координатам той же точки в другой системе координат.

Рассмотрим две системы координат (рис. 6.3). Предположим, что координаты (X_2, Y_2, Z_2) точки P в системе координат $x_2y_2z_2$ должны быть вычислены по координатам (X_1, Y_1, Z_1) той же точки в системе $x_1y_1z_1$. Далее предположим, что вычисление производится применением матрицы преобразования $T_{1\text{-}2}$ к известным координатам:

$$\begin{bmatrix} X_2 & Y_2 & Z_2 & 1 \end{bmatrix}^T = T_{1-2} \begin{bmatrix} X_1 & Y_1 & Z_1 & 1 \end{bmatrix}^T. \qquad (6.9)$$

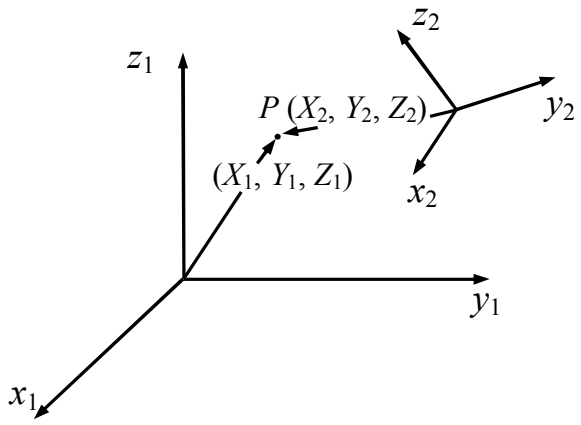


Рис. 6.3. Отображение из одной системы координат в другую

Записав матрицу $T_{1\text{-}2}$ в явном виде, получим из формулы (6.9) следующее выражение:

$$\begin{bmatrix} X_2 \\ Y_2 \\ Z_2 \\ 1 \end{bmatrix} = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \\ 1 \end{bmatrix}. \tag{6.10}$$

Чтобы найти неизвестные в уравнении (6.10), подставим в него конкретные значения $X_1 = 0$, $Y_1 = 0$ и $Z_1 = 0$, в результате чего получим

$$X_2 = p_x, Y_2 = p_y, Z_2 = p_z. \tag{6.11}$$

Можно сказать, что p_x, p_y и p_z определяют координаты начала отсчёта системы $x_1y_1z_1$ в системе координат $x_2y_2z_2$.

Теперь подставим в уравнение (6.10) значения $X_1 = 1$, $Y_1 = 0$ и $Z_1 = 0$ и получим

$$X_2 = n_x + p_x, Y_2 = n_y + p_y, Z_2 = n_z + p_z. \tag{6.12}$$

Вычитая формулы (6.11) из (6.12), можно заключить, что n_x, n_y и n_z – компоненты x_2, y_2 и z_2 единичного вектора, направленного вдоль оси x_1 системы координат $x_1y_1z_1$. Следовательно, коэффициенты n_x, n_y и n_z легко вычислить с учётом взаимной ориентации систем координат.

Аналогичным образом o_x, o_y и o_z представляют собой компоненты x_2, y_2 и z_2 единичного вектора оси y_1, а компоненты a_x, a_y и a_z – вектора оси z_1.

6.4. Масштабирование и зеркальное отображение

Помимо матриц преобразования, рассмотренных в предыдущих разделах, часто используются матрицы масштабирования и зеркального отображения.

Для масштабирования объекта с коэффициентом s_x по оси x, s_y – по оси y, s_z – по оси z применяется следующая матрица преобразования:

$$\begin{bmatrix} X' \\ Y' \\ Z' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \tag{6.13}$$

Для двухмерных объектов матрица масштабирования редуцируется до размера 3×3, как это было с матрицами трансляции и поворота. Эффекта масштабирования можно достичь, изменив размеры видового экрана или окна, не меняя значений координат.

Матрица преобразования (6.13) используется при масштабировании объекта относительно начала координат. Часто бывает необходимо масштабировать объект относительно одной из его точек P с координатами (X_p, Y_p, Z_p). В этом случае сначала к точке P применяется преобразование

трансляции $Trans(-X_p, -Y_p, -Z_p)$, которое перемещает эту точку в начало координат, затем применяется матрица масштабирования из (6.13), после чего объект возвращается в исходное положение действием $Trans(X_p, Y_p, Z_p)$.

Отражение относительно зеркальной плоскости *xy* может быть достигнуто при помощи приведённой ниже матрицы преобразования. Преобразование заключается в изменении знака координаты *z*.

$$\begin{bmatrix} X' \\ Y' \\ Z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \quad (6.14)$$

Матрицы преобразования для других отражений (относительно плоскостей *xz* и *yz*) выводятся аналогичным образом.

7. Представление кривых и работа с ними

Для каждого криволинейного ребра в компьютере хранится либо уравнение кривой, либо эквивалентные характеристические параметры (центр, радиус и вектор нормали к плоскости, в которой лежит окружность, – примеры характеристических параметров, эквивалентных уравнению окружности). Эти сведения важны как для систем автоматизированной разработки чертежей, так и для систем объемного моделирования. Расчёт точек пересечения кривых необходим для определения границ ксегментов при применении булевских операций. Ксегментом называется часть кривой, по которой пересекаются две грани, относящиеся к разным объемным телам. Ксегмент принадлежит обеим граням. Границы ксегмента определяются путем вычисления точек пересечения кривой, ограничивающей пересекающиеся поверхности, с кривой, по которой пересекаются эти поверхности (относящиеся к разным телам). После нахождения границ ксегмента нужно разделить кривую пересечения в точках пересечения. Аналогичная процедура выполняется при создании и модифицировании кривых в системах автоматизированной разработки чертежей и системах поверхностного моделирования. Рассмотрим методы представления уравнений кривых и методы работы с ними.

7.1. Типы уравнений

Уравнения кривых могут быть разделены на два основных типа. К первому типу относятся параметрические уравнения, описывающие связь координат *x*, *y* и *z* точки кривой с параметром. Ко второму – относятся непараметрические уравнения, связывающие координаты *x*, *y* и *z* некоторой функцией. Продемонстрируем различие между ними на примере. Рассмотрим окружность

радиуса R, расположенную в начале системы координат. Если окружность лежит в плоскости xy, её параметрическое уравнение может быть таким:

$$x = R\cos\theta,\ y = R\sin\theta,\ z = 0\ \ (0 \le \theta \le 2\pi). \qquad (7.1)$$

Ту же окружность можно описать уравнением и без параметра θ:

$$x^2 + y^2 - R^2 = 0,\ z = 0 \qquad (7.2)$$

или

$$y = \pm\sqrt{R^2 - x^2},\ z = 0\,. \qquad (7.3)$$

Уравнение (7.2) задает окружность в неявной непараметрической форме, а уравнение (7.3) – в явной непараметрической форме.

У каждого типа уравнений, примеры которых приведены выше, есть свои преимущества и недостатки, определяющие удобство их применения для различных целей. Сосредоточим внимание на применении уравнений к отображению кривых, поскольку интерактивная графика является одной из важнейших функций САПР. Кривая, отображаемая на экране, в действительности представляет собой набор коротких отрезков. Поэтому постоянно возникает необходимость вычислять координаты точек кривой, находящихся на равном расстоянии друг от друга. Это называется *вычислением кривой (curve evaluation).* Можно ожидать, что точки окружности, заданной уравнением (7.1), могут быть получены подстановкой последовательных значений параметра, отличающихся друг от друга на небольшую величину. При использовании уравнения (7.2), однако не известно, какую переменную следует выбрать в качестве независимой и последовательно увеличивать от точки к точке. Даже если выбрать независимую переменную, для каждого её значения находится два значения зависимой переменной. Это означает, что придётся выбирать одну из них таким образом, чтобы она располагалась по соседству с предыдущей найденной точкой. Уравнение (7.3) обладает тем же недостатком, несмотря на то что в нём независимая переменная уже выделена.

Из-за перечисленных недостатков непараметрического представления в системах автоматизированного проектирования чаще всего используются параметрические уравнения кривых и поверхностей, поэтому в дальнейшем рассматриваются только они.

7.2. Конические сечения

Кривые или части кривых, получаемые сечением конуса плоскостью, называются *коническими сечениями (conic sections).* В зависимости от положения и ориентации секущей плоскости по отношению к конусу кривая сечения может быть окружностью, эллипсом, параболой или гиперболой. В большинстве случаев профили деталей могут быть представлены в виде конических сечений, поскольку детали машин чаще всего обладают осевой симметрией.

7.3. Окружность и дуга окружности

Окружность или дуга окружности, лежащие в плоскости *xy*, с заданным радиусом *R* и координатами центра *Xc*, *Yc* могут быть представлены уравнениями

$$x = R\cos\theta + X_c;$$
$$y = R\sin\theta + Y_c. \qquad (7.4)$$

Как уже отмечалось, вычисление кривой может выполняться путем последовательной подстановки в уравнение значений θ с шагом $\Delta\theta$. Значение θ может достигать 2π для полной окружности или меньшего числа для дуги окружности. Значение $\Delta\theta$ должно быть подобрано таким образом, чтобы вычисление было достаточно быстрым, но окружность не получилась бы похожей на многоугольник. Уравнение окружности, лежащей в другой плоскости, может быть получено применением матриц преобразования к уравнению (7.4).

Рассмотрим пример получения параметрического уравнения окружности, применяя соответствующие матрицы преобразований к уравнению (7.4). Окружность единичного радиуса с центром в точке (0, 1, 1) лежит в плоскости *yz*, как показано на рис. 7.1. Уравнение этой кривой может храниться в памяти в виде характеристических параметров, таких как вектор нормали (1, 0, 0), координаты центра (0, 1, 1) и радиус 1. Как уже отмечалось, задание этих параметров эквивалентно написанию уравнения.

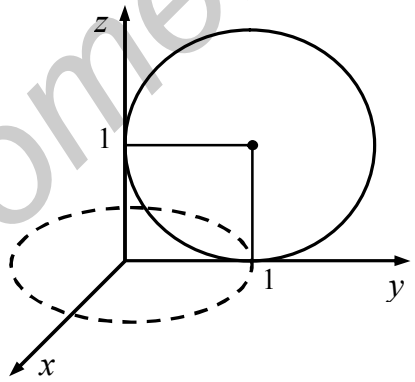


Рис. 7.1. Вычисление окружности

Исходная единичная окружность, лежащая в плоскости *xy,* изображена на рисунке пунктиром, а интересующая нас окружность – сплошной линией. Сплошная окружность получается из пунктирной поворотом на угол –90° вокруг оси *y* и последующей трансляцией на 1 в направлениях *y* и *z*. Обозначим координаты точек сплошной окружности буквами x^*, y^* и z^*, а координаты точек пунктирной окружности – буквами *x*, *y* и *z*. Тогда преобразования запишутся следующим образом:

$$\begin{bmatrix} x^* & y^* & 0 & 1 \end{bmatrix}^T = Trans(0,1,1)Rot(y,-90^0)\begin{bmatrix} x & y & 0 & 1 \end{bmatrix}^T =$$

$$= \begin{bmatrix} \cos(-90^0) & 0 & \sin(-90^0) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(90^0) & 0 & \cos(-90^0) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} =$$

$$= [0 \quad y+1 \quad x+1 \quad 1].$$

Отсюда

$$x^* = 0;$$
$$y^* = y + 1 = R\sin\theta + 1;$$
$$z^* = x + 1 = R\cos\theta + 1 \ \ (0 \le \theta \le 2\pi).$$

7.4. Эллипс и эллиптическая дуга

Эллипс, как и окружность, может быть задан параметрическим уравнением. Запишем такое уравнение для эллипса, лежащего в плоскости *xy*, с центром в начале координат. Положим, что большая ось эллипса направлена вдоль оси *x* и имеет длину *a*, а малая ось направлена вдоль оси *y* и имеет длину *b*. Параметрическое уравнение эллипса будет таким:

$$\begin{aligned} x &= a\cos\theta; \\ y &= b\sin\theta; \\ z &= 0. \end{aligned} \qquad (7.5)$$

Диапазон значений параметра для эллипса составляет [0, 2], а для дуги эллипса может быть более узким. Произвольный эллипс на произвольной плоскости с произвольными направлениями большой и малой осей получается в результате применения матриц преобразования подобно тому, как это делалось с окружностью.

Рассмотрим пример получения параметрического уравнения эллипса, лежащего в плоскости *xy,* с координатами центра X_C, Y_C. Оси эллипса направлены так, как показано на рис. 7.2.

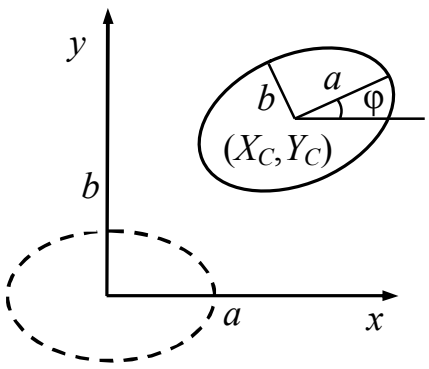


Рис. 7.2. Вычисление эллипса

Требуемый эллипс может быть получен поворотом исходного эллипса на угол вокруг оси z и трансляцией его на величину X_C в направлении x и на величину Y_C в направлении y. Обозначим координаты точек интересующего нас эллипса – буквами x^*, y^* и z^*, а координаты точек исходного эллипса буквами x, y и z. Тогда преобразования запишутся следующим образом:

$$\begin{bmatrix} x^* & y^* & 0 & 1 \end{bmatrix}^T = Trans(X_C, Y_C, 1)\, Rot(z, \varphi) \begin{bmatrix} x & y & 0 & 1 \end{bmatrix}^T =$$

$$= \begin{bmatrix} \cos\varphi & -\sin\varphi & 0 & 0 \\ \sin\varphi & \cos\varphi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & X_C \\ 0 & 1 & 0 & Y_C \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} =$$

$$= [x\cos\varphi - y\sin\varphi + X_C \quad x\sin\varphi + y\cos\varphi + Y_C \quad 0 \quad 1].$$

Следовательно

$$x^* = x\cos\varphi - y\sin\varphi = X_C = a\cos\theta\cos\varphi - b\sin\theta\sin\varphi + X_C;$$

$$y^* = x\sin\varphi + y\cos\varphi = Y_C = a\cos\theta\sin\varphi + b\sin\theta\cos\varphi + Y_C;$$

$$z^* = 0 \quad (0 \le \text{и} \le 2\pi).$$

7.5. Гипербола

Известно, что неявное уравнение гиперболы (рис. 7.3) имеет следующий вид:

$$\frac{x^2}{a^2} - \frac{y^2}{b^2} = 1. \qquad (7.6)$$

Уравнение (7.6) может быть записано в параметрической форме исходя из того, что $\text{ch}u = (e^u + e^{-u})/2$ и $\text{sh}u = (e^u - e^{-u})/2$. Из уравнения (7.6) можно получить и другие параметрические уравнения.

$$x = a\,\text{ch}\,u, \quad y = b\,\text{sh}\,u. \qquad (7.7)$$

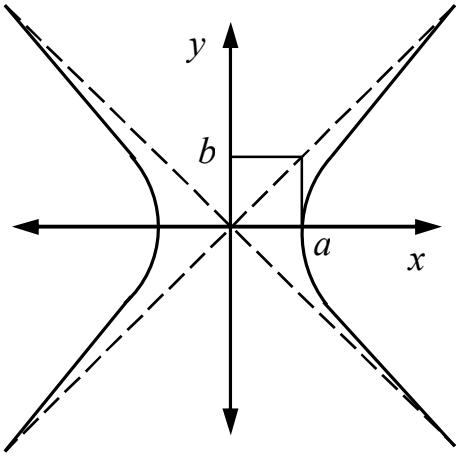


Рис. 7.3. Гипербола

Здесь используется известное тождество $\mathrm{ch}^2 u - \mathrm{sh}^2 u = 1$. Диапазон значений параметра u для уравнения (7.7) определяется исходя из координат конечных точек описываемой гиперболы. Применение соответствующих матриц преобразований к уравнению (7.7) позволяет получить уравнение гиперболы с центром в любой точке пространства, ориентированной произвольным образом.

7.6. Парабола

Парабола, симметричная относительно оси x и проходящая через начало координат, может быть задана следующим явным уравнением:

$$x = cu^2. \tag{7.8}$$

Это уравнение может быть преобразовано к параметрическому виду:

$$\begin{aligned} x &= cu^2; \\ y &= u. \end{aligned} \tag{7.9}$$

Заметим, что параметрическое уравнение (7.9) не является уникальным по отношению к уравнению (7.8): возможен выбор любого удобного параметрического уравнения. Диапазон значений параметра u в уравнении (7.9) выбирается исходя из координат концов описываемой параболы. Уравнение произвольной параболы с произвольными координатами центра и ориентацией может быть получено применением соответствующих матриц преобразований к уравнению (7.9).

7.7. Эрмитовы кривые

Чаще всего для описания кривых, используемых в программах CAD, используются уравнения третьего порядка, потому что они обладают важным свойством: две кривые, описываемые такими уравнениями, могут быть соединены таким образом, что вторые производные в точке соединения будут равны друг другу. Это означает, что кривизна в точке соединения остается постоянной, отчего две кривые кажутся одним целым. Ту же непрерывность можно получить и для кривых более высоких порядков, однако работа с ними требует интенсивных вычислений.

Простейшее параметрическое уравнение третьего порядка выглядит следующим образом:

$$\mathbf{P}(u) = \begin{bmatrix} x(u) & y(u) & z(u) \end{bmatrix} = \mathbf{a}_0 + \mathbf{a}_1 u + \mathbf{a}_2 u^2 + \mathbf{a}_3 u^3 \quad (0 \le u \le 1). \tag{7.10}$$

В формуле (7.10) $\mathbf{a}_0$, $\mathbf{a}_1$, $\mathbf{a}_2$, $\mathbf{a}_3$ – векторные коэффициенты параметрического уравнения, то есть векторы-строки с компонентами x, y и z. Координаты точки были представлены вектором-столбцом, следуя соглашениям, принятым создателями OpenGL. Здесь же используются векторы-строки, так как это делает более удобным представление матрицы геометрических коэффициентов. Эти коэффициенты являются алгебраическими

(уравнение (7.10) называется соответственно алгебраическим), поэтому их изменение не приводит к интуитивно понятному изменению формы кривой.

Чтобы преодолеть этот недостаток алгебраических коэффициентов, заменим их векторами, обладающими конкретным геометрическим значением. Один из возможных вариантов – использование радиус-векторов конечных точек кривой $\mathbf{P}_0$ и $\mathbf{P}_1$, а также векторов $\mathbf{P}_0'$ и $\mathbf{P}_1'$, задающих направление касательных в этих точках. Добавление граничных условий в виде векторов $\mathbf{P}_0$, $\mathbf{P}_1$, $\mathbf{P}_0'$, $\mathbf{P}_1'$ к формуле (7.10) даёт

$$\begin{aligned}
\mathbf{P}_0 &= \mathbf{P}(0) = \mathbf{a}_0;\\
\mathbf{P}_1 &= \mathbf{P}(1) = \mathbf{a}_0 + \mathbf{a}_1 + \mathbf{a}_2 + \mathbf{a}_3;\\
\mathbf{P}_0' &= \mathbf{P}'(0) = \mathbf{a}_1;\\
\mathbf{P}_1' &= \mathbf{P}'(1) = \mathbf{a}_1 + 2\mathbf{a}_2 + 3\mathbf{a}_3.
\end{aligned} \tag{7.11}$$

Уравнения (7.11) могут быть решены относительно $\mathbf{a}_0$, $\mathbf{a}_1$, $\mathbf{a}_2$, $\mathbf{a}_3$:

$$\begin{aligned}
\mathbf{a}_0 &= \mathbf{P}_0;\\
\mathbf{a}_1 &= \mathbf{P}_0';\\
\mathbf{a}_2 &= -3\mathbf{P}_0 + 3\mathbf{P}_1 - 2\mathbf{P}_0' - \mathbf{P}_1';\\
\mathbf{a}_3 &= 2\mathbf{P}_0 - 2\mathbf{P}_1 + \mathbf{P}_0' + \mathbf{P}_1'.
\end{aligned} \tag{7.12}$$

Подстановка (7.12) в (7.10) даст уравнение кривой в новом виде:

$$\mathbf{P}(u) = \begin{bmatrix} 1 - 3u^2 + 2u^3 & 3u^2 - 2u^3 & u - 2u^2 + u^3 & -u^2 + u^3 \end{bmatrix} \begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_0' \\ \mathbf{P}_1' \end{bmatrix}. \tag{7.13}$$

Теперь в уравнение кривой уже не входят алгебраические коэффициенты – вместо них стоят векторы $\mathbf{P}_0$, $\mathbf{P}_1$, $\mathbf{P}_0'$, $\mathbf{P}_1'$. Новые векторные коэффициенты называются геометрическими (*geometric coefficients*), а уравнение (7.13) называется уравнением *эрмитовой кривой* (*Hermite curve*), определяемой четырьмя векторами, поэтому они вводятся при её создании и сохраняются в качестве задающих ее параметров. Преимущество эрмитовой кривой в том, что изменение ее формы может быть интуитивно предсказано по изменению геометрических коэффициентов. Например, изменение $\mathbf{P}_0$ или $\mathbf{P}_1$ вызовет такое изменение кривой, что ее концы переместятся в новое положение, задаваемое измененными векторами $\mathbf{P}_0$ и $\mathbf{P}_1$.Точно так же изменение $\mathbf{P}_0'$ или $\mathbf{P}_1'$ приведет к тому, что касательные к кривой на ее концах станут совпадать с новыми векторами $\mathbf{P}_0'$ и $\mathbf{P}_1'$ (рис. 7.4).

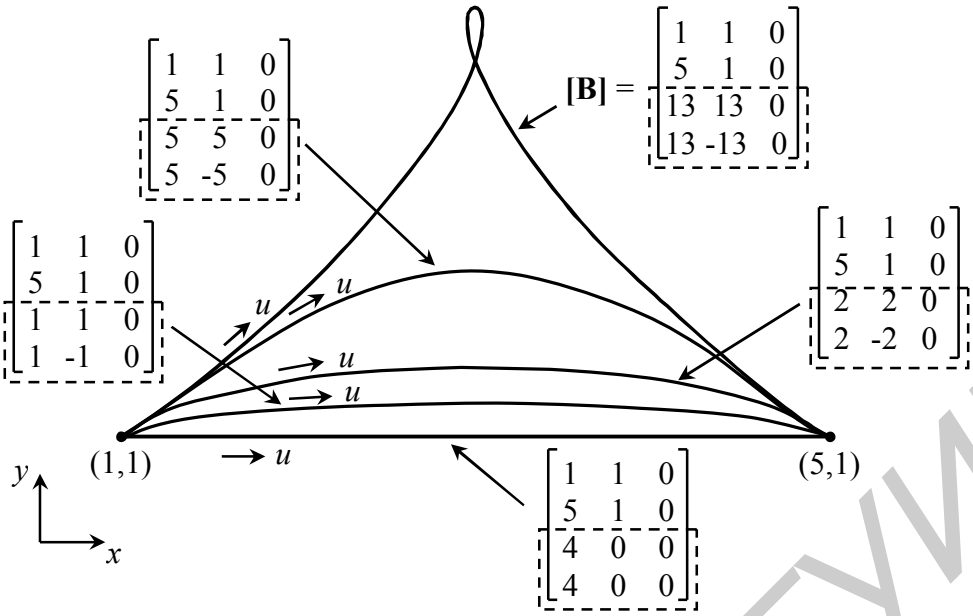


Рис. 7.4. Влияние касательных на форму кривой

Матрица геометрических коэффициентов обозначается буквой [**B**] и включает векторы $\mathbf{P}_0$, $\mathbf{P}_1$, $\mathbf{P}_0'$, $\mathbf{P}_1'$. Каждый вектор записывается в своей строке в той последовательности, в которой они перечислены выше. Рисунок демонстрирует только влияние изменения векторов, определяющих направление и длину касательных. Обратим внимание на то, что происходит с кривой, если меняется только длина, но не направление вектора касательной. Можно заключить, что длина вектора определяет, насколько далеко вдоль кривой по направлению к её середине распространяется влияние этого вектора.

Возможна иная интерпретация формулы (7.13). Векторы $\mathbf{P}_0$, $\mathbf{P}_1$, $\mathbf{P}_0'$, $\mathbf{P}_1'$ влияют на форму кривой, причём относительная степень их влияния определяется функциями

$$\begin{aligned} f_1(u) &= 1 - 3u^2 + 2u^3; \\ f_2(u) &= 3u^3 - 2u^3; \\ f_3(u) &= u - 2u^2 - u^3; \\ f_4(u) &= -u^2 + u^3. \end{aligned} \tag{7.14}$$

Можно сказать, что эти функции «сопрягают» граничные условия $\mathbf{P}_0$, $\mathbf{P}_1$, $\mathbf{P}_0'$, $\mathbf{P}_1'$ друг с другом, поэтому и называют их *функциями сопряжения.*

7.8. Кривая Безье

Хотя преобразование алгебраического уравнения кривой в форму эрмитова уравнения позволяет работать с кривой на интуитивном уровне, эрмитова форма не полностью удовлетворяет требованиям разработчиков. Как было

показано, предсказать форму кривой по величине векторов касательных $\mathbf{P}_0'$ и $\mathbf{P}_1'$ не так-то просто (рис. 7.4).

В 60-х гг. XX века независимо друг от друга Пьером Безье из автомобилестроительной компании «Рено» и Полем де Кастельжо из компании «Ситроен» были разработаны кривые Безье или кривые Бернштейна-Безье, где применялись для проектирования кузовов автомобилей. Несмотря на то, что открытие де Кастельжо было сделано несколько ранее Безье (1959), его исследования не публиковались и скрывались компанией как производственная тайна до конца 1960-х гг. Кривая Безье является частным случаем многочленов Бернштейна, описанных Сергеем Натановичем Бернштейном в 1912 году. Впервые кривые были представлены широкой публике в 1962 году Пьером Безье в системе поверхностного моделирования UNISURF и были названы его именем, а именем де Кастельжо назван разработанный им рекурсивный способ определения кривых (алгоритм де Кастельжо). Эта кривая строится по вершинам многоугольника, заключающего ее в себе. Вершины сопрягаются соответствующими функциями подобно тому, как это делается при построении эрмитовой кривой. Безье выбрал функции сопряжения таким образом, чтобы получающаяся кривая удовлетворяла следующим требованиям:

– кривая проходит через первую и последнюю вершины многоугольника.

– направление вектора касательной в первой точке кривой совпадает с направлением первого отрезка многоугольника (рис. 7.5). Аналогичным образом, последний отрезок многоугольника определяет направление касательной в конечной точке кривой;

– производная степени n в начальной (или конечной) точке кривой определяется положением первых (или последних) $n+1$ вершин многоугольника. Это свойство очень удобно при соединении двух кривых Безье, если требуется удовлетворить требованию непрерывности высших производных в точке соединения. Вообще говоря, второе свойство есть частный случай данного свойства. При изменении порядка вершин многоугольника на противоположный получается та же самая кривая.

Задавшись этими требованиями, Безье выбрал в качестве функций сопряжения полином Бернштейна:

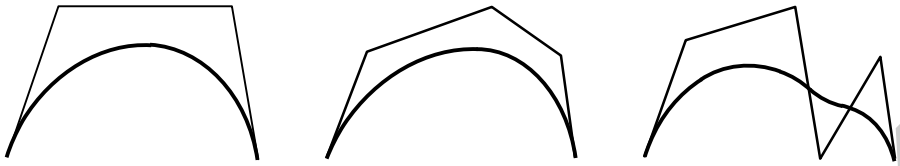
$$B_{i,n}(u) = \binom{n}{i} u'(1-u)^{n-i}, \tag{7.15}$$

где $\binom{n}{i} = \frac{n!}{i!(n-i)!}$.

Если функцию сопряжения (7.15) применить к вершинам многоугольника, получается уравнение кривой Безье:

$$\mathbf{P}(u) = \sum_{i=0}^{n} \binom{n}{i} u^i (1-u)^{n-i} \mathbf{P}_i, \tag{7.16}$$

где $\mathbf{P}_i$ – радиус-вектор i-й вершины. Вершины многоугольника называются *задающими (control vertices)*, как и сам многоугольник (*control polygon)*. Из формулы (7.16) видно, что для кривой, заданной $n + 1$ точками, максимальная степень будет u^n. Таким образом, степень кривой Безье определяется количеством задающих точек. Кривые Безье разных степеней с разным количеством задающих точек показаны на рис. 7.5.



а *б* *в*

Рис. 7.5. Кривые Безье различных степеней

Покажем теперь, что кривая Безье, заданная уравнением (7.16), удовлетворяет требованиям, перечисленным в начале раздела. Для начала проверим, что кривая проходит через первую и последнюю задающие точки. В этом легко убедиться, подставив в уравнение значения 0 и 1 параметра u. Раскроем суммирование следующим образом:

$$\mathbf{P}(u) = \binom{n}{0}(1-u)^n \mathbf{P}_0 + \binom{n}{1}u(1-u)^{n-1}\mathbf{P}_1 + \binom{n}{2}u^2(1-u)^{n-2}\vec{P}_2 + \ldots$$

$$+ \binom{n}{n-1}n^{n-1}(1-u)^1\mathbf{P}_{n-1} + \binom{n}{n}u^n\mathbf{P}_n.$$

Рассчитаем значения $\mathbf{P}(0)$ и $\mathbf{P}(1)$:

$$\mathbf{P}(0) = \binom{n}{0}\mathbf{P}_0 = \mathbf{P}_0;$$

$$\mathbf{P}(1) = \binom{n}{n}\mathbf{P}_n = \mathbf{P}_n.$$

Таким образом, кривая проходит через первую и последнюю задающие точки.

Второе и третье свойства можно проверить дифференцированием уравнения кривой Безье. Четвёртое свойство проверяется путём рассмотрения двух кривых, определяемых уравнениями

$$\mathbf{P}(u) = \sum_{i=0}^{n}\binom{n}{i}u^i(1-u)^{n-i}\mathbf{P}_i, \quad (0 \le u \le 1); \tag{7.17}$$

$$\mathbf{P}^*(v) = \sum_{i=0}^{n}\binom{n}{j}v^j(1-v)^{n-j}\mathbf{Q}_j \quad (\mathbf{Q}_i = \mathbf{P}_{n-j}, 0 \le v \le 1). \tag{7.18}$$

Уравнения (7.17), (7.18) можно интерпретировать следующим образом: кривая $\mathbf{P}(u)$ проводится от точки $\mathbf{P}_0$ к точке $\mathbf{P}_n$, тогда как $\mathbf{P}^*(v)$ проводится от $\mathbf{P}_n(=\mathbf{Q}_0)$ к $\mathbf{P}_0(=\mathbf{Q}_n)$. Если ввести новый параметр u^* и заменить v на $(1-u^*)$, направление $\mathbf{P}^*$ изменится на противоположное (то есть кривая будет идти от $\mathbf{P}_0$ к $\mathbf{P}_n$ при изменении u^* от 0 до 1). Поэтому можно будет сказать, что кривые $\mathbf{P}$ и $\mathbf{P}^*$ идентичны, если $\mathbf{P}^*(1-u^*)$ при разложении даст то же выражение, что и $\mathbf{P}(u)$ с точностью до названий параметров. Мы получим выражение для $\mathbf{P}^*(1-u^*)$, подставив $(1-u)$ вместо v в формулу (7.18):

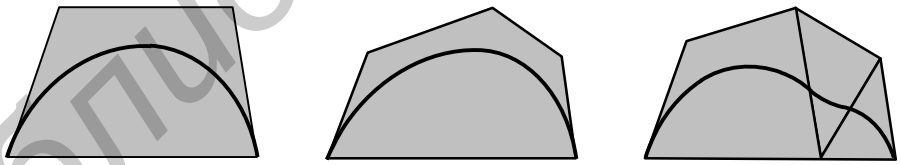
$$\mathbf{P}^*(1-u)=\sum_{j=0}^{n}\binom{n}{j}(1-u)^j u^{n-j}\mathbf{P}_{n-j}. \tag{7.19}$$

Перепишем выражение (7.19), подставив в него i вместо $(n-j)$:

$$\begin{aligned}\mathbf{P}*(1-u)&=\sum_{i=n}^{0}\binom{n}{n-i}(1-u)^{n-i}u^i\mathbf{P}_i=\\&=\sum_{i=0}^{n}\binom{n}{n-i}(1-u)^{n-i}u^i\mathbf{P}_i.\end{aligned} \tag{7.20}$$

Поскольку $\binom{n}{n-i}$ эквивалентно $\binom{n}{i}$, то можно заключить, что формула (7.20) совпадает с (7.17).

Кривая Безье обладает ещё одним важным свойством, помимо перечисленных выше, – *выпуклостью оболочки* (*convex hull property*). Выпуклой оболочкой кривой Безье является выпуклый многоугольник, получаемый соединением задающих точек (рис. 7.6). На этом рисунке все кривые Безье полностью лежат внутри своих выпуклых оболочек.



а *б* *в*

Рис. 7.6. Примеры выпуклых оболочек

Выпуклость оболочки кривой Безье обеспечивается тем, что значения сопрягающих функций лежат на отрезке [0,1], а их сумма равна 1 для любого u. Это утверждение легко проверить для кривой Безье, определяемой двумя задающими точками. Кривая, заданная точками $\mathbf{P}_0$ и $\mathbf{P}_1$, будет отрезком прямой, а её выпуклой оболочкой будет тот же самый отрезок. Любую точку этой кривой можно получить сложением $\mathbf{P}_0B_{0,1}+\mathbf{P}_1B_{1,1}$. Поскольку функции сопряжения $B_{0,1}$ и $B_{1,1}$ положительны и их сумма равна единице для любого u, *точка, определяемая выражением* $\mathbf{P}_0B_{0,1}+\mathbf{P}_1B_{1,1}$ будет лежать на отрезке $\mathbf{P}_0\mathbf{P}_1$ и

делить его в отношении $B_{1,1}$:$B_{0,1}$. Отрезок $\mathbf{P}_0\mathbf{P}_1$ в данном случае представляет собой вырожденную выпуклую оболочку, поэтому можно сказать, что все точки кривой лежат внутри этой оболочки. Аналогичным образом можно проверить высказанное выше утверждение для кривой Безье, заданной тремя и более точками. Выпуклость оболочки полезно использовать при вычислении точек пересечения кривых Безье. Оболочка кривой содержит её целиком, поэтому кривые Безье не могут пересекаться, если не перекрываются их оболочки. В этом случае трудоёмкое вычисление точки пересечения кривых Безье можно сразу же пропустить. Проверка перекрытия выпуклых оболочек требует гораздо меньших вычислительных затрат.

7.9. *B*-сплайн

Известно, что степень кривой Безье определяется количеством задающих точек, причём все они влияют на форму всей кривой. Эти особенности кривых Безье создают определённые неудобства. Во-первых, при аппроксимации кривой сложной формы при помощи кривой Безье неизбежно используется множество задающих точек, в результате чего получается кривая высокого порядка. Такая кривая может осциллировать, создавая большую вычислительную нагрузку на компьютер. В таком случае можно попытаться представить ту же исходную кривую множеством кривых Безье низших порядков. Однако при этом возникает проблема, которая заключается в том, что соединение кривых с обеспечением непрерывности производных нужных порядков оказывается довольно сложной процедурой.

Во-вторых, трудоёмким оказывается локальное изменение формы кривой. Кажется естественным переместить задающие точки вблизи изменяемого участка кривой; и действительно, это приводит к модификации нужной области, но вместе с ней изменяется вся кривая целиком. Эта особенность называется *свойством глобальности изменений (global modification property).* Глобальность изменений нежелательна при создании кривых заданной формы, поскольку кривые всегда создаются или проектируются путем непрерывной модификации грубой формы начального приближения. В системах автоматизированного проектирования желательно наличие прямо противоположного свойства – *локальности изменений*.

Описанные недостатки кривых Безье связаны с выбором функций сопряжения. Таким образом, необходимо выбрать новый набор функций сопряжения, обладающих определёнными свойствами. Во-первых, в определение новой функции сопряжения не должно входить число точек n в отличие от функции $B_{i,n}(u)$. Степень функции сопряжения, а значит, и степень кривой должны быть независимы от числа задающих точек n. Во-вторых, все функции сопряжения должны быть отличны от нуля только на ограниченных подмножествах значений параметра, причём для каждой функции такое подмножество должно быть уникальным. В этом случае форма сегмента будет

определяться только теми задающими точками, которые учитываются функциями сопряжения, имеющими ненулевые значения на данном сегменте.

В 1972 г. М. Кокс и Карл де Бур предложил использовать функции $N_{i,k}(u)$, определяемые рекурсивно. Кривая, которая строится таким образом, называется *B-сплайном* (*B-spline*) и записывается в следующем виде:

$$\mathbf{P}(u) = \sum_{i=0}^{n} \mathbf{P}_i N_{i,k}(u) \quad (t_{k-1} \le u \le t_{k+1}), \tag{7.21}$$

где

$$N_{i,k}(u) = \frac{(u - t_i) N_{i,k-1}(u)}{t_{i+k-1} - t_i} + \frac{(t_{i+k} - u) N_{i+1,k-1}(u)}{t_{i+k} - t_{i+1}}; \tag{7.22}$$

$$N_{i,1}(u) = \begin{cases} 1 & t_i \le u \le t_{i+1}; \\ 0 & \textit{в противном случае.} \end{cases} \tag{7.23}$$

Значения t_i, называются *узловыми* – они ограничивают отрезки значений параметра, внутри которых функции сопряжения имеют ненулевые значения (Когда u совпадает с границей интервала, следует учитывать, что поскольку для любого значения u только одна функция $N_{i,1}(u)$ может быть отличной от нуля. Это предполагается в определении (7.22). Например, когда $u = t_1$, только одна из функций $N_{0,1}(t_1)$ и $N_{1,1}(t_1)$ может быть равна единице, хотя из уравнения (7.23) следует, что они обе могли бы иметь это значение. В любом случае значение $\mathbf{P}(t_1)$ окажется одним и тем же. В формуле (7.22) неопределённость 0/0 считается равной нулю. Как следует из этого уравнения, для определения $n+1$ функций сопряжения необходимо задать $n+k+1$ узловых значений от t_0 до t_{n+k}. Разные методы задания узловых значений позволяют получить разные функции сопряжения и соответственно разные кривые. Из формулы (7.22) следует, что одновременный сдвиг всех узловых значений на одно и то же число не приводит к изменению формы кривой. При этом происходит лишь изменение диапазона значений параметра для уравнения (7.21).

Проверим, удовлетворяют ли функции сопряжения, заданные уравнениями (7.22) и (7.23), требованиям, изложенным в начале данного раздела. Из уравнения (7.22) следует, что степень $N_{i,k}(u)$ на единицу выше, чем у $N_{i,k-1}(u)$ и $N_{i+1,k-1}(u)$. Следовательно, $N_{i,2}(u)$ имеет степень 1, поскольку $N_{i,1}(u)$ – константа, а $N_{i,3}(u)$ по той же причине имеет степень 2. Продолжая в том же духе, можно прийти к выводу, что функция $N_{i,k}(u)$ имеет степень $k-1$. Таким образом, степень *B*-сплайна определяется не числом задающих точек, а значением k, которое должно быть на единицу больше желаемого значения степени кривой. Значение k называется *порядком B-сплайна* (*order of the B-spline curve*).

Чтобы показать, что любой участок *B*-сплайна определяется лишь конечным числом задающих точек, рассмотрим сегмент кривой, соответствующий значениям параметра из отрезка $[t_i, t_{i+1}]$. Влиять на этот сегмент будут те задающие точки, функции сопряжения которых (порядка k)

отличны от нуля на отрезке $[t_i, t_{i+1}]$. Среди функций первого порядка от нуля отлична только $N_{i,1}(u)$. Подставив $N_{i,1}(u)$ в правую часть формулы (7.32), получим на рассматриваемом отрезке ненулевые функции $N_{i,2}(u)$ и $N_{i-1,2}(u)$. Первая функция получается подстановкой $N_{i,1}(u)$ в первое слагаемое, а вторая функция – подстановкой во второе слагаемое. Затем по ненулевым функциям $N_{i,2}(u)$ и $N_{i-1,2}(u)$ получаем функции третьего порядка и т. д., пока не будут получены функции порядка k. Распространение ненулевых значений иллюстрирует рис. 7.7.

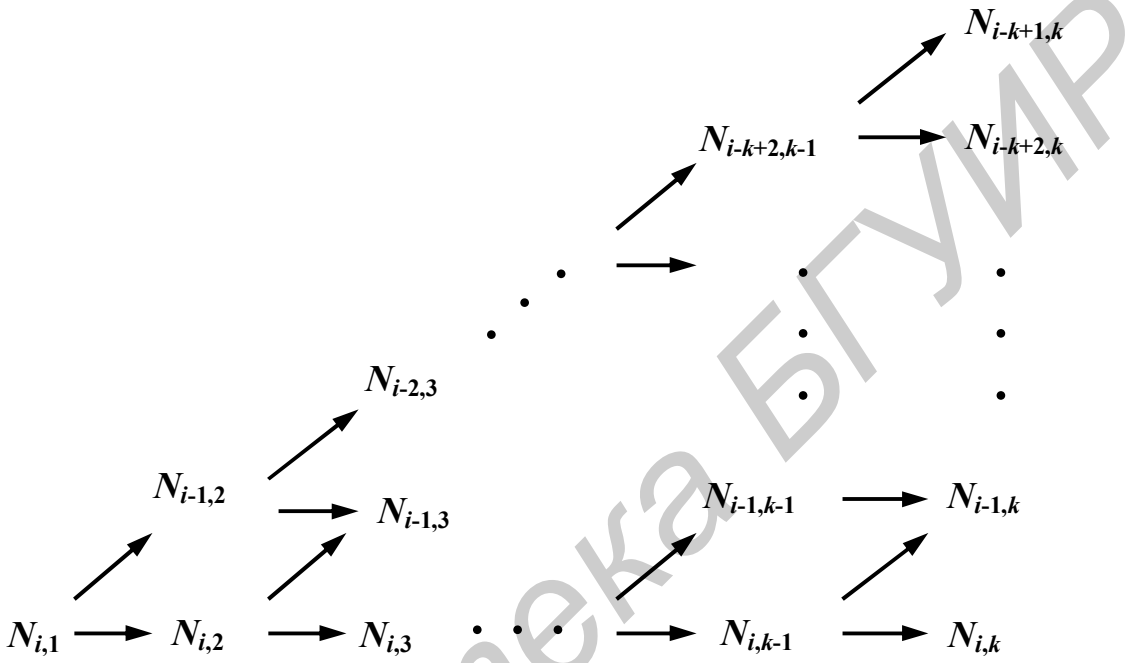


Рис. 7.7. Распространение значений $N_{i,1}(u)$

По рис. 7.7 видно, что ненулевые значения на отрезке $[t_i, t_{i+1}]$ будут иметь только функции $N_{i-k+1,k}$, $N_{i-k+2,k}$, … , $N_{i,k}$. Поэтому и влиять на форму отрезка кривой будут только точки $\mathbf{P}_{i-k+1}$, $\mathbf{P}_{i-k+2}$, … , $\mathbf{P}_i$. (всего k штук). Например, если взять кривую четвёртого порядка, то на форму отрезка будут влиять четыре задающие точки, а все остальные не будут.

Определим $n+k+1$ узловых значений от t_0 до t_{n+k}. Узлы бывают двух основных типов: периодические и непериодические. Периодические узлы определяются из равенства

$$t_i = i - k \quad (0 \le i \le n+k). \tag{7.24}$$

Непериодические узлы задаются формулой

$$t_i = \begin{cases} 0 & 0 \le i \le k; \\ i-k+1 & k \le i \le n; \\ n-k+2 & n \le i \le n+k. \end{cases} \tag{7.25}$$

Главное отличие узлов разных типов состоит в том, что первый и последний непериодические узлы повторяются k раз. Повторение узлов приводит к тому, что B-сплайн проходит через первую и последнюю задающие точки подобно кривой Безье. В периодическом B-сплайне первая и последняя точки влияют на форму кривой в той же степени, что и все остальные точки, а потому кривая не проходит через них. Вообще говоря, функция сопряжения для периодических узлов повторяется через равные промежутки значений параметра, почему узлы и называются периодическими. Непериодические кривые чаще используются в САПР, поскольку большинству конструкторов привычнее работать с кривыми, проходящими через первую и последнюю точки.

Выражения (7.24) и (7.25) показывают, что расстояние между соседними узлами всегда одинаково и равно единице. Такие узлы называются *однородными*, как и B-сплайн, через них проходящий. Однородный B-сплайн может быть как периодическим, так и непериодическим. При изменении формы кривой конструкторы часто добавляют и удаляют узлы, в результате чего B-сплайн становится неоднородным. Поскольку однородность является частным случаем неоднородности, неоднородные B-сплайны считаются обобщением однородных. Таким образом, большинство САПР позволяют создавать и модифицировать неоднородные и непериодические B-сплайны.

7.10. Неоднородный рациональный *B*-сплайн (NURBS)

Неоднородный рациональный B-сплайн, или попросту *NURBS* (*nonuniform rational B-spline curve*), подобен обычному неоднородному B-сплайну, поскольку основан на тех же функциях сопряжения, получаемых для неоднородных узлов. Однако задающие точки рационального B-сплайна указываются с использованием однородных координат, к которым применяются функции сопряжения. Координаты точки на кривой NURBS в однородном пространстве вычисляются по следующим формулам:

$$x \cdot h = \sum_{i=0}^{n} (h_i \cdot x_i) N_{i,k}(u); \tag{7.26}$$

$$y \cdot h = \sum_{i=0}^{n} (h_i \cdot y_i) N_{i,k}(u); \tag{7.27}$$

$$z \cdot h = \sum_{i=0}^{n} (h_i \cdot z_i) N_{i,k}(u); \tag{7.28}$$

$$h = \sum_{i=0}^{n} h_i N_{i,k}(u). \tag{7.29}$$

Координаты точки в трёхмерном пространстве x, y и z получаются делением xh, yh и zh на h, поэтому уравнение кривой NURBS в векторном виде может быть записано следующим образом (здесь поделены уравнения (7.26), (7.27) и (7.28) на (7.29)):

$$\mathbf{P}(u) = \frac{\sum_{i=0}^{n} h_i \mathbf{P}_i N_{i,k}(u)}{\sum_{i=0}^{n} h_i N_{i,k}(u)}, \qquad (7.30)$$

где $\mathbf{P}_i$, – вектор (x_i, y_i, z_i), объединяющий координаты i-й задающей точки в трёхмерном пространстве, как и для нерациональных B-сплайнов. Диапазон значений параметра t_{k-1} и t_{n+1} такой же, как и для B-сплайна.

Уравнение (7.30) отражает следующие свойства кривой NURBS:

– подобно нерациональному B-сплайну, кривая NURBS, представленная уравнением (7.30), обязательно проходит через первую и последнюю задающие точки, если используются непериодические узлы. Это утверждение можно доказать следующим образом. Числитель формулы (7.30) может считаться B-сплайном с задающими точками $h_i\mathbf{P}_i$. Граничным значениям параметра будут соответствовать координаты $h_0\mathbf{P}_0$ и $h_n\mathbf{P}_n$, поскольку B-сплайн с непериодическими узлами проходит через первую и последнюю задающие точки. Знаменатель (7.30) также может считаться B-сплайном с задающими точками h_i. Граничным значениям параметра для этого B-сплайна будут соответствовать координаты h_0 и h_n. Следовательно, граничным значениям параметра для $\mathbf{P}(u)$ в целом будут отвечать координаты $\mathbf{P}_0$ и $\mathbf{P}_n$, то есть граничные задающие точки;

– касательная в начальной точке совпадает по направлению с вектором $\mathbf{P}_1 - \mathbf{P}_0$, а касательная в конечной точке – с вектором $\mathbf{P}_n - \mathbf{P}_{n-1}$;

– знаменатель (7.30) становится равным 1, когда все h_i-е становятся равными 1, поскольку $\sum_{i=0}^{n} N_{i,k}(u) = 1$. Уравнение (7.30) при этом превращается в уравнение B-сплайна. Поэтому можно сказать, что уравнение NURBS может описывать не только NURBS-кривые, но и обычные B-сплайны. Кривая Безье является частным случаем B-сплайна, поэтому уравнение NURBS может также описывать кривые Безье и рациональные кривые Безье (уравнение рациональной кривой Безье получается подстановкой $B_{in}(u)$ вместо $N_{jk}(u)$ в уравнение (7.30), то есть функции сопряжения Безье используются с задающими точками в однородных координатах).

Уравнение NURBS-кривой обладает определёнными преимуществами перед уравнением B-сплайна:

– форма B-сплайна изменяется при изменении координат x, y и z задающих точек. Для каждой задающей точки, таким образом, имеется три степени свободы. Кривая NURBS позволяет изменять четвёртую координату для каждой точки – h_i. Появляется возможность работать с кривой на более тонком уровне. Увеличение значения однородной координаты задающей точки приводит к тому, что NURBS-кривая притягивается ближе к этой точке;

– уравнение NURBS позволяет точно воспроизвести все конические сечения – окружность, эллипс, параболу и гиперболу. B-сплайны, напротив,

допускают лишь приближение к коническим сечениям. Таким образом, в NURBS-представлении можно работать с коническими сечениями, кривыми Безье, рациональными кривыми Безье и *B*-сплайнами. Преобразование всех этих кривых к NURBS может значительно сократить объём программирования. Например, можно написать одну-единственную программу для расчёта точки пересечения кривых NURBS и использовать её для расчёта пересечений кривых любых типов, поскольку эти кривые всегда могут быть преобразованы к соответствующим NURBS-представлениям.

8. Представление поверхностей и работа с ними

Системам геометрического моделирования приходится хранить уравнения кривых (или эквивалентные характеристические параметры) для всех рёбер и уравнения поверхностей (или эквивалентные параметры) для всех граней. Полезно знать, какими бывают уравнения поверхностей и какими параметрами они характеризуются. Для вычислений пространственных положений трёхмерных тел, реализованных при помощи булевских операций необходимо рассчитывать точки пересечения кривых. В системах поверхностного моделирования с той же целью приходится определять кривые, по которым пересекаются поверхности. Например, если поверхность в процессе моделирования отсекается другой поверхностью, это требует вычисления кривой, по которой они пересекаются. Эта задача требует вычисления координат точек на поверхностях, а также производных в этих точках.

8.1. Типы уравнений поверхностей

Уравнения поверхностей, как и уравнения кривых, делятся на два основных типа. К первому типу относятся *параметрические уравнения*, связывающие значения координат *x, y* и *z* со значениями параметра. Ко второму относятся *непараметрические уравнения,* связывающие координаты *x*, *y* и *z* непосредственно друг с другом какой-либо функцией. Поясним эти определения на простом примере. Рассмотрим сферу радиуса *R* с центром в начале координат. Параметрическое уравнение этой сферы будет иметь вид

$$\begin{aligned}&\mathbf{P}(u,v) = R\cos u\cos v\mathbf{i} + R\sin u\cos v\mathbf{j} + R\sin v\mathbf{k};\\&(0 \le u \le 2\pi,\ -\pi/2 \le v \le \pi/2),\end{aligned} \tag{8.1}$$

где параметр *u* может рассматриваться как долгота, а *v* – как широта. Ту же сферу можно описать и без параметров *u* и *v*:

$$x^2 + y^2 + z^2 - R^2 = 0 \tag{8.2}$$

или

$$z = \pm\sqrt{R^2 - x^2 - y^2}. \tag{8.3}$$

Уравнение (8.2) называется *неявным непараметрическим*, а уравнение (8.3) – *явным непараметрическим*.

У каждого типа уравнений есть свои преимущества и недостатки, но нас интересуют только параметрические уравнения. Параметрическое уравнение позволяет эффективно вычислять точки на поверхности или кривой, отстоящие друг от друга на небольшие расстояния, что облегчает интерактивное отображение объекта и работу с ним. Это одна из главных причин, по которым параметрические уравнения используются для представления поверхностей в большинстве САПР. Кривые, по которым пересекаются поверхности, в некоторых случаях бывает удобно рассчитывать, представляя одну из этих поверхностей параметрическим уравнением, а другую – непараметрическим. Поэтому в некоторых системах могут использоваться и непараметрические уравнения поверхностей, хотя и не как основной тип уравнений для хранения сведений о кривых. В этом случае возникает потребность в процедуре преобразования из параметрической формы в непараметрическую и обратно.

8.2. Билинейная поверхность

Билинейная поверхность строится по четырём заданным точкам и описывается линейными уравнениями с параметрами u и v. Эти точки оказываются в углах построенной поверхности. Обозначим их буквами $\mathbf{P}_{0,0}$, $\mathbf{P}_{1,0}$, $\mathbf{P}_{0,1}$, $\mathbf{P}_{1,1}$ (рис. 8.1). Вывести уравнение билинейной поверхности – это значит найти выражение для координат произвольной точки по значениям параметров u и v. Предположим, что эта точка делит отрезок $[\mathbf{P}_{0,v}, \mathbf{P}_{1,v}]$ в отношении u:$(1-u)$. Точки $\mathbf{P}_{0,v}$ и $\mathbf{P}_{1,v}$ делят отрезки $\mathbf{P}_{0,0}\mathbf{P}_{0,1}$ и $\mathbf{P}_{1,0}\mathbf{P}_{1,1}$ соответственно в отношении v:$(1-v)$. Определённая таким образом точка $\mathbf{P}(u, v)$ будет перемещаться по всей поверхности при изменении параметров u и v от 0 до 1. С учётом сделанных предположений координаты точек $\mathbf{P}_{0,v}$ и $\mathbf{P}_{1,v}$ запишутся следующим образом:

$$\mathbf{P}_{0,v} = (1-v)\mathbf{P}_{0,0} + v\mathbf{P}_{0,1}; \tag{8.4}$$

$$\mathbf{P}_{1,v} = (1-v)\mathbf{P}_{1,0} + v\mathbf{P}_{1,1}. \tag{8.5}$$

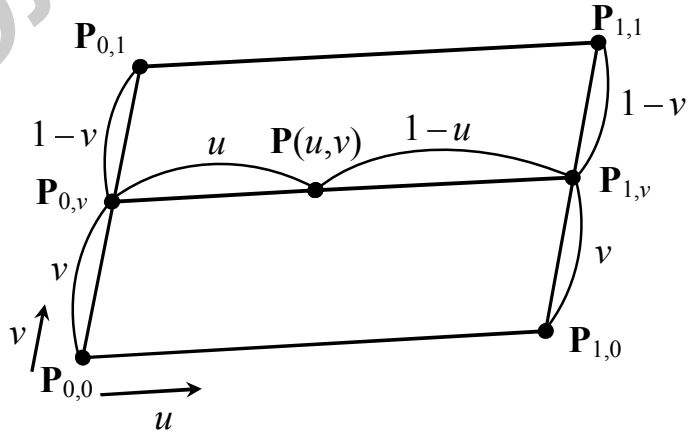


Рис. 8.1. Билинейная поверхность и её опорные точки

Аналогичным образом получаются координаты точки $\mathbf{P}(u,v)$:

$$\mathbf{P}(u,v) = (1-u)\mathbf{P}_{0,v} + u\mathbf{P}_{1,v}. \tag{8.6}$$

Подстановка уравнений (8.4) и (8.5) в (8.6) даст приведённое ниже уравнение билинейной поверхности:

$$\mathbf{P}(u,v) = (1-u)[(1-v)\mathbf{P}_{0,0} + v\mathbf{P}_{0,1}] + u[(1-v)\mathbf{P}_{1,0} + v\mathbf{P}_{1,1} =$$

$$= [(1-u)(1-v) \quad u(1-v) \quad (1-u)v \quad uv] \begin{bmatrix} \mathbf{P}_{0,0} \\ \mathbf{P}_{1,0} \\ \mathbf{P}_{0,1} \\ \mathbf{P}_{1,1} \end{bmatrix} \quad (0 \le u \le 1,\ 0 \le v \le 1). \tag{8.7}$$

Можно убедиться, что заданные точки расположены по углам билинейной поверхности, подставив соответствующие комбинации нулей и единиц в уравнение (8.7). Это уравнение говорит также о том, что билинейная поверхность представляет собой сопряжение угловых точек при помощи функций сопряжения $(1-u)(1-v)$, $u(1-v)$, $(1-u)v$, uv. Из-за того что эти функции сопряжения линейны по соответствующим параметрам, билинейная поверхность обычно оказывается плоской.

8.3. Лоскут Куна

Сопряжение углов даёт билинейную поверхность. Сопряжение граничных кривых произвольной формы даёт поверхность, называемую *лоскутом Куна*. Слово «лоскут» указывает на то, что описываемая поверхность представляет собой сегмент, соответствующий значениям параметров $0 \le u < 1$, $0 \le v < 1$. Комбинирование лоскутов позволяет образовать поверхность произвольной формы и размеров.

Выведем уравнение лоскута Куна. Предположим, что известны уравнения четырёх граничных кривых: $\mathbf{P}_0(v)$, $\mathbf{P}_1(v)$, $\mathbf{Q}_0(u)$ и $\mathbf{Q}_1(u)$ (рис. 8.2). Предположим также, что направление кривых $\mathbf{Q}_0(u)$ и $\mathbf{Q}_1(u)$ совпадает (на рис. 8.2 эти кривые направлены вправо, что обозначено стрелкой). То же предположение выскажем и относительно $\mathbf{P}_0(v)$ и $\mathbf{P}_1(v)$.

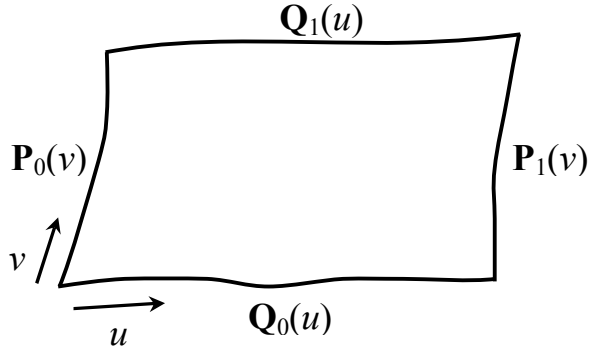


Рис. 8.2. Граничные кривые, определяющие лоскут Куна

Если граничные кривые не удовлетворяют этим требованиям, то придётся выполнить преобразование их к описанному выше виду. Направление и интервал изменения параметра легко изменить инверсией или масштабированием. Удовлетворяющие описанным требованиям кривые интерполируются следующим образом.

Выберем две кривые, расположенные друг напротив друга, например $\mathbf{P}_0(v)$ и $\mathbf{P}_1(v)$. Интерполяция этих кривых в направлении u осуществляется линейным уравнением

$$\mathbf{P}_1(u,v) = (1-u)\mathbf{P}_0 + u\mathbf{P}_1(v). \tag{8.8}$$

Поверхность, определённая уравнением (8.8), будет ограничена кривой $\mathbf{P}_0(v)$ при $u=0$ и кривой $\mathbf{P}_1(v)$ при $u=1$. Однако две другие границы будут отрезками прямых, соединяющих угловые точки. Убедиться в этом можно, подставив в уравнение (8.8) $v=0$ или $v=1$. Таким образом, полученная поверхность не ограничивается кривыми $\mathbf{Q}_0(u)$, $\mathbf{Q}_1(u)$.

Определим вторую поверхность, интерполируя $\mathbf{Q}_0(u)$ и $\mathbf{Q}_0(v)$ в направлении v:

$$\mathbf{P}_2(u,v) = (1-v)\mathbf{Q}_0(u) + v\mathbf{Q}_1(u). \tag{8.9}$$

Подставляя граничные значения u и v в уравнение (8.9), можно убедиться, что новая поверхность ограничивается кривыми $\mathbf{Q}_0(u)$, $\mathbf{Q}_0(v)$, но не $\mathbf{P}_0(v)$ или $\mathbf{P}_1(v)$. Попробуем определить ещё одну поверхность $\mathbf{P}_3(u, v)$, сложив $\mathbf{P}_1(u, v)$ и $\mathbf{P}_2(u, v)$, и проверим, не будет ли она ограничиваться требуемыми кривыми.

$$\mathbf{P}_3(u,v) = (1-u)\mathbf{P}_0(v) + u\mathbf{P}_1(v) + (1-v)\mathbf{Q}_0(u) + v\mathbf{Q}_1(u). \tag{8.10}$$

Подстановка граничных значений u и v в (8.10) даёт:

$$\mathbf{P}_3(0,v) = \mathbf{P}_0(v) + (1-v)\mathbf{Q}_0(0) + v\mathbf{Q}_1(0); \tag{8.11}$$

$$\mathbf{P}_3(1,v) = \mathbf{P}_1(v) + (1-v)\mathbf{Q}_0(1) + v\mathbf{Q}_1(1); \tag{8.12}$$

$$\mathbf{P}_3(u,0) = \mathbf{Q}_0(u) + (1-u)\mathbf{P}_0(0) + u\mathbf{P}_1(0); \tag{8.13}$$

$$\mathbf{P}_3(u,1) = \mathbf{Q}_1(u) + (1-u)\mathbf{P}_0(1) + u\mathbf{P}_1(1). \tag{8.14}$$

Если $\mathbf{P}_3(u, v)$ удовлетворяет поставленным требованиям к граничным кривым, правые два слагаемых в уравнениях (8.11) – (8.14) должны быть равны нулю. Заметим, что эти слагаемые представляют собой интерполяцию конечных точек соответствующих граничных кривых. Другими словами, слагаемые, которые должны быть равны нулю, описывают границы билинейной поверхности. Следовательно, правильное выражение для лоскута Куна получается вычитанием уравнения билинейной поверхности из $\mathbf{P}_3(u, v)$:

$$\begin{aligned}\mathbf{P}(u,v) = {} & (1-u)\mathbf{P}_0(v) + u\mathbf{P}_1(v) + (1-v)\mathbf{Q}_0(u) + v\mathbf{Q}_1(u) - (1-u)(1-v)\mathbf{P}_{0,0} - \\ & - u(1-v)\mathbf{P}_{1,0} - (1-u)v\mathbf{P}_{0,1} - uv\mathbf{P}_{1,1} \quad (0 \le u \le 1,\ 0 \le v \le 1).\end{aligned} \tag{8.15}$$

Здесь $\mathbf{P}_{0,0}=\mathbf{P}_0(0)=\mathbf{Q}_0(0)$, $\mathbf{P}_{1,0}=\mathbf{P}_1(0)=\mathbf{Q}_0(1)$, $\mathbf{P}_{0,1}=\mathbf{P}_0(1)=\mathbf{Q}_1(0)$, $\mathbf{P}_{1,1}=\mathbf{P}_1(1)=\mathbf{Q}_1(1)$.

Благодаря простоте концепции и уравнений лоскут Куна использовался достаточно широко. Однако он непригоден для точного моделирования поверхностей, поскольку форма поверхности не может задаваться одними лишь её границами.

8.4. Бикубический лоскут

Бикубический лоскут (bicubic patch) – это поверхность, определяемая полиномиальным уравнением третьего порядка по параметрам *u* и *v*:

$$\mathbf{P}(u,v)=\sum_{i=0}^{3}\sum_{j=0}^{3}a_{ij}u^{i}v^{j}\quad(0\le u\le 1,\ 0\le u\le 1). \tag{8.16}$$

Уравнение (8.16) можно переписать в матричной форме:

$$\mathbf{P}(u,v)=\begin{bmatrix}1 & u & u^2 & u^3\end{bmatrix}\begin{bmatrix}a_{00} & a_{01} & a_{02} & a_{03}\\ a_{10} & a_{11} & a_{12} & a_{13}\\ a_{20} & a_{21} & a_{22} & a_{23}\\ a_{30} & a_{31} & a_{32} & a_{33}\end{bmatrix}\begin{bmatrix}1\\ v\\ v^2\\ v^3\end{bmatrix}. \tag{8.17}$$

В уравнениях (8.17) $\mathbf{a}_{ij}$ – алгебраические векторные коэффициенты с компонентами *x*, *y* и *z*. Влияние этих коэффициентов на форму поверхности не является интуитивно понятным, точно так же, как по алгебраическим коэффициентам уравнения (7.10) нельзя было представить себе форму кривой. Возникает желание заменить алгебраические коэффициенты на геометрические, что было сделано ранее при выводе эрмитовой кривой. Поскольку для поверхности насчитывается 16 алгебраических коэффициентов, необходимо ввести 16 граничных условий.

Первый набор граничных условий получим, потребовав, чтобы четыре граничные точки **P**(0,0), **P**(0,1), **P**(1,0), **P**(1,1) удовлетворяли уравнению (8.17). Чтобы получить второй набор граничных условий, зададим векторы касательных к граничным кривым поверхности в угловых точках по параметрам *u* и *v*: $\mathbf{P}_u(0,0)$, $\mathbf{P}_u(0,1)$, $\mathbf{P}_u(1,0)$, $\mathbf{P}_u(1,1)$ и $\mathbf{P}_v(0,0)$, $\mathbf{P}_v(0,1)$, $\mathbf{P}_v(1,0)$, $\mathbf{P}_v(1,1)$. Перечисленные граничные условия определяют форму граничных кривых поверхности, поскольку они определяют конечные точки этих кривых и векторы касательных в этих точках. Через заданные граничные кривые можно провести бесконечно много поверхностей, поэтому необходимо добавить граничные условия, которые определяли бы форму внутренней области поверхности. Потребуем, чтобы вторая производная в угловых точках имела определенные значения $\mathbf{P}_{uv}(0,0)$, $\mathbf{P}_{uv}(0,1)$, $\mathbf{P}_{uv}(1,0)$, $\mathbf{P}_{uv}(1,1)$. Под второй производной необходимо понимать следующее выражение, называемое *вектором кручения (twist vector)*:

$$\frac{\partial^2\mathbf{P}(u,v)}{\partial u\,\partial v}.$$

Далее рассмотрим влияние векторов кручения в угловых точках на форму внутренней области поверхности.

Подстановка 16 граничных условий в уравнение (8.17) даст 16 линейных по $\mathbf{a}_{ij}$ уравнений. Эти уравнения образуют систему, результат решения которой подставляется обратно в уравнение (8.17) (уравнение эрмитовой кривой получается при помощи аналогичной процедуры), в результате чего получаем приведённое ниже уравнение бикубического лоскута:

$$\mathbf{P}(u,v) =$$

$$= \begin{bmatrix} F_1(u) & F_2(u) & F_3(u) & F_4(u) \end{bmatrix} \begin{bmatrix} \mathbf{P}(0,0) & \mathbf{P}(0,1) & \mathbf{P}_v(0,0) & \mathbf{P}_v(0,1) \\ \mathbf{P}(1,0) & \mathbf{P}(1,1) & \mathbf{P}_v(1,0) & \mathbf{P}_v(1,1) \\ \mathbf{P}_u(0,0) & \mathbf{P}_u(0,1) & \mathbf{P}_{uv}(0,0) & \mathbf{P}_{uv}(0,1) \\ \mathbf{P}_u(1,0) & \mathbf{P}_u(1,1) & \mathbf{P}_{uv}(1,0) & \mathbf{P}_{uv}(1,1) \end{bmatrix} \begin{bmatrix} F_1(v) \\ F_2(v) \\ F_3(v) \\ F_4(v) \end{bmatrix} \quad (8.18)$$

$(0 \le u \le 1, 0 \le v \le 1)$,

где функции сопряжения F_1 F_2, F_3 и F_4 определяются следующим образом:

$$\begin{aligned} F_1(u) &= 1 - 3u^2 + 2u^3; \\ F_2(u) &= 3u^2 - 2u^3; \\ F_3(u) &= u - 2u^2 + u^3; \\ F_4(u) &= -u^2 + u^3. \end{aligned} \quad (8.19)$$

Это те же функции сопряжения, что и в уравнении эрмитовой кривой. Вообще говоря, уравнение (8.18) представляет собой попросту расширение уравнения эрмитовой кривой для описания поверхности. Уравнение (8.18) может быть редуцировано до уравнения кривой путём подстановки конкретного значения одного из параметров, например $v = v_0$. Произведение последних двух матриц в правой части уравнения (8.18) даёт вектор-столбец. Это умножение может быть интерпретировано следующим образом. Первая строка дает уравнение левой граничной кривой между **P**(0,0) и **P**(0,1), а вторая строка дает уравнение правой граничной кривой между **P**(1,0) и **P**(1,1) (рис. 8.3).

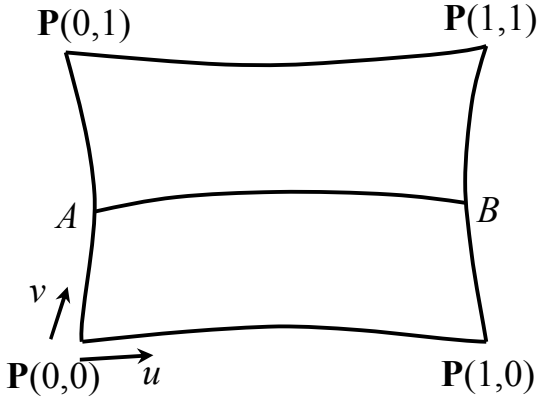


Рис. 8.3. Изопараметрическая кривая $v = v_0$

При подстановке конкретного значения v_0 первые две строки дают конечные точки A и B. Последние две строки столбца для $v = v_0$ задают векторы касательных в направлении u в точках A и B. Последнее утверждение проверим позже. Следовательно, уравнение (8.18) представляет собой объединение уравнений эрмитовых кривых, соответствующих разным значениям v. Граничные кривые, определяемые условиями $v = 0$ и $v = 1$, будут эрмитовыми кривыми. Аналогичным образом можно показать, что и граничные кривые для $u = 0$, $u = 1$ тоже будут эрмитовыми.

Если воспользоваться уравнением (8.18), то можно столкнуться с проблемой задания векторов кручения, влияние которых на форму поверхности не является интуитивно понятным. Иногда для простоты им присваивают нулевые значения. Получаемая таким способом поверхность называется *лоскутом Фергюсона*, или *F-лоскутом (Ferguson's patch*, или *F-patch)*. Поскольку нулевые векторы кручения делают поверхность более плоской, F-лоскут не может использоваться для моделирования поверхностей с большой кривизной. Однако он хорошо описывает поверхности с нулевой кривизной по крайней мере в одном направлении во всех точках.

Покажем теперь, как векторы кручения влияют на форму внутренней области поверхности. Если показать, что векторы кручения определяют форму изопараметрической кривой $v = v_0$ на рис. 8.3, то можно утверждать, что эти векторы влияют на любую внутреннюю изопараметрическую кривую, а значит, и на всю внутреннюю область лоскута. Точки A и B определяются граничными кривыми – эрмитовыми кривыми, которые не зависят от векторов кручения. Следовательно, необходимо показать, что векторы касательных в точках A и B определяются векторами кручения. Сделаем это, продифференцировав уравнение (8.18) по параметру u:

$$\frac{\partial \mathbf{P}(u,v)}{\partial u} =$$

$$= \begin{bmatrix} \frac{\partial F_1(u)}{\partial u} & \frac{\partial F_2(u)}{\partial u} & \frac{\partial F_3(u)}{\partial u} & \frac{\partial F_4(u)}{\partial u} \end{bmatrix} \begin{bmatrix} \mathbf{P}(0,0) & \mathbf{P}(0,1) & \mathbf{P}_v(0,0) & \mathbf{P}_v(0,1) \\ \mathbf{P}(1,0) & \mathbf{P}(1,1) & \mathbf{P}_v(1,0) & \mathbf{P}_v(1,1) \\ \mathbf{P}_u(2,0) & \mathbf{P}_u(1,0) & \mathbf{P}_{uv}(0,0) & \mathbf{P}_{uv}(0,1) \\ \mathbf{P}_u(3,0) & \mathbf{P}_u(1,1) & \mathbf{P}_{uv}(1,0) & \mathbf{P}_{uv}(1,1) \end{bmatrix} \begin{bmatrix} F_1(v) \\ F_2(v) \\ F_3(v) \\ F_4(v) \end{bmatrix} =$$

$$= \begin{bmatrix} -6u + 6u^2 & 6u - 6u^2 & 1 - 4u + 3u^2 & -2u + 3u^2 \end{bmatrix} \begin{bmatrix} \mathbf{G}_1(v) \\ \mathbf{G}_2(v) \\ \mathbf{G}_3(v) \\ \mathbf{G}_4(v) \end{bmatrix}. \qquad (8.20)$$

Произведение второй и третьей матриц записано в виде вектора-столбца с элементами $\mathbf{G}_1(v)$, $\mathbf{G}_2(v)$, $\mathbf{G}_3(v)$, $\mathbf{G}_4(v)$. Чтобы вычислить производную по u в точке A, необходимо подставить в уравнение (8.20) значения $u = 0$, $v = v_0$. Подстановка $u = 0$ в вектор-строку делает её равной [0 0 1 0], благодаря чему получается, что $\mathbf{P}_u(A) = \mathbf{G}_3(v_0)$. Поскольку $\mathbf{G}_3(v_0)$ определяется векторами

$\mathbf{P}_u(0,0)$, $\mathbf{P}_u(0,1)$, $\mathbf{P}_{uv}(0,0)$ и $\mathbf{P}_{uv}(0,1)$, векторы кручения $\mathbf{P}_{uv}(0,0)$ и $\mathbf{P}_{uv}(0,1)$ определяют $\mathbf{P}_u(A)$.

Подставим $u = 1$, $v = v_0$ в уравнение (8.20), чтобы выяснить, от чего зависит $\mathbf{P}_u(B)$:

$$P_u(B) = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{G}_1(v_0) \\ \mathbf{G}_2(v_0) \\ \mathbf{G}_3(v_0) \\ \mathbf{G}_4(v_0) \end{bmatrix} =$$

$$= \mathbf{G}_4(v_0) = \begin{bmatrix} \mathbf{P}_u(1,0) & \mathbf{P}_u(1,1) & \mathbf{P}_{uv}(1,0) & \mathbf{P}_{uv}(1,1) \end{bmatrix} \begin{bmatrix} F_1(v_0) \\ F_2(v_0) \\ F_3(v_0) \\ F_4(v_0) \end{bmatrix}. \quad (8.21)$$

Таким образом, вектор касательной в точке B зависит от векторов кручения $\mathbf{P}_{uv}(1,0)$ и $\mathbf{P}_{uv}(1,1)$. Отсюда следует вывод, что векторы кручения влияют на любую внутреннюю изопараметрическую кривую, а следовательно, определяют форму бикубического лоскута.

8.5. Поверхность Безье

Можно расширить концепцию кривой Безье, определяемой задающим многоугольником, на одно измерение, в результате чего получится поверхность Безье, определяемая задающим многогранником. Уравнение поверхности Безье выглядит следующим образом:

$$\mathbf{P}(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} \mathbf{P}_{i,j} B_{i,n}(u) B_{j,m}(v) \quad (0 \le u \le 1, 0 \le v \le 1), \quad (8.22)$$

где $\mathbf{P}_{i,j}$ – радиус-векторы задающих точек, находящихся в вершинах задающего многогранника (рис. 8.4), а $B_{i,n}$ и $B_{j,m}$ – функции сопряжения, обычные для кривых Безье. Таким образом, степень уравнения поверхности по u и v определяется количеством задающих точек в соответствующих направлениях.

Уравнение (8.22) можно раскрыть, записав сумму по j в явном виде:

$$\mathbf{P}(u,v) = \sum_{i=0}^{n} \left[\mathbf{P}_{i,0} B_{0,m}(v) + \mathbf{P}_{i,1} B_{1,m}(v) + \ldots + \mathbf{P}_{i,m} B_{m,m}(v) \right] B_{i,n}(u). \quad (8.23)$$

Уравнение (8.23) иллюстрирует, что поверхность Безье получается сопряжением $n + 1$ кривых Безье, каждая из которых определяется задающими точками $\mathrm{P}_{i,0}$, $\mathrm{P}_{i,1}$, $\mathrm{P}_{i,2}$, … $\mathrm{P}_{i,m}$, сопрягаемыми функциями $B_{i,n}(u)$. Можно показать, что та же поверхность Безье получается при сопряжении $m + 1$ кривых Безье, определяемых задающими точками $\mathrm{P}_{0,j}$, $\mathrm{P}_{1,j}$, $\mathrm{P}_{2,j}$, … $\mathrm{P}_{n,j}$, и функциями $B_{j,m}(v)$. *Итак, поверхность Безье получается, если задающие точки кривой Безье заменить кривыми Безье.*

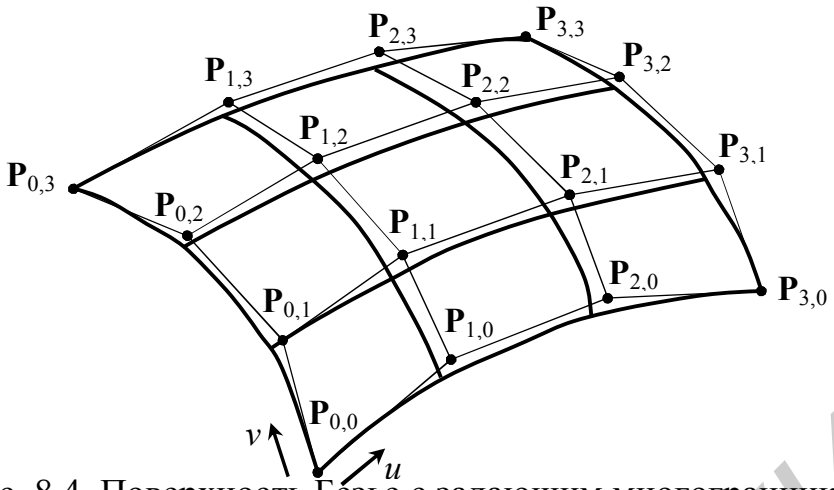


Рис. 8.4. Поверхность Безье с задающим многогранником

Как известно, степень поверхности Безье определяется количеством задающих точек. Уравнения поверхностей высоких степеней страдают теми же недостатками, что и уравнения кривых высоких степеней, поэтому при моделировании поверхностей обычно используются поверхности Безье степени 3 по *u* и *v*, точно так же как при моделировании кривых использовались кривые Безье степени 3. При моделировании сложной поверхности создают несколько поверхностей Безье третьей степени и соединяют их друг с другом. Поверхности должны соединяться таким образом, чтобы обеспечивалась непрерывность на границе, по которой осуществляется соединение. Это достигается наложением ограничений на задающие точки, расположенные слева и справа от границы. Ограничение состоит в том, что эти точки должны лежать на прямой линии, проходящей через задающую точку, лежащую на общей границе (рис. 8.5). Если это требование выполняется, первая производная оказывается непрерывной во всех точках границы.

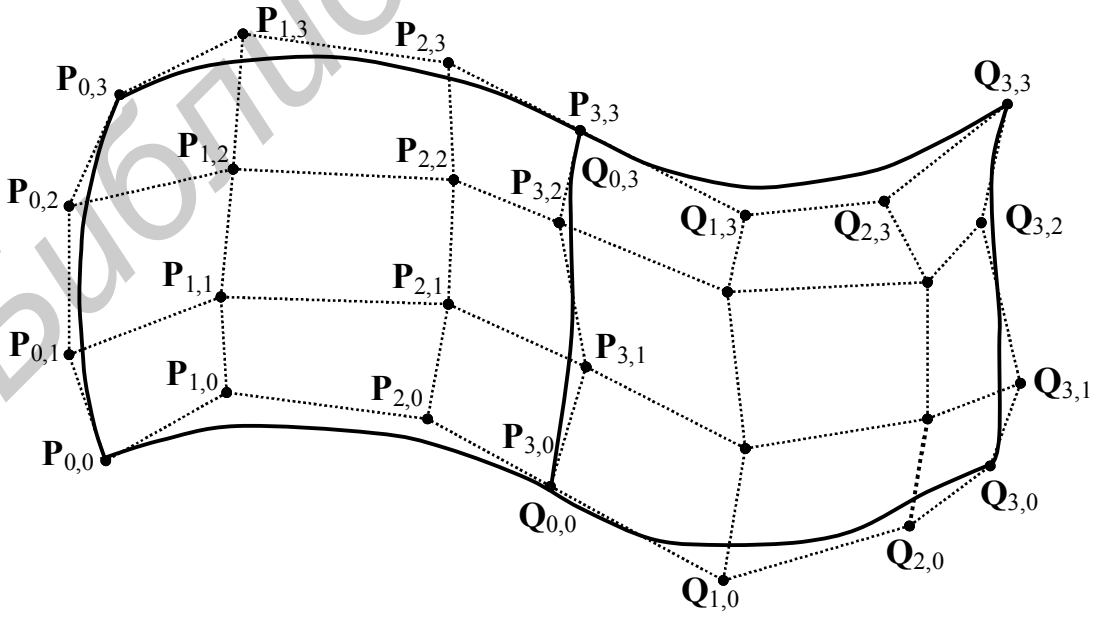


Рис. 8.5. Объединение лоскутов Безье

8.6. *B*-сплайновая поверхность

Подобно тому, как от уравнения кривой Безье был осуществлён переход к уравнению поверхности Безье, можно перейти и от уравнения *B*-сплайна к уравнению *B*-сплайновой поверхности:

$$\mathbf{P}(u,v)=\sum_{i=0}^{n}\sum_{j=0}^{m}\mathbf{P}_{i,j}N_{i,k}(u)N_{j,l}(v)\ \ (s_{k-1}\le u\le s_{n+1}, t_{l-1}\le v\le t_{m+1}), \qquad (8.24)$$

где $\mathbf{P}_{i,j}$ – задающие точки, расположенные в вершинах задающего многогранника, как и для поверхности Безье, а $N_{i,k}(u)$ и $N_{j,l}(v)$ – функции сопряжения, используемые для построения *B*-сплайнов. Эти функции сопряжения определяются узловыми значениями s_0, s_1, …, s_{n+k}, и t_0, t_1, ..., t_{l+m} соответственно. Диапазоны параметров используются в определении *B*-сплайна, поскольку функции сопряжения $N_{i,k}(u)$ и $N_{j,l}(v)$ определены только на этих интервалах (см. раздел 7). Это верно как для периодических узлов, так и для непериодических. Далее будем рассматривать только непериодические узлы по тем же причинам, что и раньше. В этом случае функции сопряжения *B*-сплайна будут совпадать с функциями сопряжения поверхности Безье, если $k=n+1$, $l=m+1$. Поэтому можно утверждать, что поверхность Безье является лишь частным случаем *B*-сплайновой поверхности, а уравнение (8.24) описывает как поверхности Безье, так и *B*-сплайновые. Чаще всего порядки k и l принимаются равными 4, поскольку степени уравнений, описывающих поверхности, не должны превышать 3.

B-сплайновая поверхность с непериодическими узлами обладает свойствами, напоминающими поверхность Безье (например, четыре угла задающего многогранника лежат на поверхности, а граничные кривые представляют собой *B*-сплайны, определяемые соответствующими подмножествами задающих точек). Покажем, что значению параметра $u=0$ соответствует граничная кривая, являющаяся *B*-сплайном. Подстановка $u=0$ в уравнение (8.24) даёт

$$\mathbf{P}(0,v)=\sum_{j=0}^{m}\left[\sum_{i=0}^{n}\mathbf{P}_{i,j}N_{i,k}(u)\right]_{u=0}N_{j,l}(v)=\sum_{j=0}^{m}\mathbf{P}_{0,j}N_{j,l}. \qquad (8.25)$$

В соответствии с формулой (8.23), граничная кривая при $u=0$ является *B*-сплайном с задающими точками $\mathbf{P}_{0,0}$, $\mathbf{P}_{0,1}$, ..., $\mathbf{P}_{0,m}$. Аналогичным образом можно показать, что и остальные граничные кривые являются *B*-сплайнами, а их задающие точки являются крайними вершинами задающего многогранника.

8.7. Поверхность NURBS

Перейдя к однородным координатам задающих точек, из уравнения *B*-сплайна получено уравнение NURBS-кривой. Точно так же уравнение поверхности NURBS получается из *B*-сплайновой поверхности:

$$\mathbf{P}(u,v)=\frac{\sum_{i=0}^{n}\sum_{j=0}^{m}h_{i,j}\mathbf{P}_{i,j}N_{i,k}(u)N_{j,l}(v)}{\sum_{i=0}^{n}\sum_{j=0}^{m}h_{i,j}N_{i,k}(u)N_{j,l}(v)}\ (s_{k-1}\le u\le s_{n+1},\ t_{l-1}\le v\le t_{m+1}),\qquad(8.26)$$

где $\mathbf{P}_{i,j}$ – векторы задающих точек с компонентами x, y и z, а $h_{i,j}$ – однородные координаты задающих точек. Узловые значения и диапазоны значений параметров совпадают с указанными в формуле (8.24).

Уравнение (8.26) становится уравнением B-сплайновой поверхности при $h_{i,j}=1$, следовательно, уравнение B-сплайновой поверхности является частным случаем уравнения NURBS-поверхности. Поверхность NURBS к тому же позволяет точно описать квадратичные поверхности, такие как цилиндр, конус, сфера, параболоид и гиперболоид. Эти поверхности называются квадратичными, потому что их уравнения имеют степень 2 по u и v. Уравнение NURBS-поверхности часто используется для внутреннего представления квадратичных поверхностей в системах геометрического моделирования.

Проиллюстрируем использование уравнений NURBS-поверхности в системах объемного моделирования на примере поверхности, полученной трансляцией кривой (рис. 8.6). Предположим, что транслируемая кривая задана NURBS-уравнением. Это предположение не накладывает никаких ограничений, поскольку уравнение любой кривой из рассмотренных выше может быть преобразовано к форме NURBS. Пусть порядок этой кривой l, узловые значения t_p ($p = 0, 1, ..., m + l$), задающие точки $\mathbf{P}_j$ ($m + 1$ штук).

$$\mathbf{P}(v)=\frac{\sum_{j=0}^{m}h_j\mathbf{P}_jN_{j,l}(v)}{\sum_{j=0}^{m}h_jN_{j,l}(v)}\ (t_{l-1}\le v\le t_{m+1}).\qquad(8.27)$$

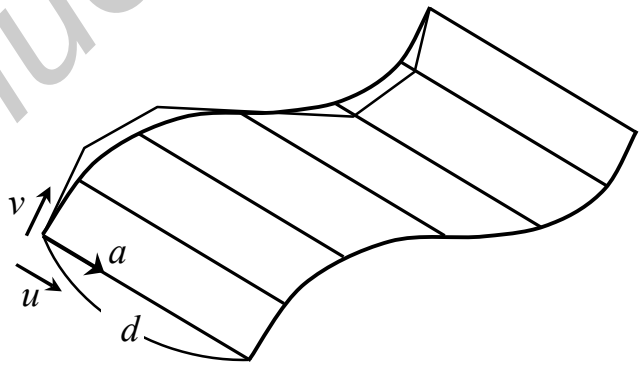


Рис. 8.6. Поверхность трансляции

Граничная кривая поверхности NURBS представляет собой NURBS-кривую с соответствующими задающими точками, которые являются крайними вершинами задающего многогранника поверхности. Порядок и узловые значения граничной кривой совпадают с теми же характеристиками поверхности в соответствующем направлении. Этим же свойством обладают

B-сплайновые поверхности. Следовательно, задающие точки, порядок и узловые значения поверхности в одном из направлений, совпадающем с направлением изменения параметра кривой, получаются из характеристик транслируемой кривой, поскольку она является одной из граничных кривых. В частности, задающими точками на одном из краев поверхности будут точки $\mathbf{P}_j$ (см. рис. 8.6). Далее, порядок поверхности в направлении v будет равен l, а узловые значения будут равны t_p (если направления параметров выбрать так, как показано на рис. 8.6).

Однако необходима информация о характеристиках поверхности в направлении u. Предполагается, что направление u совпадает с направлением трансляции (см. рис. 8.6), а значит, в этом направлении достаточно линейного уравнения. Следовательно, порядок NURBS-кривой в этом направлении будет равен 2 и задающих точек тоже будет две. Узловые значения в направлении u будут равны 0, 0, 1, 1, а наборов узловых точек будет два. Один набор, как уже отмечалось, получается из $\mathbf{P}_j$, а второй – трансляцией $\mathbf{P}_j$ на расстояние d в направлении трансляции. Однородные координаты для обоих наборов будут совпадать с h_j транслируемой кривой. Итак, координаты x, y и z задающих точек $\mathbf{P}_{i,j}$ и однородные координаты h_{jj} описываются следующими выражениями:

$$\mathbf{P}_{0,j} = \mathbf{P}_j;$$
$$\mathbf{P}_{1,j} = \mathbf{P}_j + d\mathbf{a};$$
$$h_{0,j} = h_{1,j} = h_j,$$

где d – расстояние трансляции, а $\mathbf{a}$ – единичный вектор в направлении трансляции.

Уравнение NURBS-поверхности может быть записано так, как показано ниже. Вычисление осуществляется подстановкой нужных значений параметров:

$$\mathbf{P}(u,v) = \frac{\sum_{i=0}^{1}\sum_{j=0}^{m} h_{i,j}\mathbf{P}_{i,j}N_{i,2}(u)N_{j,l}(v)}{\sum_{i=0}^{1}\sum_{j=0}^{m} h_{i,j}N_{i,2}(u)N_{j,l}(v)} \quad (0 \le u \le 1,\ t_{l-1} \le v \le t_{m+1}). \qquad (8.28)$$

Обратим внимание на то, что $N_{j,l}(v)$ в уравнении (8.28) определяется узловыми значениями t_p.

9. Структуры для хранения данных о 3D-объектах

Рассмотрим, каким образом и в каких форматах хранится математическое описание, отражающее однозначное описание объемных тел. Структуры данных, используемые для описания объемных тел, делятся на три типа.

Первая структура представляет собой дерево, описывающее историю применения булевских операций к примитивам. Журнал операций называется

конструктивным представлением объемной геометрии (*Constructive Solid Geometry – CSG representation*). Дерево называется *деревом CSG* (*GSG tree*).

Вторая структура содержит сведения о границах объема (вершинах, ребрах, гранях и их соединении друг с другом). Это представление называется граничным представлением (*boundary representation – B-rep*), а структура данных – *структурой B-rep* (*B-rep data structure*).

Третья структура представляет объем в виде комбинации элементарных объемов (например кубов).

9.1. Дерево CSG

Дерево CSG содержит историю применения булевских операций к примитивам. Рассмотрим тело, изображенное на рис. 9.1, *а*. Его историю булевских операций можно представить в виде дерева так, как показано на рис. 9.1, *б*. Это дерево может быть представлено взаимосвязанными элементами данных (рис. 9.1, *в*).

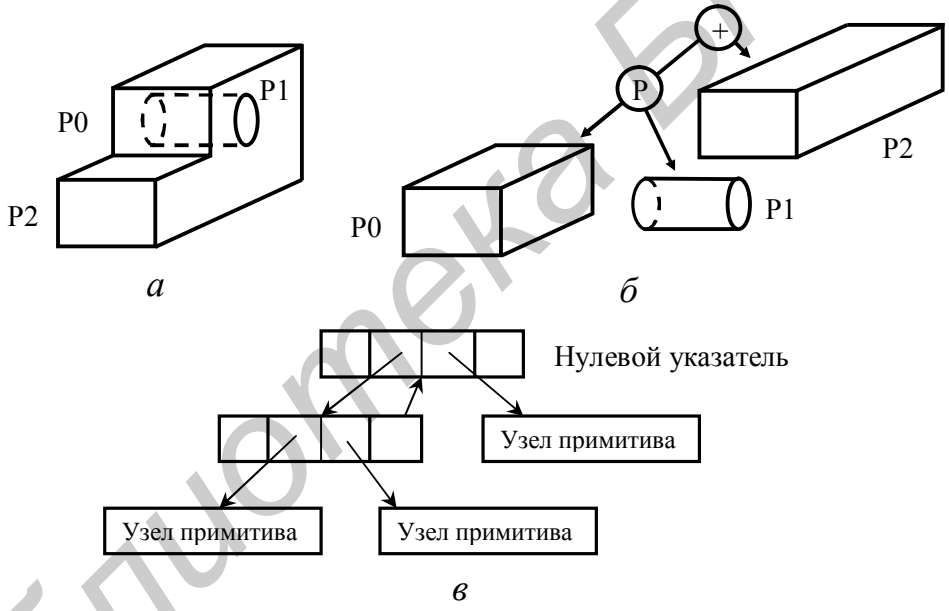


Рис. 9.1. Пример дерева CSG

Дерево CSG обладает следующими преимуществами:

– структура данных проста, а их представление компактно;

– внутренний объём тела однозначно отделен от внешнего. Примером некорректного объемного тела является тело с лишним ребром;

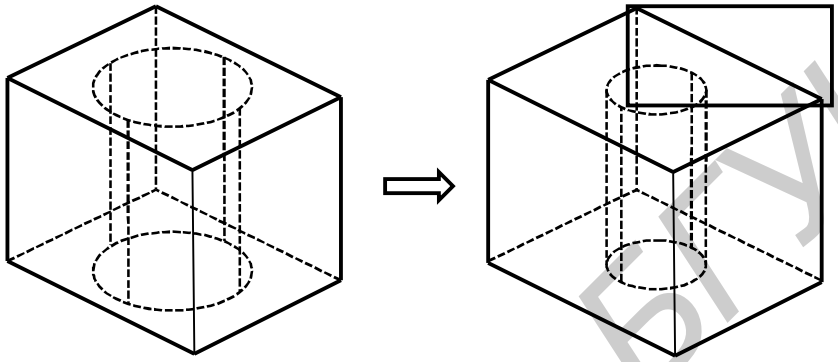
– представление CSG всегда может быть преобразовано к соответствующему представлению B-Rep. Это позволяет взаимодействовать с программами, ориентированными на использование B-Rep;

– легко реализуется параметрическое моделирование.

Есть у этого дерева и недостатки:

– в процессе моделирования может использоваться только история применения булевских операций, что жестко ограничивает диапазон моделируемых объектов. Более того, исключается использование удобных функций локального изменения, таких как поднятие и скругление;

– для получения сведений о граничных поверхностях, их рёбрах и связях между этими элементами из дерева CSG требуются сложные вычисления (в частности для отображения тел). Чтобы отобразить затушеванное изображение или чертеж объемного тела, нужно иметь информацию о гранях или вершинах этого тела, а получить эти сведения из дерева CSG очень непросто.



а *б*

Рис. 9.2. Изменение параметров тела

Из-за этих недостатков разработчики программ, основанных на представлении CSG, стараются добавить соответствующие сведения о границах. Такое комбинированное математическое представление называется гибридным и требует поддержания согласованности между двумя структурами данных.

9.2. Структура данных B-Rep

Границы объемных тел состоят из элементарных геометрических объектов: вершин, ребер и граней.

Грань – часть граничной поверхности, граница которой состоит из криволинейных сегментов, при пересечении которых происходит существенное изменение вектора нормали к поверхности. Криволинейные сегменты, ограничивающие грань, называются *рёбрами*. Точки, в которых встречаются соседние рёбра, называются *вершинами*.

В структуре данных B-Rep хранятся все эти элементы вместе со сведениями о том, как они соединены друг с другом. Одна из простейших структур данных приведена в табл. 9.1. Структура данных представляет объёмное тело, изображенное на рис. 9.3. В таблице граней хранится список ограничивающих рёбер для каждой грани. Последовательность рёбер для каждой грани дается обходом против часовой стрелки, если смотреть на тело снаружи. Благодаря тому что рёбра хранятся согласованно, вместе с каждой

гранью сохраняется информация о том, с какой стороны от неё находится внутренний объём тела. Другими словами, имея сведения о гранях, можно определить, где расположена конкретная точка: снаружи или внутри тела. Вершины, рёбра и грани, изображенные на рис. 9.3, нумеруются системой геометрического моделирования в произвольном порядке в момент сохранения сведений из табл. 9.1.

Таблица 9.1

Три таблицы представления B-Rep

Таблица граней	Таблица рёбер	Таблица вершин			
Грань	Рёбра	Ребро	Вершины	Вершина	Координаты
F_1	E_1, E_5, E_6	E_1	V_1, V_2	V_1	x_1, y_1, z_1
F_2	E_2, E_6, E_7	E_2	V_2, V_3	V_2	x_2, y_2, z_2
F_3	E_3, E_7, E_8	E_3	V_3, V_4	V_3	x_3, y_3, z_3
F_4	E_4, E_8, E_5	E_4	V_4, V_1	V_4	x_4, y_4, z_4
F_5	E_1, E_2, E_3, E_4	E_5	V_1, V_5	V_5	x_5, y_5, z_5
		E_6	V_2, V_5	V_6	x_6, y_6, z_6
		E_7	V_3, V_5		
		E_8	V_4, V_5		

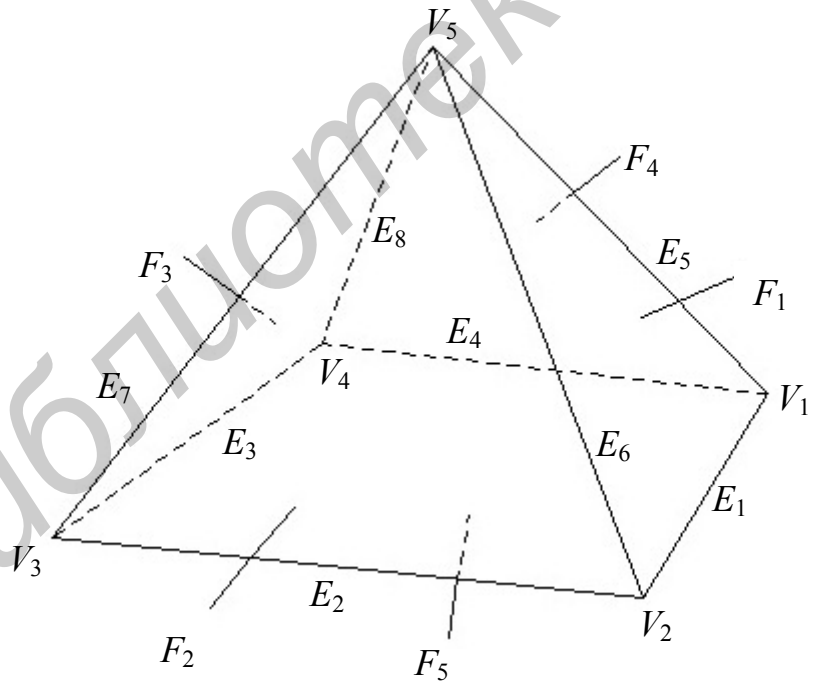


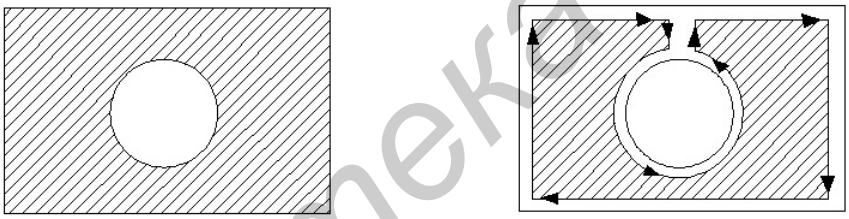
Рис. 9.3. Объёмное тело, сохраняемое в табл. 9.1

В каждой строке таблицы ребер хранятся вершины, находящиеся на концах соответствующего ребра, а в строках таблицы вершин хранятся координаты всех вершин. Эти координаты обычно определяются в модельной системе координат, связанной с данным телом. Если убрать отсюда таблицу

граней, эту структуру данных можно будет использовать для хранения форм, созданных в системах каркасного моделирования. Структура данных для каркасной модели может использоваться в качестве базовой для систем автоматизированной разработки чертежей, если допустить указание двумерных координат для точек. Структура данных B-Rep выглядит очень простой и компактной. Однако она не используется в развитых системах твердотельного моделирования из-за следующих недостатков:

– B-Rep ориентирована на хранение плоских многогранников. Если потребуется сохранить данные о теле с криволинейными гранями и ребрами, строки таблиц граней и ребер придется изменять таким образом, чтобы в них можно было включить уравнения поверхности и кривой;

– грань с внутренними и внешними границами (рис. 9.4, *а)* не может быть сохранена в таблице граней, поскольку для нее нужно два списка ребер вместо одного. Такие грани появляются, например, при моделировании объемных тел со сквозными отверстиями. Простым решением этой проблемы является добавление ребра, соединяющего внешнюю и внутреннюю границы (рис. 9.4, *б)*. В этом случае два списка вершин могут быть объединены. Соединительное ребро называется *мостиком*, или *перемычкой (bridge edge)*, и попадает в список ребер в двух экземплярах;



а *б*

Рис. 9.4. Поверхность с двумя границами и метод их обхода

– количество рёбер у разных граней может быть различно (см. табл. 9.1). Невозможно заранее определить количество столбцов, необходимых для конкретной грани, поскольку это количество может меняться в процессе моделирования, а работа с таблицей переменного размера создаёт неудобства;

– получать сведения о связности непосредственно из данных, сохранённых в трёх таблицах, может быть довольно утомительно. Представим себе поиск двух граней с общим ребром в случае граничного представления тела в трёх таблицах. Для этого придётся просмотреть всю таблицу граней, чтобы найти строки, в которых присутствует нужное ребро. Если нужно найти все рёбра, соединяющиеся в конкретной вершине, опять-таки придется просматривать всю таблицу рёбер. При больших размерах таблиц поиск в них становится крайне неэффективным.

9.3. Структура декомпозиционной модели

Объёмная модель может быть приближенно представлена в виде совокупности простых тел, например кубов. Такая модель называется *декомпозиционной (decomposition model).* Можно предложить много декомпозиционных моделей описания одного и того же тела. Модель включает в себя простейшее тело и метод объединения в совокупность. К типичным декомпозиционным моделям с соответствующими структурами данных относятся воксельное представление, представление октантного дерева и ячеечное представление.

9.4. Воксельное представление

Воксельное представление (voxel representation) объёмного тела – это просто трёхмерный аналог растрового представления плоской фигуры. Чтобы рассказать о воксельном представлении, придётся вспомнить процедуру получения растрового изображения. Растровое изображение двухмерного объекта формируется следующим образом. Сначала создаётся квадрат, размер которого соответствует интересующей нас области двухмерного пространства. Затем квадрат делится на много маленьких квадратиков путём нанесения на него линий сетки. Расстояние между линиями сетки определяется желаемой точностью растрового представления. Другими словами, если это расстояние будет очень маленьким, то растровое изображение будет очень точно воспроизводить форму исходного двумерного объекта. В противном случае получится лишь грубое приближение. Квадрат, содержащий много маленьких квадратиков, представляется в компьютере в виде двухмерного массива, количество элементов в котором совпадает с количеством квадратиков. Наконец, большой квадрат накладывается на двухмерный объект, и элементы массива, соответствующие квадратикам, находящимся над объектом, получают значение 1, а остальные элементы – значение 0. Получившийся массив нулей и единиц становится растровым представлением двухмерного объекта.

Воксельное представление объёмного тела получается при помощи той же процедуры, что и растровое представление. Однако начинается она не с большого квадрата и маленьких квадратиков, а с большого куба и маленьких кубиков, называемых *вокселами.* Деление на вокселы осуществляется сеткой плоскостей, расположенных на равном расстоянии друг от друга перпендикулярно осям *x, y* и *z.* Исходный куб представляется в виде трёхмерного массива, количество элементов которого совпадает с количеством кубиков, и каждому элементу массива присваивается значение 0 или 1 в зависимости от положения элемента в теле. Несмотря на то что эта процедура практически идентична процедуре формирования растра, проверка пересечения тела и кубика требует больших вычислений, чем аналогичная двухмерная задача. Воксельное представление объёмного тела, имеющего форму бублика, демонстрирует рис. 9.5.

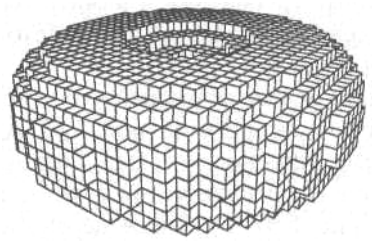


Рис. 9.5. Визуализация воксельного представления

Воксельное представление обладает следующими преимуществами:

– позволяет точно или по крайней мере приблизительно описать объёмное тело совершенно произвольной формы. Например, модели человеческих костей и органов часто являются вексельными представлениями данных цифровой томографии. Моделировать такие формы при помощи обычных функций очень сложно. Даже применение сложных процедур всё равно не позволяет построить точную модель;

– позволяет с лёгкостью рассчитывать такие параметры тела, как масса и моменты инерции. Расчёт осуществляется путём суммирования параметров отдельных вокселов. Также легко получить результат булевской операции. Для этого достаточно всего лишь применить булевскую операцию к целочисленным значениям соответствующих вокселов двух тел;

– хотя воксельное представление предназначено для описания тела в пространстве, оно автоматически описывает и пространство вне тела. Поэтому оно удобно для расчёта объёма полых структур и расчёта траекторий движения роботов, уклоняющихся от препятствий;

Есть у воксельного представления и некоторые недостатки:

– объём памяти, требуемый для хранения воксельного представления тела, резко возрастает с уменьшением размеров вокселов. Размер вокселов определяет точность приближения исходного тела, поэтому моделирование может потребовать предельного его снижения.

– воксельное представление по определению является приближенным описанием исходного тела. Поэтому систем твердотельного моделирования, в которых оно является основным математическим описанием объектов, довольно мало. Вокселы часто используются в качестве внешнего представления, повышающего эффективность вычислений.

9.5. Представление октантного дерева

Представление октантного дерева (octree representation) аналогично воксельному в том плане, что тело рассматривается как совокупность шестигранников (куб – это правильный шестигранник). Однако это представление предъявляет значительно менее серьёзные требования к памяти благодаря иной схеме деления пространства. В воксельном представлении исходный куб делится сеткой плоскостей, находящихся на равном расстоянии друг от друга

по осям *x, y* и *z*. В представлении октантного дерева исходный куб делится каждый раз на восемь равных кубов поперечными плоскостями (рис. 9.6, *а).* Объём маленького куба в восемь раз меньше объема исходного, отсюда и название представления. Более того, если кубики представить в виде узлов дерева, то от каждого узла будет отходить восемь ветвей (рис. 9.6, *б).* Такое дерево и называется *октантным.* Если бы каждый кубик всегда делился на восемь меньших вне зависимости от формы моделируемого тела, то результат был бы полностью аналогичен воксельному представлению с кубиками постоянного размера. Однако в представлении октантного дерева некоторые кубики делятся на восемь частей, тогда как другие кубики того же уровня остаются целыми. Определяется это положением кубиков по отношению к представляемому телу.

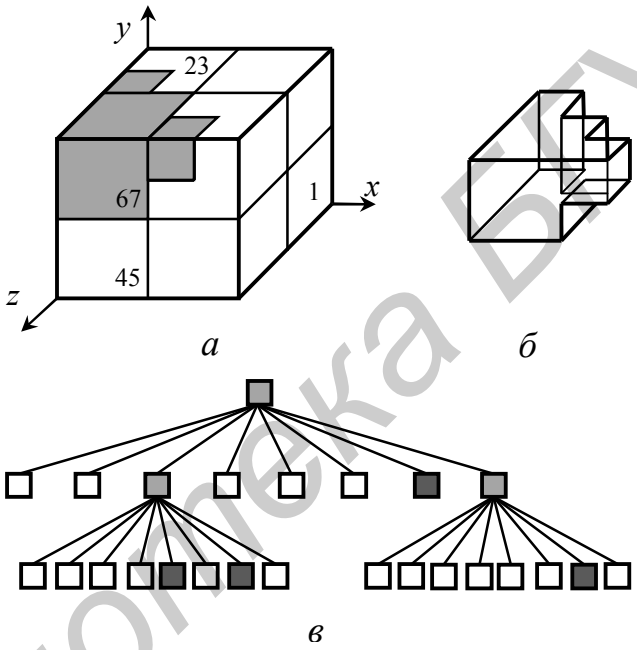


Рис. 9.6. Пример формирования октантного дерева

Процедура построения представления октантного дерева выглядит следующим образом. Сначала создается шестигранник, в который моделируемое тело помещается целиком. Этот шестигранник называется *корневым октантом (root octant)*. Затем корневой октант делится на восемь октантов, после чего анализируется их положение в пространстве по отношению к моделируемому телу. Если октант находится полностью внутри тела, он считается «черным»; если снаружи – «белым». Если же октант частично лежит внутри тела, а частично – снаружи, то он считается «серым» и делится на восемь октантов меньшего размера. Черные и белые октанты дальше не делятся. Процедура повторяется до тех пор, пока не будет достигнут заданный минимальный размер октанта. После этого октанты, окрашенные в черный цвет, считаются относящимися к исходному телу.

На рис. 9.6, *в* показано октантное дерево для тела с рис. 9.6, *б.* Количество октантов, под которые приходится отводить память, много меньше количества

вокселов для тела того же объема, поскольку белые и черные октанты дальше не делятся. Октантное дерево с рис. 9.6, *в* хранится в компьютере в виде структуры данных. Несмотря на относительно простой вид процедуры формирования октантного дерева на языке С, подпрограмма требует сложных геометрических вычислений, поскольку она определяет, где находится конкретный октант: внутри тела, снаружи его или на границе. Серые октанты преобразуются в чёрные после окончания процесса деления. Поэтому объём модели, полученной в результате применения этой процедуры, будет заведомо большим объёма исходного тела.

9.6. Ячеечное представление

Ячеечное представление (cell representation) – это ещё один метод представления объёмного тела в виде комбинации простых элементов, подобный воксельному представлению. Однако, как следует из названия представления, ячеечный метод не накладывает жёстких ограничений на форму этих элементов. Практически любое объёмное тело можно представить при помощи небольшого набора простых ячеек. Пример ячеечного представления представлен на рис. 9.7. Как видно из этого рисунка, формирование сетки конечных элементов для конечно-элементного анализа является частным случаем ячеечного представления.

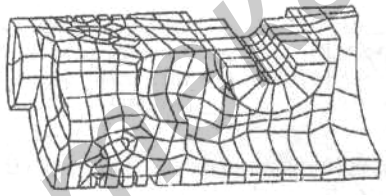


Рис. 9.7. Пример ячеечного представления

9.7. Булевы операторы

Из всех функций моделирования булевы операторы являются наиболее сложными с точки зрения реализации, однако они предоставляют наиболее широкие возможности пользователю системы моделирования. К булевым операциям относятся объединение, пересечение и разность объёмных тел. Результат операции сохраняется в структуре данных, характерной для используемой системы твердотельного моделирования. Если эта система основана на дереве CSG или декомпозиционном представлении, результат булевой операции легко будет представить в той же структуре. Дерево CSG результата любой булевой операции получается простым комбинированием деревьев исходных тел при помощи соответствующей операции булевой алгебры. То же относится и к декомпозиционной модели: там булевы операции применяются к пространственным элементам тел. Например, воксельное представление результата булевой операции может быть получено

применением соответствующей операции (побитовой булевой операции) к значениям вокселов двух тел, попадающих в одну и ту же точку пространства.

Если же система твердотельного моделирования использует структуру B-Rep, ситуация оказывается принципиально иной. В этом случае структуру B-Rep исходного тела приходится вычислять по структурам B-Rep исходных тел, к которым применяется булева операция. Этот процесс называется *вычислением границ (boundary evaluation).*

Алгоритм вычисления границ реализуется в три этапа. Воспользуемся плоским рисунком (рис. 9.8), но тот же подход применим и к трёхмерным телам. Сначала обозначим две грани (два тела в трёхмерном случае), к которым должна быть применена булева операция, буквами А и Б, а результат операции назовём буквой В (рис. 9.8, *а).* На первом этапе все рёбра А и Б, а также рёбра, получаемые пересечением этих граней, помещаются в единый пул рёбер. Подмножеством пула рёбер, очевидно, являются все рёбра грани В. На втором этапе все рёбра пула классифицируются по их положению относительно граней А и Б. Любое ребро может находиться, во-первых, внутри, на границе или снаружи грани а, во-вторых, внутри, на границе или снаружи грани Б (рис. 9.8, *б).* На третьем этапе ребра группируются по их относительному положению в соответствии с применяемой булевой операцией. Заметим, что для операции «А объединить с Б» нужно отбросить все рёбра, находящиеся «внутри А» и «внутри Б» (см. рис. 9.8, *б*). Аналогичным образом для операции «А пересечь с Б» отбрасываются рёбра, лежащие «вне А» и «вне Б». Собранные рёбра формируют грань. Для этого в структуру заносятся все необходимые сведения о вершинах, рёбрах, кольцах и прочих элементах. Для объёмных тел такой подход требует серьёзных геометрических вычислений, поскольку требуется определить, где именно находятся рёбра, а конструирование структуры B-Rep получившегося тела по собранным рёбрам представляет сложную топологическую задачу.

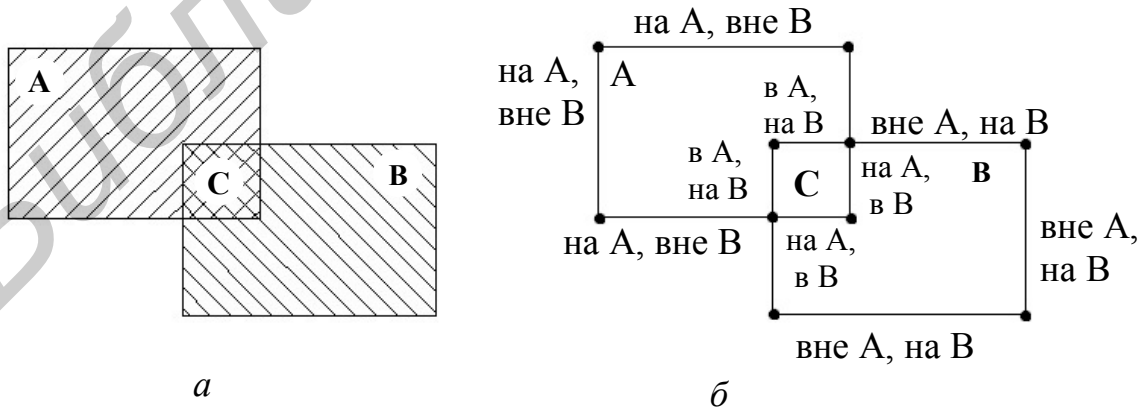


Рис. 9.8. Классификация рёбер

9.8. Расчёт объёмных параметров

Одним из преимуществ объемной модели является возможность расчёта объёмных параметров тела непосредственно исходя из его математического описания. Первые системы твердотельного моделирования использовались главным образом для визуализации формы объекта, так что расчёт объёмных параметров был одной из немногих инженерных функций, поддерживавшихся этими системами. К объёмным параметрам объекта относятся его объём, центр тяжести, моменты инерции и центробежные моменты инерции, которые определяются следующим образом.

Объем:

$$V = \iiint_V dV.$$

Центр тяжести:

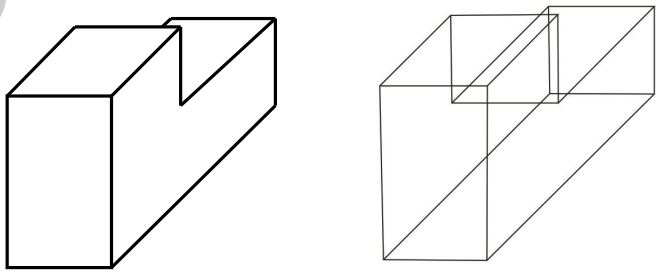
$$x_c = \frac{1}{V}\iiint_V x\, dV;$$

$$y_c = \frac{1}{V}\iiint_V y\, dV;$$

$$z_c = \frac{1}{V}\iiint_V z\, dV.$$

10. Удаление невидимых линий и поверхностей

Проекция на экран станет более наглядной, если будет содержать только видимые линии и поверхности. *Удаление невидимых линий* заключается в блокировании отображения отрезков, скрытых от наблюдателя, а *удаление невидимых поверхностей* есть то же самое по отношению к поверхностям (рис. 10.1, *б*).Эта процедура облегчает восприятие объекта.



а *б*

Рис. 10.1. Изображение: *а* – до удаления невидимых линий;
б – после удаления невидимых линий

Существует множество алгоритмов удаления невидимых линий и поверхностей. С точки зрения вычислительной эффективности наилучшим образом удаление реализуется посредством *z*-буфера – области памяти, аналогичной буферу кадра, в которой хранятся данные о Z-координатах объектов (проекциям пикселов). Координата Z является мерой расстояния от наблюдателя *до* объекта.

10.1. Алгоритм удаления невидимых граней

Данный алгоритм основан на том, что грань объекта может быть видимой только в том случае, если вектор внешней нормали к этой грани направлен в сторону наблюдателя. В противном случае грань будет невидима. Например, верхняя грань бруска, изображенного на рис. 10.2, считается видимой, если вектор внешней нормали **N** имеет положительную составляющую в направлении вектора **M**, проведённого из точки на грани к наблюдателю. Математически это записывается так: если $\mathbf{M}\cdot\mathbf{N} > 0$, поверхность видима;

если $\mathbf{M}\cdot\mathbf{N} = 0$, поверхность проецируется в отрезок;

если $\mathbf{M}\cdot\mathbf{N} < 0$, поверхность невидима.

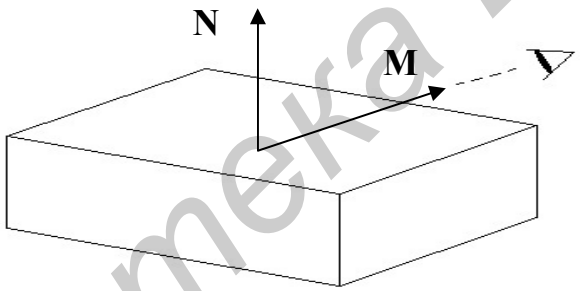


Рис. 10.2. Векторы, определяющие видимость грани

Этот алгоритм легко применить к объекту, ограниченному плоскими поверхностями, поскольку вектор нормали **N** постоянен в пределах поверхности. Однако к вогнутому объекту (вогнутым называется объект, если по крайней мере две его грани смыкаются под внутренним углом, большим 180°. Если меньшим 180°, то объект называется выпуклым.) алгоритм неприменим, поскольку грань, направленная к наблюдателю, может быть закрыта другой гранью того же объекта (см. рис. 10.2). Та же проблема возникает в случае нескольких выпуклых объектов, которые могут закрывать грани друг друга. Следовательно, алгоритм удаления невидимых граней применим только к одному выпуклому объекту. Более того, алгоритм неприменим к объектам, для которых вектор внешней нормали определяется неоднозначно (рис. 10.3).

Если поверхности объекта не являются плоскими, значение **N** будет меняться в пределах одной грани в зависимости от выбранной точки. Одновременно может меняться и знак произведения $\mathbf{M}\cdot\mathbf{N}$. Это означает, что у одной и той же грани будут как видимые, так и невидимые участки. Поэтому

грань должна быть разделена на две части вдоль кривой, на которой выполняется равенство $\mathbf{M}\cdot\mathbf{N}=0$. Эта кривая называется *силуэтной линией (silhouette line).* После разделения грани вдоль силуэтной линии знак $\mathbf{M}\cdot\mathbf{N}$ будет постоянным на каждой из частей грани. Процедура может показаться легкой, но рассчитать силуэтную линию очень сложно, а из-за этого теряется главное преимущество алгоритма удаления невидимых граней – простота реализации.

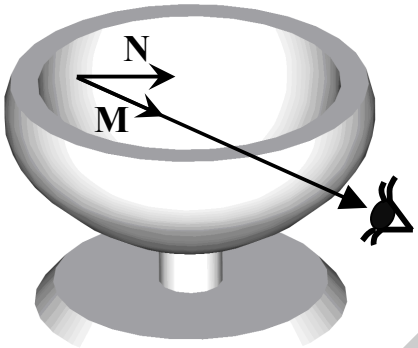


Рис. 10.3. Пример вогнутого объекта

После того как все грани классифицируются как видимые или невидимые, на экран выводятся ребра видимых граней, в результате чего получается рисунок без невидимых линий. Если же нужно получить рисунок без невидимых поверхностей, видимые поверхности заливаются выбранными цветами.

10.2. Алгоритм художника

Основной принцип *алгоритма сортировки по глубине,* или *алгоритма художника*, можно представить так. Поверхности объектов сортируются по удалённости от наблюдателя и заполняются соответствующими цветами, начиная с самой дальней. В результате закрашивания поверхностей дальние поверхности автоматически скрываются ближними, если они проецируются в одну и ту же область экрана. Расстояние от поверхности до наблюдателя определяется *z*-координатами точек на этой поверхности в наблюдательской системе координат. Точка с большей координатой Z_V считается находящейся ближе к наблюдателю. Значит, остается только сравнить координаты Z_P всех поверхностей и отобразить их, начиная с поверхности с наименьшим значением Z_V.

Легко сравнивать координаты Z_V поверхностей, если максимальное значение Z_V одной из них меньше минимального значения Z_V другой. Однако в большинстве случаев диапазон значений Z_V точек одной поверхности перекрывается с диапазоном Z_V другой поверхности. Неоднозначности можно избежать, разбивая все поверхности на отдельные части до тех пор, пока диапазоны Z_V не перестанут перекрываться. Есть и другой, более простой в реализации способ решения проблемы. Все поверхности преобразуются в

наборы небольших треугольников таким образом, чтобы диапазоны Z_V разных треугольников не перекрывались один с другим, после чего треугольники окрашиваются в соответствующие цвета в нужной последовательности. Чем меньше размер треугольников, тем меньше шансов, что их диапазоны Z_V перекроются. Такое приближение поверхности объекта называется *триангуляцией (triangulation)*, или *фасетированием.*

10.3. Алгоритм удаления невидимых линий

Алгоритм сортировки по глубине используется для удаления невидимых *поверхностей*. Алгоритм невидимых *граней* позволяет построить рисунок со скрытыми линиями, но имеет множество ограничений в общем случае. При применении алгоритма невидимых граней к множеству объектов удалено будет лишь около 50 % невидимых линий. Поэтому нужен алгоритм, который удалял бы все невидимые линии независимо от количества объектов, их выпуклости и наличия криволинейных поверхностей.

Один из таких алгоритмов действует следующим образом. Для каждого ребра (ребро – кривая пересечения соседних поверхностей, ограничивающих внутренний объём объекта) каждого объекта производится проверка, не закрыто ли оно гранями (грани – поверхности, ограничивающие объем объекта) каких-либо объектов. Закрытые части рёбер последовательно исключаются до тех пор, пока не останется непроверенных поверхностей. Оставшиеся части всех рёбер выводятся на экран.

Реализация алгоритма включает несколько этапов.

1. Поверхности, направленные к наблюдателю, выделяются из всех остальных в отдельную группу при помощи алгоритма невидимых граней. Выделенные поверхности сохраняются в массив FACE-TABLE. Грани, направленные от наблюдателя, учитывать не требуется, поскольку они сами по себе скрыты, а потому не скрывают ребра других граней. Для каждой грани сохраняется максимальное и минимальное значение Z_V. Криволинейные поверхности разделяются по силуэтным линиям (как в алгоритме невидимых граней), а видимые части этих поверхностей также сохраняются в массиве FACE-TABLE вместе с плоскими гранями.

2. Рёбра граней из массива FACE-TABLE выделяются из всех прочих ребер и собираются в отдельный список. Ребра других граней, не входящих в FACE-TABLE, можно не рассматривать, поскольку они невидимы. Затем для каждого ребра из списка производится проверка, не закрывается ли это ребро гранью из FACE-TABLE.

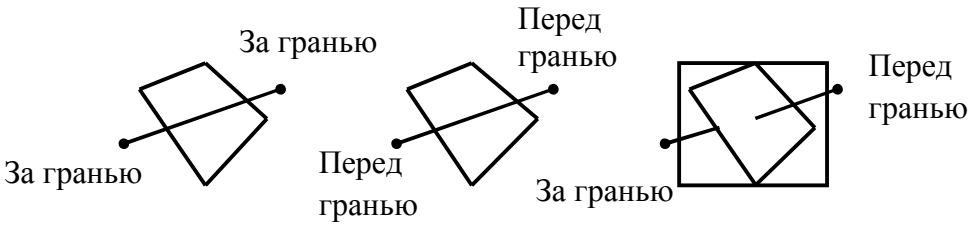


Рис. 10.4. Три возможных положения грани и ребра

3. Скрытие ребра гранью можно обнаружить, сравнивая диапазоны значений Z, ребра и грани. Возможны три случая (рис. 10.4). В случае рис. 10.4, *а* все значения Z_V ребра меньше минимального значения Z_v грани, то есть грань находится перед ребром. В случае рис. 10.4, *б* значения Z_V ребра больше максимального значения Z_V грани, то есть грань находится за ребром. В случае рис. 10.4, *в* диапазоны значений Z_V грани и ребра перекрываются, то есть часть ребра находится за гранью, а другая часть – перед ней. Если ребро находится перед проверяемой гранью, из массива FACE-TABLE выбирается следующая грань и ребро сравнивается уже с ней. Если ребро оказывается за гранью, или проходит её насквозь, приходится выполнять дополнительное действие.

4. Ребро и грань проецируются на экран, после чего производится проверка перекрытия проекций. Если перекрытия нет, из этого следует, что ребро не закрывает проверяемую грань. Из массива FACE-TABLE выбирается следующая грань и проверяется согласно пункту 3. Если проекции перекрываются, ребро делится на две части в той точке, где она проходит сквозь проверяемую грань (рис. 10.5). Закрытая часть ребра отбрасывается, а видимые части добавляются в список. Затем пункт 3 повторяется для новых элементов списка. Исходное ребро из списка удаляется.

5. Рёбра, прошедшие проверку со всеми гранями из FACE-TABLE, считаются видимыми и выводятся на экран.

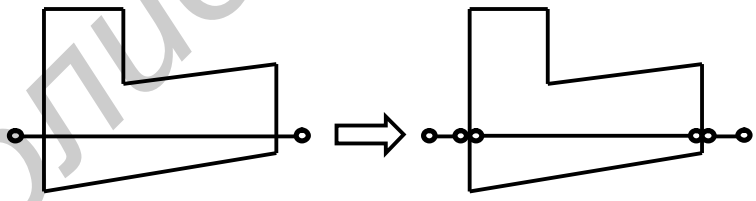


Рис. 10.5. Разбиение ребра

10.4. Метод *z*-буфера

Метод *z*-буфера основан на том же принципе, что и алгоритм сортировки по глубине: на любом участке экрана оказывается проекция элемента, расположенного ближе всех прочих к наблюдателю. Здесь под элементами понимаются точки, кривые или поверхности. Данный метод требует использования области памяти, называемой *z*-буфером. В этом буфере для каждого пиксела хранится значение координаты Z_V того элемента, проекция которого изображается данным пикселом. Как уже говорилось, значение Z_V (то

есть координата z в наблюдательской системе) есть мера удалённости объекта от наблюдателя. Объём z-буфера определяется количеством пикселов, для каждого из которых требуется сохранить вещественное число.

Грани, векторы нормали которых направлены от наблюдателя, невидимы для него, поэтому на экран проецируются только те грани, векторы нормали которых направлены к наблюдателю. Однако в отличие от метода сортировки по глубине в данном случае порядок проецирования значения не имеет. Причина станет очевидной при рассмотрении алгоритма работы.

1. Сначала проецируется произвольно выбранная поверхность, и в ячейки памяти z-буфера, соответствующие пикселам проекции поверхности, записываются значения координат Z_v точек поверхности, являющихся прообразами данных пикселов. Пикселы окрашиваются в цвет первой поверхности.

2. Затем проецируется следующая поверхность, и все неокрашенные пикселы, относящиеся к её проекции, окрашиваются в цвет второй поверхности. Если пикселы уже были окрашены, соответствующие им значения Z_v сравниваются со значениями Z_v точек текущей поверхности. Если сохраненное значение Z_v какого-то пиксела оказывается больше текущего (то есть точка на предыдущей поверхности ближе к наблюдателю, чем точка на текущей поверхности), цвет пиксела не меняется. В противном случае пиксел окрашивается в цвет текущей поверхности. Изначально z-буфер инициализируется координатами Z_v, соответствующими дальней плоскости, поэтому пикселы проекции первой поверхности автоматически окрашиваются в её цвет.

3. При повторении этой процедуры для всех имеющихся плоскостей, все пикселы экрана окрасятся в цвета ближайших поверхностей (рис. 10.6).

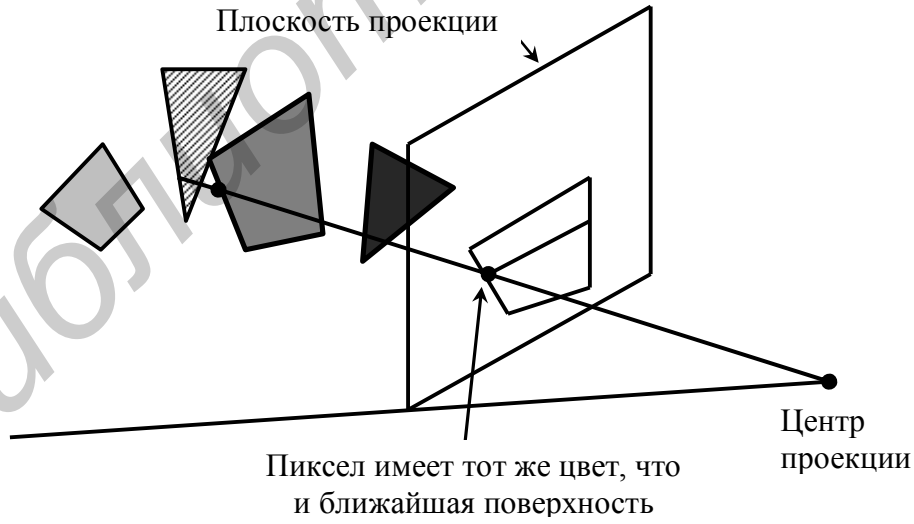


Рис. 10.6. Основы метода z-буфера

Как следует из предшествующего описания, метод z-буфера предназначен главным образом для удаления скрытых поверхностей, как и метод сортировки по глубине. Однако метод z-буфера с небольшими изменениями позволяет реализовать и построение рисунков со скрытыми линиями. Сначала все поверх-

ности проецируются на экран, причем пикселы окрашиваются в цвет фона (на экране ничего не появляется). При этом *z*-буфер заполняется правильными значениями Z_V. Затем на экран проецируются ребра поверхностей. При этом значения Z_V рёбер сравниваются со значениями Z_V ближайших к наблюдателю поверхностей, уже найденными на предыдущем этапе.

Окрашиваются только те пикселы, для которых новые значения Z_V больше исходных. Таким образом, участки ребер, скрытые поверхностями, на экране не появляются. Эта процедура даёт правильный рисунок без скрытых линий, но некоторые граничные линии могут оказаться слишком тонкими, потому что некоторые пикселы этих линий будут относиться к поверхностям, ограниченным соответствующими ребрами. Эту проблему легко решить, пододвинув весь объект поближе в момент проецирования рёбер.

Литература

1. Пашкевич, А. П. Основы систем автоматизированного проектирования : метод. пособие для студ. спец. 1-53 01 03 «Автоматическое управление в технических системах» и 1-53 01 07 «Информационные технологии и управление в технических системах» днев. формы обуч. / А. П. Пашкевич, О. А. Чумаков. – Минск : БГУИР, 2004. – 48 с.

2. Ли, К. Основы САПР (CAD/CAM/CAE) / К. Ли. – СПб. : Питер, 2004. – 560 с.

3. Кондаков, А. И. САПР технологических процессов / А. И. Кондаков. – Академия, 2007. – 272 с.

4. Ганин, Н. Б. Проектирование в системе КОМПАС-3D V11 (+ DVD-ROM) / Н. Б. Ганин. – ДМК Пресс, 2010. – 776 с.

5. Полещук, Н. Н. AutoLISP и Visual LISP в среде AutoCAD / Н. Н. Полещук. – СПб. : БХВ-Петербург, 2006. – 960 с.

6. Полещук, Н. Н. Visual LISP и секреты адаптации Auto CAD / Н. Н. Полещук. – СПб. : БХВ-Петербург, 2001. – 576 с.

7. Полещук, Н. Н. AutoCAD 2008 / Н. Н. Полещук. – СПб. : БХВ-Петербург, 2007. – 1158 с.

8. Смалюк, А. Ф. AutoCAD2000 / А. Ф. Смалюк, Д. В. Маккарчук, И. В. Кальник. – Минск : Кузьма, 2000. – 160 с.

9. Стинчкомб, К. Mechanical Desktop 6 : краткий визуальный курс / К. Стинчкомб. – Вильямс, 2003. – 272 с.

10. Кудрявцев, Е. М. Mechanical Desktop Power Pack. Основы работы в системе / Е. М. Кудрявцев. – М. : ДМК Пресс, 2001. – 544 с.

11. Кречко, Ю. А. Автокад 13: новые возможности : В 2-х ч. / Ю. А. Кречко, В. В. Полищук. – М. : ДИАЛОГ-МИФИ, 1996. – 288 с.

Св. план 2010, поз. 35

Учебное издание

Чумаков Олег Анатольевич
Стасевич Наталья Александровна

ОСНОВЫ СИСТЕМ АВТОМАТИЗИРОВАННОГО ПРОЕКТИРОВАНИЯ

УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ

Редактор *Т. П. Андрейченко*
Корректор *А. В. Бас*

Подписано в печать 18.07.2012. Формат 60x84 1/16. Бумага офсетная. Гарнитура «Таймс».
Отпечатано на ризографе. Усл. печ. л. 5,7. Уч.-изд. л. 5,4. Тираж 100 экз. Заказ 547.

Издатель и полиграфическое исполнение: учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники»
ЛИ №02330/0494371 от 16.03.2009. ЛП №02330/0494175 от 03.04.2009.
220013, Минск, П. Бровки, 6