Министерство образования и науки
Российской Федерации

Государственное бюджетное образовательное
учреждение высшего профессионального образования
Томский государственный университет
систем управления и радиоэлектроники

ЛЮКШИН Б.А.

КОПЬЮТЕРНАЯ ГРАФИКА

УЧЕБНОЕ ПОСОБИЕ

Томск
ТУСУР
2012

АННОТАЦИЯ

В учебном пособии изложены теоретические вопросы компьютерной графики и ее базовых понятий. Приведены краткая справка об истории становления дисциплины. Схематически приведены сведения об одной из распространенных графических систем Автокад, о реализованных в этой системе основных идеях, которые используются в большинстве других графических редакторов.

Значительное внимание уделено математическим вопросам компьютерной графики: базису преобразований координат точки, матричной форме записи этих преобразований. Приведены сведения об особенностях проекций гладких поверхностей. Даны понятия сглаживающих и интерполирующих кривых, рассмотрены примеры построения таких объектов с помощью полинома Лагранжа, кубических и бикубических сплайнов, базовых сплайнов, бета-сплайнов, кривых Безье.

Приведены сведения о растровых алгоритмах построения примитивов, закрашивания сплошных областей, построении реалистических изображений. Приведены сведения об алгоритмах триангуляции, построении линий уровня, об алгоритме плавающего (всплывающего) горизонта и т.д. Кратко рассмотрены задачи изображения поверхностей с использованием текстуры.

Предназначено для студентов всех специальностей, в учебные планы которых входят курсы компьютерной или инженерной и компьютерной графики

Оглавление

В В Е Д Е Н И Е

"Лучше один раз увидеть,
чем сто раз услышать"
(русская пословица)
"Одна картина стоит
тысячи рассказов"
(китайская пословица)

По некоторым оценкам средний человек от семидесяти до девяноста процентов информации получает с помощью зрения.

Визуальное (наглядное - в виде чертежа, схемы, плана, диаграммы и т.д.) представление информации может заменить длинное описание. Даже в обычном разговоре мы усиливаем «доходчивость» сведений до собеседника – зачастую неосознанно – жестикуляцией, помогая себе выразить свои мысли наглядно, воздействуя тем самым на зрение собеседника и закрепляя в его памяти сообщаемые сведения. Объясняя, как найти дорогу в незнакомом месте, мы чаще всего пытаемся сделать это с помощью примитивного чертежа или схемы.

Можно привести примеры использования наглядного изображения той или иной информации в технике, математике, экономике, социологии и других областях деятельности человека. Можно сказать, что почти всюду, где идет речь об обмене информацией, так или иначе используются визуальные ряды.

Язык графики понятен без перевода, достаточно знать лишь правила - и то в некоторых случаях, например, в инженерной графике, - по которым строится изображение. В этом отношении стандартные графические изображения – машиностроительные чертежи, электрические или электронные схемы – понятны зачастую даже без сопровождающих их надписей. Например, российскому специалисту в области радиоэлектроники понятна соответствующая схема любого устройства, даже если комментарий к ней отсутствует вообще или сделан, например, в виде иероглифов. Современные пиктограммы, особенно часто используемые при проведении спортивных мероприятий с участием спортсменов многих стран, также понятны без перевода, несмотря на простоту и условность в их изображениях. Во всяком случае, невозможно спутать символику соревнований по боксу с символикой плавания и т.д. Это же относится к знакам дорожного движения. Число аналогичных примеров можно множить. Не случайно первыми свидетельствами цивилизации являются наскальные изображения разного рода житейских ситуаций. По существу их тоже можно расценивать как доведение

информации – в художественной, пусть примитивной, форме – до современников и потомков.

К машинной графике в ее нынешнем понимании относится много различных изображений - реклама и мультфильмы на телевидении, набранные тексты с помощью персональных компьютеров (ПК) с разного рода украшениями и иллюстрациями, визитки и т.п. Компьютерная графика широко используется и в современном кино – как составная часть обычных игровых фильмов. Каждый из нас интуитивно может дать свое определение предмета машинной (или компьютерной) графики, отталкиваясь от своих представлений о ней. Можно говорить, что машинная графика – это новая отрасль информатики, и ее можно определить как науку о математическом моделировании геометрических форм и облика объектов, а также способов их визуального представления.

Три направления выделяются при обработке информации, связанной с построениями изображений на мониторе:

1. Распознавание образов.
2. Обработка изображений.
3. Машинная графика.

Основная задача в первом направлении – преобразование имеющегося изображения на формальный язык символов. Она решается с помощью совокупности методов, позволяющих либо получить описание изображения, либо отнести это изображение к определенному классу. Последние задачи возникают, например, при автоматизированной сортировке почты. Такие же задачи возникают в медицине, когда диагностика проводится с помощью томографии на основе отклонения изображения от нормы. Для проведения такого анализа изображение переводится в абстрактное описание – набор чисел, символов и т.д. Часто такой анализ проводится с помощью так называемой скелетизации объекта – когда вместо полного объекта рассматривается его основная, базовая составляющая. Это направление (computer vision - CV) можно

Рис. 1

схематически представить в виде рис. 1 (input – ввод изображения, description – описание).

Обработка изображений занимается задачами, в которых и входные, и выходные данные являются изображениями. Такие задачи возникают, например, при передаче изображений, когда необходимо убрать помехи или сжать объем передаваемой информации. При этом используется и переход от

полутоновых изображений к скелетным (каркасным). К задачам этого класса относятся и задачи синтеза полного изображения по набору изображение нескольких сечений объекта. Обработка изображений (image processing – IP) может быть представлена схемой рис. 2.

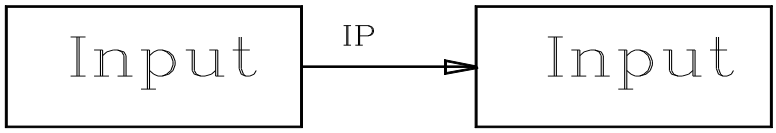


Рис. 2

Наконец, машинная (компьютерная) графика (computer graphics - CG) имеет дело с визуализацией изображений, изначально не имеющих изобразительной природы. В известном смысле здесь решается задача, обратная обработке изображений – из символьного или числового массива нужно восстановить изображение. Соответственно в схеме, отражающей компьютерную графику, прямоугольники на рис. 1 меняются местами.

Резкие границы между этими тремя задачами провести достаточно трудно. Поэтому можно нарисовать общую схему, вмещающую в себя все три описанных выше направления (см. рис. 3).

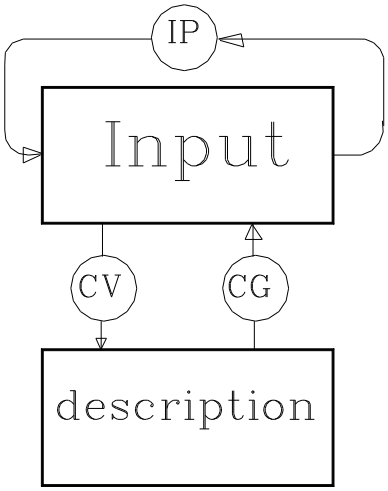


Рис. 3

В настоящее время в компьютерной графике, прошедшей иллюстративный отрезок пути своего развития, стремительно развиваются два основных направления – придание динамичности изображению и его реалистичности. Достижения компьютерной графики мы все видим в рекламных заставках на телевидении. В данном случае реклама является не только и не столько двигателем торговли, сколько стимулом развития компьютерной графики.

В дальнейшем мы будем ориентироваться на строгое определение понятия "машинная графика", даваемое Международной организацией по стандартизации (ISO – International Standard Organization):

машинная графика - это совокупность методов и средств для преобразования *данных* в графическую форму представления и из графической формы представления с помощью электронно-вычислительной машины (ЭВМ).

Следует заметить, что в соответствии с этим определением чертеж, показанный с помощью видеокамеры на экране и внешне схожий с чертежом, полученным с помощью графической системы, не является продуктом машинной графики.

В определении три основных составляющих:

1. Данные (“базы данных”).
2. ЭВМ.
3. Графическая форма представления.

Следует обратить внимание, что в этом определении речь идет не только о построении чертежей или схем с помощью компьютера и соответствующего программного обеспечения. В этом смысле «рисование и черчение» представляет собой хотя и важную, но только часть предмета компьютерной графики. Иначе сформулированное определение машинной графики можно изложить в виде утверждения: **машинная графика – это способ визуального представления информации с помощью компьютерных технологий.**

Когда речь идет о ***данных*** в определении выше, имеется в виду, что эти данные могут быть самого разного рода. С помощью компьютерной графики в наглядную (визуальную) форму можно перевести информацию об изменении цен, о популярности политического деятеля или артиста, о распределении мнений населения по какому-либо вопросу и т. д.

Разница между визуальными объектами, полученными другими способами и с помощью компьютерной графики, сводится прежде всего к тому, что источником входной информации для изображения на экране монитора в компьютерной графике является не физический объект (как, например, при съемке фото- или видеокамерой), а математическая модель, или, что еще точнее, некоторый числовой массив (файл) данных. Этот массив отображает структуру, свойства, взаимосвязи и отношения между элементами изображаемого объекта, а также между объектом и его окружением. Как правило, модели являются обобщенными, т.е. каждая такая математическая модель предназначена для описания некоторого класса изображений, а при воде конкретных значений параметров модели строится изображение. Например, при использовании команды **Отрезок** программа всегда работает одна и та же, но вводимая при обращении к этой команде информация определяет длину, расположение отрезка на рабочем поле, а также тип линии, с помощью которой строится изображение.

Следует с самого начала ясно понимать, что любая система машинной графики (или компьютерной графики) включает в себя две основные составляющие: аппаратное обеспечение (“железо”) и программное обеспечение (или “software” - “мягкий продукт”). Говорить о большей или меньшей важности какой-либо из этих составляющих не имеет смысла. Аналогию можно усмотреть у Лонгфелло в «Песни о Гайавате» (в переводе Бунина):

«…Муж с женой подобны луку,
Луку с крепкой тетивою.
Хоть она ему покорна,
Но сама его сгибает.
Порознь оба бесполезны…»

Создание систем машинной графики и ее развитие на современном этапе включает в себя эти обе составляющие - аппаратную и программную части. Программное обеспечение позволяет использовать в полную силу все возможности машины, а само формулирует те потребности, которые новое поколение персональных компьютеров должно обеспечивать в аппаратно-техническом отношении.

В настоящее время принято выделять три периода в истории машинной графики.

Начальный (1963-1970 гг.)

Создан первый пакет программ для обеспечения машинной графики – SKETCHPAD – и первый алгоритм удаления невидимых линий (Сазерленд). На экране с помощью электронного пучка относительно легко что-то нарисовать, но очень трудно стереть. Именно это достижение – получение возможности стереть часть изображения, а не все целиком – и дает отсчет началу компьютерной графики. Первые изображения на экранах запоминающих электронных трубок были получены, когда изображение строилось на слое люминофора, сплошь покрывающего поверхность экрана. В зависимости от состава люминофора время, которое держалось изображение на экране, составляло от нескольких секунд до нескольких часов. Изображение можно было построить сколько угодно четким. Технически дело сводилось к фокусировке электронного пучка. Позднее мы остановимся на растровом принципе построения изображений, принятом в настоящее время как стандарт, и тогда будут обсуждаться вопросы, почему на разных экранах мониторов четкость изображения получается разной. Так вот, наряду с этим достоинством – четкостью изображения – главный недостаток запоминающих трубок заключается в том, что технически стереть часть изображения на сплошном

слое люминофора практически невозможно. Лишь в 1965 г. был создан первый набор алгоритмов и пакет программ для создания рисунков на цифровом графопостроителе. К 1967г. разработаны первые программы, иллюстрирующие сложные поверхности, приемы освещения и затенения, алгоритмы устранения невидимых частей для векторных рисунков и изображений. Первые работы были направлены на развитие векторной графики, т.е. на рисование изображений с помощью отрезков. В этой связи следует отметить работы Брезенхема, которые внесли наибольший вклад в развитие векторной графики.

Период ученичества (1971-1978 гг.)

В этом периоде развиваются два направления – в соответствии с двумя составляющими компьютерных технологий, о которых шла речь выше – аппаратной и программной. Большое число работ направлено на развитие методов отображения пространственных форм и объектов, что сейчас принято называть трехмерной машинной графикой. Математическое моделирование изображений требует учета трехмерности пространства, взаимного расположения изображаемых предметов, источников освещения и наблюдателя.

В аппаратной составляющей улучшаются и развиваются базовые технические средства, позволяющие:

- разрабатывать методы сглаживания поверхностей;
- моделировать эффекты освещения;
- генерировать структуры и рельефы;
- подавлять алиайзинг (ступеньки) в изображении.

Здесь и далее термином «алиайзинг» называется эффект появления ступенек, когда строится, например, отрезок на экране, ориентированный близко к вертикали или горизонтали. Подробнее об этом речь пойдет при обсуждении растровых алгоритмов построения изображений.

Все это можно назвать просто улучшением визуального качества изображений.

В программной части выявляются основные концепции развития машинной графики, в соответствии с которыми:

- определяется базовое математическое обеспечение, не связанное с аппаратурой;
- проводится нормирование графических средств;
- стандартизуются графические средства.

Современный период (с 1979 г.)

На этом временном отрезке прогресс коснулся и аппаратной, и программной составляющих компьютерной графики за счет улучшения технико-эксплуатационных характеристик и развития специфических приложений. Широкое использование машинной графики в этот период прежде всего обусловлено появлением персональных компьютеров, доступных широкому кругу пользователей. В результате машинная графика стала инструментом не только для инженеров-исследователей в этой области, но и для специалистов других отраслей, не связанных непосредственно ни с вычислительной техникой, ни с программированием.

Можно выделить основные моменты в этом периоде, касающиеся развития компьютерной графики:

- разработка так называемых параллельных архитектур на основе БИС (больших интегральных схем) и СБИС (сверхбольших интегральных схем); это означает, что компьютер получает возможность проводить различные вычисления, связанные с построением и преобразованием изображений «параллельно» на нескольких процессорах;
- разработка методов подавления алиайзинга на основе использования элементов с жесткой логикой или средств микропрограммирования (т.е. аппаратными средствами);
- моделирование отражения света от объектов;
- использование видеотехники: микширование, врезка изображений, запись на видеокассеты и диски и т.д.
- разработка методов синтеза специфических объектов - многранников, сфер, функций двух переменных и т.д.
- преобразование цветовых пространств;
- классификация и сравнение алгоритмов удаления невидимых линий, алгоритмов синтеза изображений;

Один из важнейших признаков современного этапа: конечный пользователь является участником процесса решения своих задач с применением компьютера, без привлечения специалистов по программированию. Для современного пользователя компьютера, в том числе и применяющего в своей работе те или иные графические системы, это выглядит вполне естественно. Но еще два-три десятилетия назад вполне серьезно велась дискуссия на тему, можно ли работать с электронными вычислительными машинами человеку без специального образования, т.е. не профессиональному программисту. Современный этап развития компьютерных технологий решил эту проблему в пользу такой возможности.

На современном этапе одним из основных направлений является разработка принципов и методов формирования реалистичных изображений, которые можно наблюдать визуально или регистрировать оптическими, фотографическими или оптико-электронными устройствами. Такого рода потребности возникают в дизайне, машиностроительном и архитектурном

проектировании, в рекламе и т.д. Хотя в каждом конкретном случае цели преследуются разные (в рекламе это повышение эмоционального воздействия, в машиностроении и архитектуре – возможность оценки принятых решений и т.д.), методы решения этих задач являются достаточно сходными.

Современным направлением машинной графики является и создание динамических изображений. Такие задачи делятся на три основные группы. В первой из них получаемые изображения используются для наглядного представления процессов в научных исследованиях – химии, физике, механике и т.д. Во второй группе задач имитируются динамические ситуации в видеотренажерах разного рода. Наконец, задачи третьего рода должны решаться при построении двумерных и трехмерных изображений для телевидения и кино.

В трехмерной КГ основную роль играет так называемый метод трассирования лучей. В его основе лежит воспроизведение в математической форме хода лучей в реальных устройствах формирования изображений. Метод трассирования лучей иногда называется силовым – он практически не учитывает специфики построения изображений для объектов различной природы и сложности. Хотя это и приводит в некоторых случаях к избыточному объему вычислений, но, как правило, эта избыточность невелика при построении реалистичных изображений.

Можно выделить ряд этапов реализации любой технической или физической задачи – от постановки задачи (разработки технического задания) до анализа получаемых с помощью физического и математического моделирования результатов. Компьютерная графика (КГ) применяется и особенно удобна на этапе визуализации информации и при ее анализе.

* * *

Основные цели настоящего вводного курса компьютерной графики:

- изучение основных принципов построения систем компьютерной графики и графических пакетов;
- освоение конкретного графического пакета Автокад;
- знакомство с математическими вопросами КГ;
- знакомство с принципами технической реализации систем КГ, в том числе с растровыми алгоритмами построения изображений отрезков и других, более сложных объектов, а также с методами закрашивания сплошных областей;
- знакомство с принципами построения реалистичных изображений сложных сцен с учетом возможности взаимного загораживания объектов, удаления невидимых линий и граней многогранников;
- изучение способов построения линий уровня;
- знакомство с моделями освещения и закраски в КГ, построение изображений с учетом прозрачности, тени, фактуры и текстуры

поверхностей объектов, использование трассировки лучей и цветовых возможностей;
- знакомство с проблемами и алгоритмами распознавания образов, способами построения движущихся изображений (мультипликация, пакет 3D-Studio).

Контрольные вопросы к Введению

1. Назовите основные составляющие определения компьютерной графики, сформулируйте ваше понимание терминов «данные», «базы данных» в этом определении.
2. Назовите основные периоды в развитии компьютерной графики. В чем особенности современного этапа?
3. Сформулируйте Ваше понимание аппаратной и программной составляющей компьютерной графики.

I. Система Автокад (AutoCAD)

1.1 Вводные замечания

На английском языке сам термин AutoCAD (по-русски мы пишем и произносим как Автокад) представляет собой аббревиатуру (сокращение) от группы слов, последние из которых Computer Aided Design (CAD) с некоторой приблизительностью можно перевести как «создание с помощью компьютера». Таким образом, соответствующий пакет программ ориентирован исключительно на использование его в персональных компьютерах (ПК).

Все графические системы можно с некоторой условностью разделить на две основные группы:

- графические оболочки;
- встроенные редакторы.

В последнем случае речь идет об алгоритмических языках высокого уровня, а графические редакторы в них играют роль набора специализированных функций или процедур – это одна из составляющих предмета информатики. Далее мы о них говорить практически не будем.

Рассмотрим в этом разделе систему Автокад как мощный графический редактор («оболочку»), применяемый для автоматизации конструкторских и графических работ в машиностроении, строительстве и архитектуре, в радиоконструировании и т.д. Автокад представляет собой важнейшую составную часть многих систем автоматизированного проектирования (САПР).

Этот пакет разработан фирмой AutoDesk. Первая версия Автокада появилась в 1982 г. В настоящее время популярны Автокад – 2000, 2002, 2004. Очевидно, что процесс разработки все новых версий будет идти и впредь. Каждая новая версия является расширением предыдущей, и в нее включаются все ранее существовавшие возможности плюс новые прикладные программы, позволяющие строить изображения более коротким, наглядным и удобным путем. В связи с этим важно помнить, что любое изображение, полученное с помощью предыдущей версии, можно прочесть с помощью последующей, однако обратное утверждение в общем случае неверно.

Основное назначение системы Автокад:

- создание, редактирование, оформление и выдача на бумагу с помощью принтера графических изображений различных типов - схем, диаграмм, чертежей, иллюстраций.

Создание графического изображение – это и есть, собственно, процесс построения его с помощью конкретных команд пакета Автокад.

Редактированием называется внесение исправлений и изменений в рисунок или чертеж.

Под оформлением понимается нанесение надписей, простановка размеров, выбор цвета и типа линий, штриховка и т.д.

По некоторым оценкам, черчение составляет около 70 % затрат на проектирование. Хотя компьютерная графика не сводится к построению чертежей, как это уже отмечалось, но во многих случаях именно это приложение является важнейшим. В Автокаде процедура построения чертежа является центральной.

Если просто попробовать сделать даже простейший чертеж с помощью системы Автокад – или любой другой аналогичной – то затраты на получение изображения, особенно на первых порах, могут оказаться больше, чем с помощью карандаша и бумаги. По экспертным оценкам, по сравнению с ручным способом затраты времени при построении нового чертежа изменяются от сокращения на 20 % до увеличения вдвое. Некоторый выигрыш получается при больших объемах надписей, штриховок, простановок условных обозначений, вычерчивании рамок и «штампов» - особенно если эти элементы созданы заранее. Однако основные преимущества проявляются в следующих случаях:

- при построении серии чертежей со стандартными блоками и повторяющимися элементами;
- при внесении изменений за счет повышения качества выполнения чертежей и возможности просмотра любого элемента чертежа в произвольном масштабе;
- при копировании чертежа и повторном его использовании.

Таким образом, Автокад - прикладная система автоматизации чертежно-графических работ с удобными средствами исправления ошибок.

Основные преимущества Автокада перед другими графическими системами:

1. Очень широкий набор команд. Так, уже в 10-й версии Автокада насчитывалось более 420 команд, в более поздних версиях это число значительно больше.
2. Хорошее сочетание простых и сложных функций; так, не нужно переходить в специальные меню при пользовании такими разными по сложности командами, как «отрезок» и «масштаб» и т.п.
3. Интуитивное понимание назначения команд. В русскоязычных версиях Автокада команды не допускают различного толкования, и смысл команды «отрезок» не спутать с командой «сотри» и т.д. В версиях, начиная с 12-й, команды выведены на экран в виде так называемых кнопок, где смысл команды ясен из ее условного изображения на кнопке (пиктограммы). Название команды дублируется при обращении к соответствующей кнопке с изображением команды в виде словесной формулировки.

4. Широкий выбор периферийных устройств и их модификаций. Это означает, что Автокад позволяет использовать аппаратные возможности разных компьютеров (как это принято называть «разных конфигураций») в полном объеме.

5. Наличие различных языковых - русской и английской - версий.

6. Возможность изменять набор (меню) команд, т.е. составлять свое «персональное» меню как из стандартных команд, наиболее часто используемых пользователем, так и дополняя это меню своими командами.

7. Наличие встроенного языка программирования, что позволяет строить изображение не только привычным путем, с помощью «мыши», непосредственно работая перед экраном монитора, но и программным путем – составив программу рисования некоторого класса изображений, например, резьбового соединения, с меняющимися по вашему выбору параметрами.

8. Документированность системы – для системы Автокад существуют разнообразные пособия от простейших вводных для начинающих до сложных и полных описаний для профессиональных пользователей.

9. Большое число прикладных программ, разработанных различными фирмами.

Автокад представляет собой открытую систему, которой можно пользоваться в различных предметных областях, достаточно лишь использовать соответствующие программные блоки. Так, есть прикладные программы для машиностроителей, для строителей, для радиоинженеров и т.д. В связи с этой открытостью Автокад со временем расширяет область своего применения в самых разных приложениях.

Разумеется, можно осваивать и другие графические системы, тем более что в настоящее время нет в них недостатка. Но освоение Автокада имеет смысл еще и потому, что основные его черты, будучи стандартными, повторяются и в других графических системах. Кроме того, освоив любую версию Автокада, легко можно освоить более позднюю. Опыт показывает, что для этого требуется всего несколько часов.

1.2 Требования к аппаратному обеспечению Автокада

Как уже говорилось, исторически первые изображения на экранах были получены на запоминающих трубках. Луч (поток электронов) отклонялся управляющей системой в произвольном направлении, и для построения изображения надо было задать только это направление.

Изображение получалось непрерывным, поскольку слой люминофора является сплошным, а все линии, проводимые в любом направлении - гладкими. Качество чертежа на экране получалось весьма высоким. На мониторах такого типа легко рисуется изображение, но, как уже отмечалось выше, практически невозможно стереть ошибочно проведенную линию, можно убрать лишь все изображение целиком.

Современные экраны мониторов, как и в телевидении, рассчитаны на построение изображения на так называемой развертке, содержащей фиксированное количество строк (в стандартах телевидения разных стран это 650 или 800 строк). На цветных дисплеях изображение строится на некотором конечном наборе "точек", каждая из которых включает в себя три элемента, дающего при их "бомбардировке" потоком электронов одну из составляющих цвета (красный, синий, зеленый). Этот набор точек располагается по строкам и столбцам, ориентированным соответственно по горизонтали и вертикали, и образует так называемую растровую решетку, или растр – его называют техническим растром. С помощью этого растра и строится любое изображение на экране монитора.

Таким образом, если на старых экранах – запоминающих трубках – можно строить изображения сразу целыми линиями, то на современных мониторах - точками (или пикселами - от английских слов "picture element" - элемент рисунка; в англоязычной, обычно американской, литературе используется термин «pixel»).

Различают две системы графики - векторную и растровую. В первом случае изображение строится из элементов, как правило, представляющих собой некоторые линии или фигуры. Эти своего рода заготовки представляют собой как бы элементы известной игры «конструктор», из которых можно собирать разные модели. Примерами такого рода являются неоновая реклама, изображения цифр на электронных наручных часах и так далее. Отметим, что в Автокаде и других аналогичных графических системах эти элементы носят название примитивов. Для пользователя графической системы в этом отношении примитивы представляют собой объекты векторного типа, с помощью которых строятся изображения. Однако следует понимать, что на самом деле изображение – на экране монитора, на принтере – строится с помощью отдельных «точек», называемых пикселами или пикселями. Слово точка в предыдущем предложении взято в кавычки, так как сама точка не является таковой в математическом понимании (как объект, имеющий нулевые размеры во всех направлениях). Строго говоря, каждая такая точка имеет

конечные размеры и может состоять сама из набора отдельных элементов. Так, на современных цветных мониторах каждая «точка» представляет собой сочетание трех изображений основных цветов (красный, синий, зеленый), расположенных, как правило, в виде правильного треугольника. Размеры этого треугольника и представляют собой размеры элемента так называемого технического растра. Ниже в курсе мы дадим определение и математического растра.

Элементами растровых изображений являются пиксели, и в этом случае примерами могут служить информационные табло в аэропортах, на железнодорожных вокзалах и автовокзалах и т.д., где буквы и цифры составляются из отдельных «точек», представляющих собой в данном случае пиксели.

Современным направлением графики, в том числе и компьютерной, является растровая графика. Если даже для пользователя при работе с графической системой внешне дело выглядит так, что он работает элементами векторной графики (примитивами), на самом деле графическая система берет на себя «черновую» работу по переводу таких векторных объектов в растровую форму.

Для нормальной работы системы Автокада в 10-й версии было необходимо следующее компьютерное оборудование ("конфигурация" компьютера):

- IBM PC AT, оперативная память не менее 640 Кбайт;
- "винчестер" (жесткий диск) и один дисковод;
- сопроцессор I8087 или I80287;
- графический дисплей;
- операционная система MS DOS версии 2.0 и выше;
- мышь;
- принтер.

Эти требования по современным меркам более чем скромны. Для более поздних версий требуются значительно большие аппаратные возможности. Так, Автокад-2000 «не разворачивается», если объем оперативной памяти компьютера менее 32 Мбайт (сравните с 540 Кбайт для 10-й версии). Достаточная для учебных целей версия Автокада-10 занимает объем около 2.7 Мбайт, и ее в архивированном виде можно разместить на единственной стандартной дискете 3.5 дюйма. Автокад - 2000 в зависимости от конкретной версии занимает на жестком диске от 200 до 400 Мбайт.

При отсутствии какой-либо составляющей система Автокад работает либо не в полную силу, либо не работает совсем.

Система Автокад работает в стандартном варианте так, что на верхней (и при необходимости на левой) стороне экрана выводится так называемое экранное меню, а в нижней части экрана - строка для запросов и подсказок. Таким образом, Автокад по своему построению реализован как диалоговая

система. Набор команд размещается пользователем в той части экрана, где ему удобнее.

Экранное меню - перечень действий, обеспечиваемых системой в данный момент. Среди этих действий:

- вызов команды;
- высвечивание текстов или подсказок;
- переход в другое меню;
- выход из меню.

Существуют еще такие устройства, которые в настоящее время имеют преимущественно только исторический интерес:

- цифровой планшет - указатель ("цифровой карандаш") передвигается по полю планшета, информация снимается с помощью электромагнитного или емкостного датчика и после преобразования микропроцессором, встроенным в планшет, передается в центральный процессор;
- световое перо; подводится к нужной точке на экране и с помощью кнопки инициируется; неудобства очевидны – попробуйте подержать перед собой на весу руку хотя бы пару минут, чтобы в этом убедиться;
- джойстик – рукоятка в форме переключателя скорости в автомобиле – перемещает точку на экране монитора в направлении и со скоростью в зависимости от наклона ручки.

В качестве современного устройства ввода, пока еще не получившего широкого распространения, можно отметить так называемый сенсорный графический планшет (PenPartner) – устройство, внешне напоминающее коврик для мыши (сейчас эти технические термины понятны всем и не берутся в кавычки) и реагирующее на нажатие специальным карандашом. Понятно, что становится очень легко даже свой автограф перевести на экран дисплея.

1.3 Некоторые основные идеи КГ, реализованные в Автокаде

В этом разделе представлены некоторые реализованные создателями Автокада идеи, причем они представлены во всех его версиях.

1.3.1 Системы координат.

В Автокаде приняты и используются следующие основные системы координат.

Мировая система координат (МСК) - так называемая встроенная внутренняя система координат. Это обычная декартова система координат, горизонтальная ось X, вертикальная - У. Начало системы координат фиксировано в левом нижнем углу рабочего поля. При необходимости построения трехмерных изображений ось Z вводится по правилу правой руки. Эту систему координат пользователь изменить не может – ни положения начала отсчета, ни направления осей. Единственное, что можно сделать в этой системе координат – изменить единицы измерения, что приводит к изменению размеров рабочего поля.

При необходимости или для удобства работы пользователь может ввести собственную – **пользовательскую – систему координат** (ПСК). В этой системе начало координат и направления осей определяются пользователем. На экране в левом нижнем углу пиктограмма показывает, в какой системе координат в текущий момент времени – в мировой (МСК) или пользовательской (ПСК) – идет работа.

Положение точки на экране можно зафиксировать с помощью мыши, если не требуется большая точность. Но при необходимости можно использовать цифровой ввод с точностью до 14 значащих цифр. Координаты измеряются в некоторых условных единицах, а пользователь может сам установить масштаб. Еще раз масштаб можно менять при выводе изображения на печать.

1.3.2 Лимиты рисунка

Стороны прямоугольника, внутри которого можно что-либо рисовать на экране, называются лимитами (пределами) рисунка. Если попытаться строить что-либо за этими границами, Автокад при стандартных установках этому препятствует и выдает соответствующие сообщения.

В процессе черчения можно менять лимиты или отключить автоматический контроль соблюдения лимитов.

Когда нужно просмотреть детально часть рисунка, используется операция, называемая зуммированием. В большинстве пакетов она носит название «рамка» – при пользовании этой командой на экран выводится часть

изображения, попавшая в рамку. При этом координаты точек и расстояния между ними не меняются, изменяется только масштаб.

1.3.3 Слои

Одно из наиболее интересных изобретений компьютерной графики, используемое также и в Автокаде - так называемые слои. Их можно рассматривать как прозрачные листы (кальки) с изображениями отдельных частей рисунка. Каждый слой может нести свою информацию, при этом из итогового рисунка можно удалять отдельные слои. Так, в одном слое изображаются осевые линии, во втором – основные, на третьем – штриховка, на четвертом – размеры, и т.д. При этом границы чертежа, рисунка, единицы измерения, системы координат для всех слоев остаются едиными. При работе в Автокаде слой определяется своим именем, и в каждом слое можно использовать свои типы линий, цвет изображения. На экране можно наблюдать все, что нарисовано в разных слоях, или часть слоев отключить (“заморозить”), и в этом случае то, что на этих слоях нарисовано, на экран не выводится.

Число слоев практически не ограничено. Использование таких слоев удобно при создании проектов зданий и сооружений, при рисовании машиностроительных чертежей, радиотехнических схем и т.д. Так, в строительных чертежах можно последовательно в разных слоях рисовать и просматривать схемы электро- и радиопроводки, вентиляции, канализации, водопровода и т.д. При этом последовательное замораживание слоев позволяет анализировать эти схемы без перегруженности чертежа ненужными в данный момент сведениями.

1.3.4 Графические меню

С их помощью выбираются элементы изображений, которые трудно описать словами - например, тип штриховки, цвет линии или текста. При обращении к такому меню на экран выводятся образцы элементов изображений, и дело пользователя – выбрать нужный вариант.

1.3.5 Средства привязки

1. Можно включить так называемую сетку на экране, и перемещение курсора будет идти только по узлам сетки. Расстояние между узлами сетки устанавливается пользователем, причем можно делать разные шаги сетки по горизонтали и по вертикали.

2. При построении сложных рисунков можно использовать средства привязки типа ПЕРЕСЕЧЕНИЕ, НОРМАЛЬ, ЦЕНТР и т.д. Если сделать простейший чертеж с помощью мыши – например, изображение треугольника –

и просмотреть его затем в увеличенном масштабе, то в большинстве случаев соединение в конечной точке построения треугольника будет неточным. Отрезки либо пересекутся, либо окажутся несоединенными. Это связано с точностью позиционирования курсора на экране с помощью мыши – а она ограничена даже чисто технически. Использование средств привязки устраняет этот недостаток.

1.3.6 Рисунки-прототипы

Начиная новый рисунок, полезно заранее задать некоторые его параметры. Такими параметрами могут быть, например, лимиты рисунка, типы линий, единицы измерения координат и т.д.

Если не заказывать никакого прототипа, Автокад все равно предоставляет некоторый стандартный вариант с именем acad.dwg. Следует заметить, что все рисунки, создаваемые в Автокаде, хранятся в виде файлов с расширением .dwg (от английского слова drawing - рисунок).

Если же рисуется целая серия рисунков с одинаковыми элементами, то полезно хранить в качестве прототипа их общую часть.

* * *

В процессе практической работы с использованием Автокада нужно применять все эти и многие другие возможности. Поскольку Автокад работает в диалоговом режиме, многие его возможности он сам объясняет и раскрывает пользователю.

1.4 Графические примитивы. Рисование и редактирование

Примитив в любом графическом редакторе - это элемент или группа элементов, которые на экране можно построить, удалить или переместить с помощью одной команды.

Примеры примитивов:

- точка (маркер);
- линия (ломаная);
- дуга;
- штриховка;
- текст

и т.д. Набор разного рода примитивов существует в любом графическом редакторе, но существует их обязательный перечень, который определен международным стандартом в области машинной графики.

Рисование любого примитива требует ввода команды и информации для ее выполнения. Чем сложнее примитив, тем больше информации нужно для его определения. Так, точка рисуется при указании ее положения мышью или

заданием координат - в плоском случае это два числа, декартовы координаты x, y или в полярной системе радиус r и угол β.

Если рисуется отрезок, нужно задать положение двух точек (4 числа, по две координаты каждого из концов отрезка).

Для рисования окружности можно использовать разные возможности (окружность рисуется по таким исходным данным, как центр и радиус, центр и диаметр, три точки, две касательные и радиус и т.д.). В любом случае примитив должен определяться однозначно своими параметрами. Например, для рисования дуги в Автокаде предусмотрено более 10 вариантов задания ее параметров.

Особый примитив - ПОЛИЛИНИЯ (в ранних версиях был только примитив ПОЛОСА). Это связанная последовательность отрезков и дуг, и рассматривается она Автокадом как единый элемент (примитив) рисунка.

Многоугольники в Автокаде строятся с числом сторон от 3 до 1024. Если задать число сторон дробным или меньше 3, редактор эту информацию не воспринимает.

Примитив (команда) ТЕКСТ можно использовать для нанесения надписей на рисунке - в любом его месте, произвольного размера и ориентации. Важно запомнить, что при выходе из команды все, что набрано, является единым примитивом. Если вы обнаружите ошибку и захотите ее исправить после выхода из команды, то стереть одну букву вам не удастся – сотрется сразу весь текст. Поэтому все исправления в тексте следует делать до выхода из команды. При следующем входе в команду ТЕКСТ следующий текст будет уже новым примитивом.

Любые изменения в уже нарисованном рисунке делаются с помощью команды РЕДАКТ. Это касается стирания части или всего рисунка, переноса изображения, копирования, отражения, изменения масштаба, "размножения" части изображения (команда МАССИВ), удаления части примитива (команда ОБРЕЖЬ), разметки отрезка или дуги на равные части, изменения цвета и типа линий и т.д.

1.5 Работа в Автокаде

При обращении к Автокаду высвечивается главное меню Автокада По запросу системы начинается диалог. Имя рисунка состоит из комбинации 8 латинских букв и/или цифр, при этом большие и малые буквы не различаются, а пробелы внутри имени запрещены. Задав имя, мы тем самым создаем файл на диске, куда пишется в дальнейшем вся информация о рисунке. Нужно помнить, что файл - это набор чисел (в двоичном их представлении), и любой

рисунок в файле - это числовой массив. Преобразование рисунка - это всегда работа с массивом чисел (базой данных).

В Автокаде осуществляются два состояния – режим выполнения команды и режим ожидания. Команда выполняется при ее вводе, затем автоматически включается режим ожидания.

В Автокаде различают так называемые основные команды. Команды выполняются в любой последовательности после набора на клавиатуре или после выбора из меню. Если команда сразу выполняется при обращении к ней, она заканчивается двоеточием. Если же она переводит нас в подменю, двоеточия нет. В целом построение «дерева команд» сделано по тому же принципу, что и организация каталогов, например, в MS-DOS.

В верхней части экрана помещается надпись "Слой ". Манипуляция со слоями делается с помощью команды СЛОЙ. Каждому слою присваивается имя - алфавитно-цифровая последовательность длиной до 31 символа.

Через некоторое время работы в Автокаде имеет смысл периодически фиксировать результаты. Для этого используется команда СОХРАНИ. В современных версиях Автокада это делается автоматически с помощью установок, вводимых в пакет при его использовании в первый раз.

Создание рисунка проводится серией команд РИСУЙ, редактирование - с помощью РЕДАКТ. В редакторе предусмотрена команда ОТМЕНИ - она позволяет отменить последнюю команду, если при ее выполнении получен нежелательный или неожиданный результат. При этом можно вернуться назад на некоторое целое число команд.

В главном меню Автокада есть пункт СПРАВКИ. При обращении к нему появляются возможности:

- определить площадь фигуры или длину линии;
- просуммировать (или вычесть) площади нескольких фигур;
- БДСПИСОК (список базы данных) даст информацию о всех примитивах данного рисунка.

Работа с отдельными частями (сегментами) рисунка возможна с применением команды РАМКА (зуммирование). С ее помощью можно рассмотреть изображение в увеличенном масштабе, перенести, стереть какой-либо мелкий элемент рисунка и т.д. Центр рамки автоматически становится центром всего изображения на экране.

Практически работа в Автокаде начинается с определения набора слоев, в которых строится изображение, типа линий и их цвета в каждом слое. Только после этого следует начинать рисунок. Можно и по ходу дела добавлять слои, но удобнее заранее определиться, сколько и слоев нужно и каких.

В кратком изложении нельзя изложить все команды Автокада и приемы, которые следует использовать при построении чертежей. Дело упрощается тем, что, во-первых, современные версии Автокада снабжены обучающими подсистемами, и при внимательном общении с системой Автокад она сама

многие вопросы снимает. Во-вторых, по Автокаду есть большое количество руководств, от кратких вводных описаний возможностей системы до солидных по объему фолиантов, где изложены детали, требующиеся разве что при профессиональной работе с Автокадом.

Все возможности Автокада со временем расширяются. Более наглядным становится меню команд, причем для современного пользователя, привыкшего работать в оболочке Windows, эти команды в виде пиктограмм внешне похожи, например, на меню команд Word. Размещать эти пиктограммы можно в том месте рабочего поля, где удобно пользователю. При подведении курсора к пиктограмме появляется дублирующая надпись – команда. В этом отношении работа на современных версиях Автокада даже проще, чем на предыдущих.

В целом возможности Автокада настолько широки, что конкретный пользователь всегда использует только часть их - то, что непосредственно необходимо. Ситуация схожа со многими другими сферами деятельности - например, в разговоре или при письме мы используем далеко не всю лексику родного языка. Поэтому следует помнить, что выполняемые нами лабораторные работы по Автокаду дают понятие лишь о небольшой части его возможностей. Важно уметь в дальнейшем самостоятельно осваивать возможности Автокада, благо нет недостатка в разного рода руководствах, да и сам Автокад относится к числу обучающих систем - после небольшого опыта работы с ним становится ясно, как осваивать новые его возможности.

Контрольные вопросы к разделу I:

1. Сформулируйте разницу между встроенными графическими редакторами и графическими оболочками.
2. Что представляет собой аппаратное и программное обеспечение компьютерной графики?
3. Перечислите основные аппаратные возможности для обеспечения работы 10-й версии Автокада.
4. Дайте определения понятиям:
   - мировая система координат;
   - пользовательская система координат;
   - слои рисунка;
   - командное и графическое меню;
   - рисунок-прототип.
5. Что такое примитив? Назовите основные типы примитивов в Автокаде.

II ЯДРО ГРАФИЧЕСКОЙ СИСТЕМЫ

2.1 Понятие о ядре графической системы

Ядром графической системы (ЯГС) называется первый международный стандарт в области машинной графики.

Этот стандарт определяет:

1. Каким образом математически создается изображение. Этот вопрос мы будем затрагивать, когда будем рассматривать преобразования объектов и способы представления их на растре.

2. Как изображение представляется на внешних устройствах. Здесь речь идет не только о представлении на экране монитора, но и на принтере, на графопостроителе и т.д.

3. Способы преобразования изображения (сегментация, преобразование, ввод). Эти преобразования делаются и при работе с системой на экране, и при выводе изображения на печать - здесь тоже есть необходимость и возможность менять масштаб рисунка, его ориентацию и т.д.

Самое важное преимущество, получаемое за счет наличия ЯГС:

- появляется возможность единой трактовки и единой терминологии при создании графических систем, их использовании, обсуждении, обучении.

ЯГС разделяет всех пользователей машинной графики на три группы:

1. **Разработчик**

Это специалист, создающий графическую систему в соответствии с требованиями к ней и приспосабливающий ее для нужд массового пользователя (прикладного программиста). При разработке любой графической системы возможна ее реализация на нескольких уровнях - от минимального набора возможностей до полного функционального множества. Уровень разработки определяет сам разработчик, но на любом уровне должны соблюдаться все требования стандарта.

Разработчик - не обязательно один конкретный человек, обычно это группа программистов или целая фирма, занимающаяся разработкой графических систем. Так, фирма AutoDesk является основным разработчиком системы Автокад, и в штате ее сотни программистов, каждый из которых делает определенную часть графической системы.

2. **Прикладной программист**

Это непосредственный пользователь машинной графики - мы с вами. Мы определяем, что нужно нам от графической системы в каждой конкретной

задаче и используем все возможности системы, предусмотренные разработчиком. Стандарт предусматривает, чтобы именно наши потребности, потребности пользователей, были положены в основу новых разработок и были ими удовлетворены.

2.1 Оператор

Человек, сидящий за пультом ПК и работающий с ним непосредственно, называется оператором. В отличие от пользователя, он не преследует своих целей при использовании графической системы. Как правило, он работает по некоторой «шпаргалке» (методическому руководству) или жесткой инструкции, выполняя определенный перечень действий в соответствии с ней.

2.2 Принципы, цели, основные понятия ЯГС.

Основные принципы, в соответствии с которыми вводится ЯГС как международный стандарт (т.е. для чего вообще он нужен):

1. Обеспечение возможности переноса прикладных программ с одной установки на другую.
2. Оказание помощи прикладным программистам в понимании и применении графических методов.
3. Ориентирование производителей оборудования на правильный выбор возможностей создаваемых устройств.

При разработке самого ЯГС выполняется определенный перечень требований, обеспечивающих соблюдение перечисленных принципов. Эти требования предъявляются как к самому стандарту, так и к графическим системам, разрабатываемым на его основе. Далее будем считать, что эти требования формулируются применительно к графическим пакетам.

1. **Согласованность**: обязательные требования стандарта не должны противоречить друг другу.

Это простое на первый взгляд и естественное требование нарушается уже перечисленными ниже принципами. По существу такое положение отражает простую истину: нельзя создать сложную систему (не только в компьютерной графике) без противоречий между составляющими ее отдельными подсистемами.

2. **Совместимость**: другие стандарты или общепризнанная практика не должны нарушаться.

Применительно к графическим системам это означает, например, что большинство требований государственных стандартов (ГОСТ, ГОСТ-Р) или единой системы конструкторской документации (ЕСКД) при выполнении

чертежей должны выполняться если не автоматически, то при минимальных усилиях пользователя – система должна быть «настроена» на их выполнение. На примере системы Автокад очень показательны в этом отношении способы простановки размеров на любом чертеже или на любой схеме – их практически невозможно поставить неправильно, сама система делает это правильно в соответствии со сделанными нами установками.

3. **Ортогональность**: функции и модули системы должны быть независимыми друг от друга; если же такая зависимость имеется, то она должна быть четко определенной.

Здесь имеется в виду то обстоятельство, что графический пакет, как любая программа, не может работать с неопределенным и неясным алгоритмом. Поэтому если внутри пакета отдельные части программного комплекса могут влиять друг на друга, это влияние должно быть понятным.

4**. Полнота**: все запросы большинства приложений (пользователей) должны выполняться на каком-либо функциональном уровне реализации графической системы.

Здесь следует пояснить, что реализация графического пакета может в соответствии с ЯГС проводиться на разных функциональных уровнях. В простейшем варианте реализации по существу, просто возникает возможность рисовать графические примитивы. В наиболее сложном варианте допускается построение и преобразование, в том числе динамическое, сложных геометрических объектов.

5. **Неизбыточность**: на данном функциональном уровне не нужны возможности, которые не используются.

Это означает, что при выполнении конкретных команд задействуются лишь те подпрограммы из пакета, которые необходимы, и система сама должна определять тот минимальный набор подпрограмм, который работает в данный момент.

6. **Компактность**: результат должен достигаться минимальным числом функций и их параметров.

Например, для построения отрезка можно (теоретически) задавать сколько угодно точек, лежащих внутри него. Но минимальное число точек – две, и для рисования отрезка требуется 4 числа – это две пары координат x и y для двух концов отрезка, или пара координат начала отрезка, и два числа, характеризующих направление и длину отрезка от начальной точки. В любом случае этих четырех чисел вполне достаточно для однозначного построения отрезка, и соответствующая команда настроена на ввод именно такого количества числовых значений. Если попытаться ввести больше чисел, графическая система либо не сработает и выдаст соответствующий запрос,

либо среагирует на первые четыре числа и проигнорирует ввод остальных. Аналогичные примеры можно привести и с построением других примитивов, а также выполнением более сложных команд.

7. **Широта**: набор функций должен быть достаточно богат, чтобы предлагаемые ими возможности выходили за рамки элементарных и включали средства высокого уровня.

Это требование и приводит к появлению все новых и новых версий графических пакетов – каждая новая версия расширяет этот набор функций, о котором идет речь в сформулированном требовании.

8. **Комфортабельность**: должна быть обеспечена адаптация графической системы к нуждам пользователя.

Это требование означает, что система может быть так «отрегулирована» пользователем, что при работе с ней не нужно каждый раз проводить ее настройку «под себя». Это отражается в возможности создания собственных рисунков-прототипов, в создании своих библиотек стандартных элементов чертежей и схем, в организации своего меню часто встречающихся команд и т.д.

Кроме того, это требование означает и комфортность работы с системой в самом обычном смысле этого слова. Например, психологами замечено, что если задержка выполнения команды от момента ее задания (нажатия на кнопку Enter на клавиатуре компьютера) до момента выполнения (появления соответствующего изображения на экране) составляет время более 0.2 секунды, через некоторое время работы с такими командами пользователь начинает нервничать без всяких, казалось бы, внешних причин. Не фиксируя это время задержки сознательно, на уровне подсознания пользователь замечает его, и это начинает его раздражать. Поэтому в современных графических системах такая задержка должна либо быть менее 0.2 секунды, либо система должна сообщать пользователю о том, что такая задержка есть. На экране в большинстве случаев появляется значок (обычно в виде песочных часов или циферблата), который и означает необходимость такой задержки.

9. **Прозрачность**: концепции и возможности графической системы должны быть понятны прикладному программисту.

Это отражается в наличии обучающей диалоговой системы и справочной системы, в ясном и однозначном толковании смысла той или иной конкретной команды, в наличии документации на графическую систему и т.д.

10. **Устойчивость**: реакция графической системы на ошибку программиста должна быть минимальной и понятной для пользователя.

Это означает, что при случайной или преднамеренной ошибке система не должна «зависать», уничтожать нарисованный чертеж или схему и т.п., а

выдавать диагностику ошибки и по возможности рекомендации по ее устранению.

11. **Приборная независимость**: прикладная программа, использующая возможности графической системы, может обращаться к различным устройствам ввода-вывода без предварительного изменения функций.

Это означает, что в процессе установки графической системы на ПК с конкретной конфигурацией система сама, в автоматическом режиме, настраивается на все ее устройства, и какая-либо переналадка ни конфигурации компьютера, ни графической системы не требуется. Более того, должно обеспечиваться соответствие любого чертежа или рисунка на экране с его «твердой копией», выводимой на принтер или графопостроитель. При этом при необходимости пользователь имеет возможность менять масштаб выводимого рисунка, в том числе неодинаково по разным осям.

12. **Приборная широта**: возможности всех графических устройств ввода-вывода должны использоваться в стандартном графическом пакете в полном объеме.

Это требование легко проиллюстрировать на примерах такого рода. Известно, что тип и качество монитора определяют четкость построенного на нем изображения. Графическая система автоматически заставляет строить изображение на мониторе с наибольшей доступной четкостью, не требуя при этом каких-либо дополнительных действий или команд. То же самое можно сказать при возможности принтера, графопостроителя и т.д.

13. **Реализуемость**: разработка графических систем должна быть реализуема на любом из основных языков программирования, в среде большинства операционных систем, для большинства графических устройств.

Это и определяет возможность работы графических систем, разработанных разными фирмами, на различных конфигурациях компьютеров, в различных операционных системах.

14. **Языковая независимость**: все стандартные средства графической системы можно выразить средствами стандартных языков программирования.

Это означает, что не создается специальный язык программирования для построения и преобразования графических объектов, а разработчик должен ориентироваться на уже существующие и распространенные стандартные языки.

15. **Эффективность**: стандарт должен быть реализуем без привлечения алгоритмов, требующих больших временных затрат.

Это требование перекликается с требованием комфортности, обсуждавшимся выше. При построении и преобразовании графических

объектов почти любая конкретная задача может быть решена различными способами, и стандарт требует выбора наиболее быстро работающего.

16. **Надежность**: оператор и прикладной программист должны быть максимально защищены от последствий сбоя аппаратуры или программного обеспечения.

В большинстве графических систем по прошествии определенного времени включается команда автоматического сохранения файла, который создается пользователем. Это позволяет не потерять результат длительной работы, например, если внезапно отключается электропитание.

Все эти требования связаны и иногда противоречивы. По-видимому, создание любой сложной системы невозможно без наличия таких внутренних противоречий. Например, широта противоречит неизбыточности, компактность - приборной широте и т.д. В этом отношении необходимо идти на компромиссы: система должна иметь понятную структуру и такой набор функций, который позволит большинству пользователей машинной графики составлять переносимые (с одного устройства на другое) и независимые программы, использующие весь набор графического оборудования.

Основные задачи, которые должна решать графическая система:

1. Делать синтез изображений и их воспроизведение.
2. При работе с частями изображения в разных аппаратных системах координат преобразовывать их при переходе к координатам других графических устройств.
3. Управлять работой всех доступных станций (мониторы, принтеры и т.д.).
4. Обслуживать ввод данных со станции.
5. Позволять делать при необходимости разбиение изображения на части и работать с каждой частью независимо от других.

Для решения перечисленных задач используются следующие **основные концепции построения графических систем**:

1. **Графический вывод изображения**: изображение строится с помощью элементарных объектов (примитивов вывода). Эти примитивы образуют своего рода базис, с помощью элементов которого строится любой графический объект. Примитивы характеризуются атрибутами (цвет и толщина линий, форма шрифта, тип штриховки и т.д.).
2. **Системы координат**: пользователь может работать с примитивами и затем выводить их на графические станции с различными приборными

системами координат; управление этим процессом должна брать на себя графическая система.

3. **Концепция графических станций**: устройства ввода-вывода (дисплей, принтер, графопостроитель) объединяются понятием "рабочая станция" или просто станция.

4. **Концепция логических устройств ввода**: ввод информации может идти от устройства типа потенциометра, с рабочей станции (координаты точки), с клавиатуры - графическая система должна сама реагировать на изменение физического устройства ввода. Аналогию можно провести с разделением винчестера ПК на ряд дисков, когда для одного физического диска вводится целый ряд математических.

5. **Сегментация:** разделение изображения на части (сегменты) и работа с ними, причем есть возможность обрабатывать их независимо друг от друга, вставлять и удалять эти части, копировать, масштабировать, поворачивать и т.д.

6. **Концепция метафайла** как средства для хранения, транспортировки, обмена изображениями с другими системами. Это означает, что создаваемый конкретной системой файл (на носителе информации это числовой массив, хранящий информацию о создаваемом нами рисунке) допускает делать над собой какие-либо действия, после чего соответствующая информация может восприниматься другими графическими или даже текстовыми редакторами.

7. **Таблица состояний** - определяет функциональное состояние графической системы (конфигурацию). Это позволяет, в частности, настраивать графический пакет и под конкретные аппаратные конфигурации.

8. **Концепция уровней системы**: все функции ЯГС соотнесены с 9 уровнями, при этом в нижний уровень входит минимальный набор функций, обслуживающий только вывод, а верхнему уровню отвечают все возможности ЯГС. Разработчик должен реализовать только тот уровень системы, который необходим для данных приложений.

9. **Обработка ошибок**: для каждой функции определяется набор возможных ошибок, при этом прикладная программа либо содержит собственную систему обработки ошибок, либо удовлетворяется стандартной реакцией системы. Это внешне проявляется в диалоговом окне, когда на неправильные действия оператора система выдает соответствующую диагностику.

10. **Размерность системы координат**: в настоящее время ЯГС является двумерной системой. Это удовлетворяет большинству практических запросов, когда все вводимые и выводимые позиции располагаются в двумерном пространстве. Если предусматривать операции в трехмерном пространстве, это потребует введения новых уровней системы выше девятого.

2.3 Примитивы вывода и атрибуты

Для построения изображения графическая система представляет набор базовых элементов - примитивов вывода. Вид примитивов определяется их графическим и визуальным представлением на носителе изображения (экран, графопостроитель, принтер).

Примитив - базовый графический элемент, который создается, редактируется и уничтожается одной командой.

ЯГС как стандарт определяет обязательный набор примитивов:

1. **Ломаная**. В Автокаде буквально такой команды нет. Но, если внимательно проследить, как работает команда «Отрезок», то это по существу и есть команда построения ломаной как серии последовательно соединенных отрезков.
2. **Полимаркер** – набор символов с центрами в указанных точках. Этот вид примитивов обобщает изображение точки. В обычном режиме точка обозначается как небольшого размера крестик, но есть возможность использовать большое количество разного рода символов стандартного вида или построенных пользователем для себя.
3. **Текст**. Внутри графического редактора как бы встроен текстовый редактор. При этом создаваемый единой командой «Текст» массив буквенных, цифровых и других символов является одним примитивом. Удаление единственного символа из этого массива после создания примитива невозможно – в соответствии с определением примитива будет удален весь массив.
4. **Полигональная область** – пустой, закрашенный или заштрихованный многоугольник. Например, в Автокаде этот примитив представляет собой обычный замкнутый многоугольник, число сторон которого – от 3 до 1024 – задается пользователем.
5. **Матрица ячеек** - обобщение матрицы пикселей растрового устройства. Это матрица ячеек, каждая из которых может иметь свой цвет или яркость. Сама такая матрица может рассматриваться как растр, но это уже не технический, а математический растр. Использование такого «растра» можно часто увидеть на телевидении – когда на месте обычного изображения лица человека, узнавание которого нежелательно, помещается своего рода маска. Она и представляет собой матрицу ячеек, каждая из которых включает в себя определенное точек технического растра, обычно при этом атрибуты этой ячейки (цвет, яркость) имеют средние значения по соответствующему ансамблю элементов технического растра.

6. **Обобщенный примитив вывода** - определяет возможность использования специфических средств графического вывода, например, интерполяцию кривых сплайнами, рисование дуг и окружностей и т.д.

Примитивы характеризуются признаками - атрибутами. Для линий это цвет и тип (сплошная линия, штриховая, осевая и т.д.). Если атрибуты зависят от представления на конкретной станции, то это зависимые примитивы (например, цвет на черно-белом экране характеризуется просто интенсивностью подсвечивания). Одни атрибуты полностью зависимы, другие могут быть вообще независимыми. Пример полностью независимого атрибута – форма литер: после ее задания на любом графическом устройстве она фиксирована.

2.4. Системы координат

Мы уже знаем мировую и пользовательскую системы координат. МСК встроена в графическую систему ее разработчиком, ПСК пользователь определяет сам. Помимо этого, ЯГС определяет еще следующие системы координат.

Нормированная система координат (НСК) – это заданные независимо от конкретного устройства координаты, меняющиеся в диапазоне от 0 до 1.

Если координаты (x, y) любого устройства – монитора, принтера и т.д. – могут меняться в пределах $0 \le x \le x_{max}$, $0 \le y \le y_{max}$, то нормированные координаты вводятся с помощью следующих соотношений:

$$x := x/x_{max},\ \ y := y/y_{max}.$$

Очевидно, что введенные таким образом величины x, y будут изменяться в пределах от 0 до 1. Их использование приводит к тому, что на экране любого размера мы видим одно и то же изображение (как в телевизоре с любым размером экрана мы видим изображение полностью, а не часть его – на маленьком экране). Нормированные координаты применяются при запоминании изображений (поэтому рисунок, выполненный на одном экране, будет совершенно таким же на другом), при сегментации изображений и разного рода их преобразованиях.

Координаты устройства, или аппаратная система координат (АСК) – это координатная система конкретной установки (экрана, графопостроителя, принтера и т.д.).

По существу в любой графической системе предусмотрено преобразование координат при выходе из МСК или ПСК в нормированную систему координат, а затем уже в координаты другого конкретного устройства.

Запись числового массива (файла), отвечающего любому графическому изображению, ведется в нормированных координатах. Это позволяет без каких-либо специальных приемов и дополнительных команд переносить изображения с одного монитора на другой.

Контрольные вопросы к разделу II:

1. Что такое «ядро графической системы» и что оно определяет?
2. Сформулируйте ваше понимание терминов разработчик, прикладной программист, оператор графической системы.
3. Перечислите примитивы вывода, определенные ядром графической системы. Что такое атрибуты примитивов?
4. Дайте определения нормированных координат и координат устройства.
5. Приведите примеры противоречивых требований к графической системе, кроме указанных выше в тексте.

III Математические вопросы компьютерной графики

3.1 Преобразования на плоскости и в пространстве

В задачах компьютерной графики (КГ) геометрические понятия, объекты и способы их преобразования играют особую роль.

Вообще построение любого изображения методами КГ означает создание своего рода банка данных (базы данных) о необходимости инициирования конкретных точек на экране с некоторыми признаками светимости (цвет, яркость). При любом преобразовании изображения - изменении масштаба, точки зрения на объект, учете или неучете перспективы, изменении освещенности и т.д. - вся информация об изображении должна быть также преобразована. По существу это означает, что мы должны уметь переработать тот файл, тот числовой массив в памяти ПК, что несет в себе информацию об изображении.

Поскольку рисунок на экране современного дисплея - набор точек (элементов) растра, то и преобразование изображений означает преобразования координат точек и признаков светимости (атрибутов).

3.1.1 Аффинные преобразования на плоскости

Все преобразования в математике проводятся во введенных специальным образом пространствах. Во многих случаях они вводятся таким образом, что совпадают с привычным для нас обычным трехмерным пространством. Для этих пространств даются определения, задающие правила преобразования геометрических объектов, существующих в них. Одним из наиболее известных и часто используемых пространств такого рода является так называемое аффинное пространство, которое представляет собой обобщение понятия обычного пространства на произвольную размерность. В компьютерной графике мы далее рассматриваем только двумерное (плоское) и трехмерное (обычное) пространства.

***Определение*:**
аффинное пространство - точечное множество с присоединенным векторным пространством; при этом:

1) любые точки $a, в$ определяют вектор $ав$;

2) если $в_1 \neq в_2$, то $ав_1 \neq ав_2$;

3) любой вектор можно представить в виде $ав$ при любом a;

4) $ав + вс + са = 0$ при любых $a, в, с$.

Под это определение подходит и обычное, привычное для нас трехмерное пространство, в котором мы живем. С математической точки зрения пространство может быть n-мерным, при $n = 2$ примером такого пространства может служить обычный лист бумаги, далее мы будем еще останавливаться на этом примере. Данное выше определение содержит некоторые моменты, на которые стоит обратить внимание.

Пункт первый означает, что пространство должно быть непрерывным или односвязным. Если, например, в листе бумаги вырезать отверстие, то нельзя будет построить вектор для точек, лежащих на разных сторонах отверстия. В этом случае лист бумаги – с отверстием – уже не является моделью аффинного пространства. Кроме того, первый пункт означает еще, что пространство выпуклое – иначе нельзя построить вектор, всегда лежащий целиком внутри этого пространства.

Второе условие означает привычное, известное из школьного курса математики определение равенства любых геометрических объектов (они равны, если при совмещении полностью совпадают), в том числе и векторов.

Третье означает, что пространство не должно иметь границ, иначе приближение точки a к границе сделает невозможным построение вектора ab. Точка b может оказаться за пределами листа бумаги. Таким образом, если моделью двумерного пространства и может служить лист бумаги, то только лист неограниченных размеров.

Наконец, четвертое условие означает, что мы работаем в так называемом евклидовом пространстве. Существуют пространства с так называемой неевклидовой геометрией (геометрия Римана, Лобачевского), где четвертое требование нарушается, и под определение аффинных пространств они не подходят.

Аффинная система координат - система в n-мерном аффинном пространстве, определяемая совокупностью n линейно независимых векторов, исходящих из начала координат. Координаты точки в этой системе - это компоненты разложения радиус-вектора точки по координатным векторам. Число независимых координат, которые определяют ее положение, т.е. число степеней свободы, задает размерность пространства.

Если на плоскости введена система декартовых координат ОХУ, то точке М соответствует пара чисел (x, y) - ее координаты. В другой системе координат $O_1X_1У_1$ соответствующая той же самой точке пара чисел будет (x_1, y_1), см. рис. 4. Переход от одной точки к другой

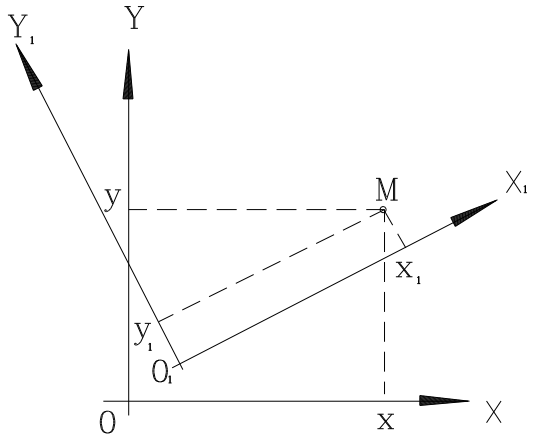


Рис. 4. К произвольному преобразованию координат точки M

описывается соотношениями

$$x_1 = \alpha x + \beta y + \lambda \ ,$$
$$y_1 = \gamma x + \delta y + \mu \ , \qquad (*)$$

где $\alpha, \beta, \gamma, \delta, \lambda, \mu -$ произвольные числа, но при этом

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

Последнее требование означает, что обязательно существует обратное преобразование, и в этом случае можно говорить о взаимно однозначном соответствии точек M и M_1, а преобразование (*) называется невырожденным. Соотношения (*) можно рассматривать:

1) как переход от одной системы координат к другой для одной и той же точки M;

2) как перевод точки M в точку M_1 в одной исходной системе координат.

В последнем случае можно мысленно совместить новую и старую систему координат с «привязанными» к ним точками, тогда в совмещенных координатах это будут две различные точки. Ясно, что в этом случае переход от координат одной точки к другой будет описываться тем же соотношением (*).

Рассмотрим частные случаи преобразований координат, соответствующие конкретным перемещениям точки на плоскости в системе OXY.

А) Поворот отрезка OM = *l* (или движение точки M) вокруг точки O на угол φ (рис. 5). В этом случае координаты точки в ее новом положении, отмеченные индексом 1, связаны с исходными координатами (без индексов) соотношениями

$$x_1 = x*\cos\varphi - y * \sin\varphi,$$
$$y_1 = x*\sin\varphi + y * \cos\varphi. \qquad \text{(а)}$$

В самом деле, из рис. 5 видно непосредственно, что

$$x_1 = l*\cos(\alpha + \varphi) = l*\cos\alpha * \cos\varphi - l*\sin\alpha *\sin\varphi ,$$

и выражения (а) получаются очевидным образом.

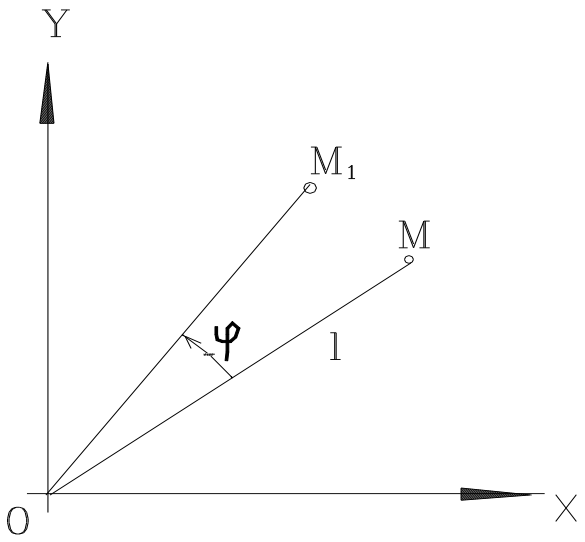


Рис. 5. Поворот (движение) точки M вокруг начала координат на угол φ

Б) Растяжение-сжатие отрезка OM (перемещение точки M) вдоль осей.

В этом случае координаты точки в новом и старом положениях связаны зависимостями

$$x_1 = \alpha * x,\ y_1 = \delta * y \quad (\alpha > 0,\ \delta > 0).$$

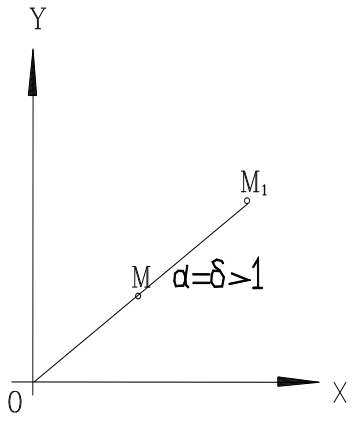


Рис. 6. Преобразование растяжения при $\alpha = \delta > 1$

B) Отражение относительно оси абсцисс

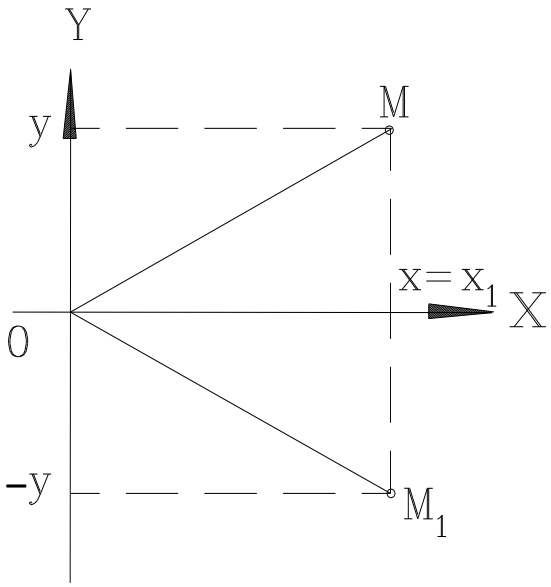


Рис. 7. Преобразование отражения относительно оси x

Очевидны преобразования координат точки при таком преобразовании: горизонтальная координата (абсцисса) не меняется, а вертикальная (ордината) меняет знак

$$x_1 = x,\ y_1 = -y.$$

Г) Перенос точки задается формулами

$$x_1 = x + \lambda,\ y_1 = y + \mu,$$

из которых видно, что знаки и количественные значения величин λ и μ определяют направление и величину переноса.

Из рассмотренных примеров следует два вывода.

1. Частные случаи имеют достаточно наглядный смысл.

2. Любое преобразование типа (*) можно представить как суперпозицию (наложение) простейших преобразований типа **А), Б), В), Г)**.

Последнее утверждение не является очевидным, но в математике оно строго доказано. Поэтому говорят, что эти четыре частных преобразования образуют базис (основной набор) любых преобразований координат точки при ее движении в двумерном аффинном пространстве (на плоскости).

В задачах КГ удобно использовать матричное представление этих преобразований:

А: $\begin{vmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{vmatrix}$; Б: $\begin{vmatrix} \alpha & 0 \\ 0 & \delta \end{vmatrix}$; В: $\begin{vmatrix} 1 & 0 \\ 0 & -1 \end{vmatrix}$.

Оказывается, в такой форме – матричной – нельзя записать четвертое преобразование. Это вносит известные неудобства, так как желательно все четыре преобразования координат записывать в едином виде – тогда и любое произвольное преобразование можно выразить такой же записью, поскольку оно сводится, как уже говорилось, к их суперпозиции (наложению).

Прежде чем двигаться дальше, приведем некоторые справочные сведения из алгебры матриц, которые нам понадобятся .

Матрицей называется упорядоченная система $m \cdot n$ чисел. Если $m = n$, матрица называется квадратной.

Если все $a_{ij} = 0$ при $i \neq j$, получаем так называемую диагональную матрицу.

Если в диагональной матрице все $a_{ii} = 1$, то матрица единичная.

Две матрицы равны A = B, если для всех i, j выполняется равенство

$$a_{ij} = в_{ij}.$$

Сумма матриц определяется по правилу: C = A + B, если

$$c_{ij} = a_{ij} + в_{ij}$$

для всех i, j.

Если A - матрица размерности $m \cdot n$, B - матрица размерности $n \cdot p$, то произведением матриц будет снова матрица - размерности $m \cdot q$:

$$C = A \cdot B,$$

причем

$$c_{ij} = \Sigma\ a_{ik} \cdot в_{kj}\ .$$

Транспонированной матрицей по отношению к исходной называется такая матрица, у которой строки и столбцы меняются ролями. В частности, для вектора-столбца транспонированная матрица - это вектор-строка, и наоборот.

Если det A = 0, матрица A называется особенной или вырожденной, и для нее нельзя построить обратную матрицу. Для обычной (невырожденной) матрицы всегда можно построить обратную матрицу, при этом

$$A^{-1} * A = A * A^{-1} = E,$$

$$\det A^{-1} = 1/\det A,$$

$$(AB)^T = B^T * A^T.$$

Следует также помнить, что произведение матриц является неперестановочным, т.е. в общем случае для произвольных матриц A и B, произведение которых существует,

$$A \cdot B \neq B \cdot A.$$

Однородные координаты

Чтобы охватить единой формой записи все четыре вида преобразований А, Б, В, Г, нужно перейти к так называемым однородным координатам произвольной точки.

Однородными координатами точки M(x, y) называется тройка одновременно не равных нулю чисел x_1, x_2, x_3, для которых выполняются равенства

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

При этом $x_3 \neq 0$ обязательно. В компьютерной графике однородные координаты вводятся следующим образом: в декартовых координатах OXYZ точке M(x, y) ставится в соответствие точка M (x, y, 1).

Заметим, что произвольная точка на прямой OM может быть задана координатами (hx, hy, h), $h \neq 0$.

Вектор, определенный этой тройкой, пересекает плоскость $z = 1$ в точке $(x, y, 1)$ - и она однозначно определяет точку (x, y) в плоскости XOУ.

Тем самым между точками (x, y) и множеством (hx, hy, h) образуется взаимно-однозначное соответствие, что позволяет считать числа (hx, hy, h) однородными координатами точки M.

В проективной геометрии для таких координат принято обозначение $x : y : 1$, или в общем случае $x_1 : x_2 : x_3$.

* * *

Использование однородных координат оказывается зачастую удобным при проведении разного рода расчетов. Так, если речь идет об изменении масштаба, а устройство отображения работает только с целыми числами, то для точки с однородными координатами (0.5; 0.1; 2.5) устройство работать не будет. Выбрав тогда $h = 10$, получим тройку целых чисел (5; 1; 25). Если же у точки координаты (80000, 40000, 1000), то при преобразовании ее координат может произойти переполнение арифметического устройства. В этом случае можно взять $h = 0.001$, при этом однородные координаты будут (80, 40, 1).

Таким образом, основная цель введения однородных координат – удобство их использования.

* * *

При $h = 1$ (*) можно переписать в виде

$$(x_1 \quad y_1 \quad 1) = (x \quad y \quad 1) * \begin{vmatrix} \alpha & \gamma & 0 \\ \beta & \delta & 0 \\ \lambda & \mu & 1 \end{vmatrix},$$

или $$\begin{vmatrix} x_1 \\ y_1 \\ 1 \end{vmatrix} = \begin{vmatrix} \alpha & \beta & \lambda \\ \gamma & \delta & \mu \\ 0 & 0 & 1 \end{vmatrix} * \begin{vmatrix} x \\ y \\ 1 \end{vmatrix}.$$

Из этих матриц 3*3 можно получить для частных случаев:

A - матрица вращения – Rotation

$$[R] = \begin{vmatrix} \cos\varphi & \sin\varphi & 0 \\ -\sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{vmatrix};$$

Б - матрица растяжения сжатия – Dilatation

$$[D] = \begin{vmatrix} \alpha & 0 & 0 \\ 0 & \delta & 0 \\ 0 & 0 & 1 \end{vmatrix};$$

В - матрица отражения (зеркало) - Reflection (Mirror)

$$[M] = \begin{vmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{vmatrix};$$

Г - матрица переноса – Translation

$$[T] = \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \lambda & \mu & 1 \end{vmatrix}.$$

Пример: построить матрицу поворота вокруг точки A(a, b) на угол φ.

Решение заключается в построении последовательности преобразований, которые были рассмотрены выше. Так как существует матрица, определяющая поворот вокруг начала координат, первый шаг и совмещает точку с заданными координатами с началом координат.

1 шаг. Переносим точку A на вектор ($-a$, $-b$) для совмещения заданного центра поворота, точки A, с началом координат:

$$[T] = \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -a & -b & 1 \end{vmatrix}$$

2 шаг. Осуществляем стандартное (базисное) преобразование поворота вокруг точки A, совмещенной с началом координат, на угол φ:

$$[\mathrm{R}] = \begin{vmatrix} \cos\varphi & \sin\varphi & 0 \\ -\sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{vmatrix}.$$

3 шаг. Переносим полученное изображение на вектор $(a,\ b)$ в исходное положение:

$$[\mathrm{T}]^{-1} = \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ a & b & 1 \end{vmatrix}.$$

Итоговое преобразование в виде матрицы получается как перемножение (следует обратить внимание, что именно произведение, а не сумма) матриц отдельных (базисных) преобразований:

$$[\mathrm{T}]\,[\mathrm{R}]\,[\mathrm{T}]^{-1} = \begin{vmatrix} \cos\varphi & \sin\varphi & 0 \\ -\sin\varphi & \cos\varphi & 0 \\ -a\cos\varphi + b\sin\varphi + a & -a\cos\varphi - b\sin\varphi + b & 1 \end{vmatrix}.$$

3.1.2. Аффинные преобразования в пространстве

По аналогии с двумерным случаем вводим однородные координаты

$$(x, y, z) \quad \Rightarrow \quad (x, y, z, 1),$$

или в общем случае

$$(hx, hy, hz, h), \text{ причем } h \neq 0.$$

Тогда для любой точки пространства ее координаты - не равные одновременно нулю четыре числа, определяемые с точностью до общего множителя.

Любое преобразование в аффинном пространстве можно представить как суперпозицию вращения, растяжения, отражения и переноса. При этом вращения рассматриваются уже не вокруг точки, а вокруг трех координатных осей, а отражения – относительно трех координатных плоскостей.

А. Матрицы вращения в пространстве:

вокруг оси *x* на угол φ :

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

вокруг оси *y* на угол ψ:

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

вокруг оси *z* на угол χ:

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

При вращении вокруг оси OX координата любой точки x не меняется, и в соответствующей матрице $[R_x]$ единица в первой строке отражает это обстоятельство. Аналогично построены и две другие матрицы вращения.

Б. Матрица растяжения-сжатия

$$[D] = \begin{vmatrix} \alpha & 0 & 0 & 0 \\ 0 & \beta & 0 & 0 \\ 0 & 0 & \gamma & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix};$$

В этой матрице коэффициенты α, β, γ задают масштаб изменения размеров изображения вдоль осей OX, OУ, OZ соответственно. При равенстве этих коэффициентов друг другу будет построено изображение, подобное исходному. Если эти коэффициенты больше единицы, изображение увеличивается, меньше – уменьшается.

В. Матрицы отражения относительно плоскостей XOY, YOZ, ZOX:

$$[M_z] = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}; \quad [M_x] = \begin{vmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}; \quad [M_y] = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix};$$

Каждая из этих матриц меняет знак одной из координат точки (и изображения в целом), что и означает получение зеркального отображения относительно плоскости, перпендикулярной этой координате.

Г. Матрица переноса

$$[T] = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \lambda & \mu & \nu & 1 \end{vmatrix};$$

В этой матрице параметры λ, μ, ν определяют смещения изображения вдоль осей x, y, z соответственно.

Пример

Построить матрицу вращения на угол вокруг прямой L, проходящей через точку A(a, b, c) и имеющей направляющий вектор (l, m, n), т.е. вектор, образующий с осями координат Ox, Oy, Oz углы, косинусы которых равны l, m, n.

1 шаг. Поскольку у нас есть стандартные матрицы вращения вокруг координатных осей, в качестве первого шага осуществим перенос точки A, лежащей на прямой, на вектор $(-a, -b, -c)$, так что точка A попадает в начало координат:

$$[T] = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -a & -b & -c & 1 \end{vmatrix}$$

а прямая L тогда будет проходить через начало координат.

2 шаг. В общем случае прямая L направлена произвольным образом. Поэтому добьемся ее совмещения с осью z двумя поворотами - вокруг оси x и вокруг оси y. L_1 - проекция L на плоскость $x = 0$; для L_1 направляющий вектор будет (0, m, n), тогда угол ψ - вращения вокруг оси x - будет определяться выражениями $\cos\psi = n/d$, $\sin\psi = m/d$, причем

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

Матрица этого вращения будет

$$[R_x] = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & n/d & m/d & 0 \\ 0 & -m/d & n/d & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}.$$

За счет этого вращения изменятся координаты вектора (l, m, n); они становятся

$$(l \quad m \quad n \quad 1)*[R_x] = (l \quad 0 \quad d \quad 1).$$

Второй поворот теперь делаем вокруг оси ординат Oy на угол υ, определяемый соотношениями

$$\cos\upsilon = l, \ \sin\upsilon = -d.$$

Матрица этого вращения имеет вид

$$[R_y] = \begin{vmatrix} l & 0 & d & 0 \\ 0 & 1 & 0 & 0 \\ -d & 0 & l & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix};$$

3 шаг. Вращение вокруг оси z (теперь эта ось совмещена с прямой L) на заданный угол φ. Матрица этого вращения будет

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

4 шаг. Поворот вокруг оси y на угол - υ, вокруг оси x - на угол - ψ;

Вращение в пространстве является некоммутативным (т.е. неперестановочным), поэтому порядок, в котором проводится вращение, весьма существенен.

5 шаг. Перенос на вектор (a, b, c).

Перемножая найденные матрицы в порядке их построения, получим окончательную матрицу требуемого преобразования:

$$[T]\,[R_x]\,[R_y]\,[R_z]\,[R_y]^{-1}\,[R_x]^{-1}\,[T]^{-1}\,.$$

Заметим, что перемножаются квадратные матрицы четвертого порядка, и в результате получаются снова матрицы такого же порядка. В результате подобных преобразований дело сводится к построению невырожденных матриц вида

$$\begin{vmatrix} \alpha_1 & \alpha_2 & \alpha_3 & 0 \\ \beta_1 & \beta_2 & \beta_3 & 0 \\ \gamma_1 & \gamma_2 & \gamma_3 & 0 \\ \lambda & \mu & \nu & 1 \end{vmatrix}. \qquad \textbf{(**)}$$

* * *

Таким образом, любое преобразование геометрической фигуры, составленной из некоторого набора точек, проводится с помощью матричных операций. По существу, единой программы перемножения матриц, в том числе матрицы на вектор, достаточно, чтобы перестроить любой элемент рисунка или рисунок в целом. В любом случае такого рода преобразования сводятся к операциям с матрицами вида (**).

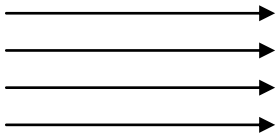
3.2 Виды проектирования

В машинной графике принято использовать два основных вида проектирования:

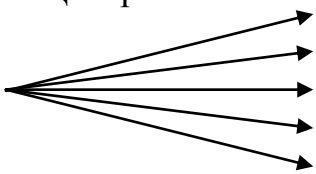
- центральное;
- параллельное.

В первом случае все проецирующие прямые идут из одного центра (он так и называется – центр проектирования), во втором случае проецирующие прямые параллельны друг другу. Смысл их можно понять из рисунков:

Параллельное



Центральное



При этом первый способ проектирования можно рассматривать как предельный случай второго способа при удалении центра проектирования на бесконечность.

Каждый вид разбивается на подвиды в соответствии с таблицами.

Таблица **1**

Параллельные проекции		
Ортографическая проекция	Аксонометрическая проекция	Косоугольная проекция
	Триметрия	Свободная проекция
	Диметрия	Кабинетная проекция
	Изометрия	

Таблица 2

Перспективные проекции		
Одноточечная	Двухточечная	Трехточечная

1. Ортографическая проекция. В этом случае картинная плоскость совпадает с одной из координатных плоскостей или параллельна ей. Рассмотрим пример проектирования вдоль оси x. Если вспомнить, как это мы делаем в инженерной графике, в проекционном черчении, то дело сводится просто к тому, что объемная в общем случае фигура при проектировании ее на плоскость $x = 0$ «сжимается» вдоль x, и вся «ложится» на плоскость, при этом размеры ее в направлении оси x значения не имеют. Это означает, что для всех точек этой фигуры координаты x становятся равными нулю. При этом две остальные координаты (z и y) не меняются. Становится понятным тогда, как должна выглядеть матрица соответствующего преобразования координат, или, как принято говорить, матрица проектирования на плоскость OZY вдоль оси X. Она имеет вид:

$$[P_x] = \begin{vmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix};$$

Воздействие этой матрицы на вектор (или любую точку в пространстве) означает обнуление координаты x - но это и должно происходить при проектировании объекта на плоскость вдоль оси x. Как видно по приведенной матрице, значения координат y и z при этом не меняются.

Если же плоскость проектирования параллельна координатной плоскости и находится на расстоянии p от нее, то нужно умножить матрицу $[P_x]$ на матрицу сдвига; тогда

$$[P_x{}^p]* \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ p & 0 & 0 & 1 \end{vmatrix} = \begin{vmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ p & 0 & 0 & 1 \end{vmatrix}.$$

По аналогии матрицы проектирования вдоль двух других осей будут

$$\left[P_y^q\right] = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & q & 0 & 1 \end{vmatrix}, \qquad \left[P_z^r\right] = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & r & 1 \end{vmatrix}.$$

Здесь обозначены сдвиги на расстояния q, r соответственно вдоль осей OY и OZ.

2. Аксонометрическая проекция. В этом случае проецирующие прямые, как и при ортографическом проецировании, перпендикулярны картинной плоскости. Сама картинная плоскость может располагаться в пространстве произвольным образом. В связи с этим различаются три частных случая.

Триметрия: нормаль к картинной плоскости образует со всеми тремя ортами координатных осей различные углы. В этом случае искажения размеров проектируемой фигуры вдоль всех трех осей различны.

Диметрия: два из этих углов равны между собой, соответственно искажения вдоль двух осей одинаковы, а вдоль третьей в общем случае будут больше или меньше.

Изометрия: все три угла равны между собой, и искажения размеров проецируемой фигуры вдоль осей одинаковы.

Каждый из этих видов проекций получается комбинацией поворотов, за которой следует параллельное проецирование. Таким образом, итоговое преобразование может быть получено в виде матрицы, полученной перемножением матрицы параллельного проецирования и матрицы поворота.

3. Косоугольные проекции. При таком проецировании пучок проецирующих прямых не перпендикулярен плоскости экрана (картинной плоскости). Принято различать два частных случая.

Свободная проекция - угол наклона проецирующих прямых к плоскости равен половине прямого.

Кабинетная проекция - частный случай свободной, при этом масштаб по третьей оси вдвое меньше, чем по двум первым.

При косоугольном проектировании орта Z на плоскость XOY

$$(0 \quad 0 \quad 1 \quad 1) \longrightarrow (\alpha \quad \beta \quad 0 \quad 1),$$

а матрица такого преобразования должна быть

$$\begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \alpha & \beta & 0 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix},$$

причем в случае свободной проекции $\alpha = \beta = \cos(\pi/4)$,

а в случае кабинетной проекции $\alpha = \beta = 1/2*\cos(\pi/4)$.

4.Перспективные или центральные проекции.

Такие проекции строятся более сложным образом.

Пусть центр проецирования находится на оси OZ, точка с координатами C(0, 0, c), а плоскость проецирования - XOY. Для произвольной точки M(x, y, z) построим прямую, проходящую через M и C. Тогда ее параметрические уравнения будут (X, Y, Z – текущие координаты)

$$X = x \cdot t, \quad Y = y \cdot t, \quad Z = c + (z - c) \cdot t.$$

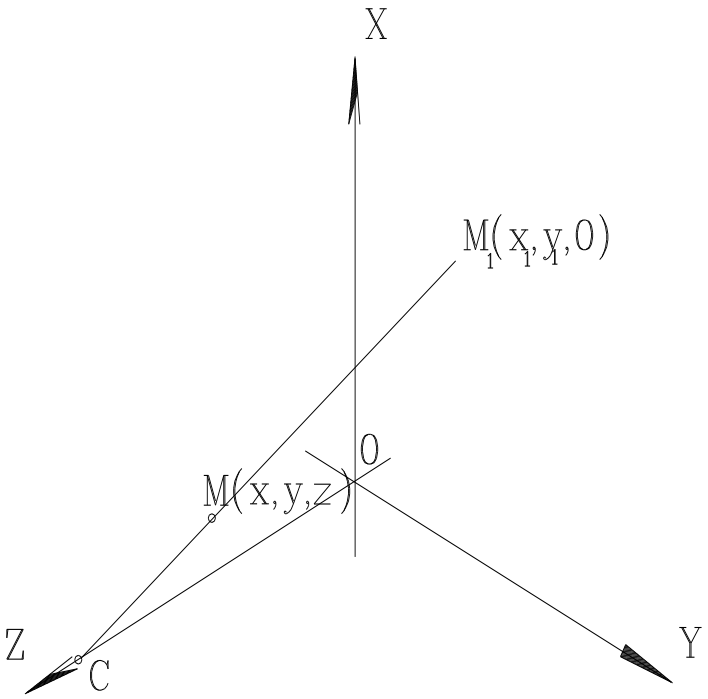


Рис. 8. К преобразованию центрального проецирования

В том, что это уравнение нужной проецирующей прямой, можно убедиться следующим образом. Линейная зависимость текущих координат X, Y, Z от параметра t означает, что это прямая линия. При $t = 0$ значения текущих координат прямой будут

$$X = 0,\ Y = 0,\ Z = c,$$

т.е. прямая проходит через точку C. При $t = 1$ текущие координаты будут

$$X = x,\quad Y = y,\quad Z = z,$$

т.е. прямая проходит и через точку M.

При $Z = 0$ будет $t = 1/(1\text{-}z/c)$. Это значит, что координаты точки пересечения этой (проецирующей) прямой с плоскостью XOY определяются значениями

$$X_1 = x/(1\text{-}z/c), \qquad Y_1 = y/(1\text{-}z/c).$$

Такое преобразование координат можно получить с помощью матрицы

$$[G] = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1/c \\ 0 & 0 & 0 & 1 \end{vmatrix} \qquad (*)$$

В самом деле,

$$(x \;\; y \;\; z \;\; 1)\,[G] = (x \quad y \quad 0 \quad 1\text{-}z/c),$$

или, поделив на последний элемент все компоненты этой вектор-строки (что можно сделать, поскольку мы используем однородные координаты):

$$(x/(1\text{-}z/c) \qquad y/(1\text{-}z/c) \qquad 0 \qquad 1).$$

Матрица проецирования [G] вырождена. С точки зрения построения проекций это означает, что точке на картинной плоскости может соответствовать бесконечное число точек проецируемой фигуры. В самом деле, любая точка фигуры, лежащая на проецируемой прямой, даст то же изображение, что и исходная точка M. С математической точки зрения вырожденная матрица (по определению) не имеет обратной, т.е. между исходной точкой и ее изображением на картинной плоскости нет взаимно-однозначного соответствия.

В то же время матрица соответствующего перспективного преобразования (без проецирования) может быть записана в виде

$$[PP] = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1/c \\ 0 & 0 & 0 & 1 \end{vmatrix}.$$

Для того, чтобы убедиться в этом, рассмотрим пучок прямых, параллельных OZ, и воздействуем на него матрицей [PP].

Каждая прямая этого пучка однозначно определяется точкой своего пересечения с плоскостью OXY и описывается уравнением

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

Переходя к однородным координатам и используя матрицу [PP], получаем

$$(x \;\; y \;\; t \;\; 1) [PP] = (\;\; x \;\; y \;\; t \;\; 1\text{-}t/c),$$

или

$$(x/(1- t/c) \qquad y/(1- t/c) \qquad ct/(c - t) \qquad 1).$$

При $t \to \infty$ точка $(x \;\; y \;\; z \;\; 1)$ преобразуется в точку $(0 \;\; 0 \;\; 1 \;\; 0)$ – для того, чтобы убедиться в этом, достаточно все разделить на t и перейти к пределу при $t \to \infty$

$$\lim_{t \to \infty} (x/t \;\; y/t \;\; 1 \;\; 1/t);$$

Соответствующая ей точка $(0 \;\; 0 \;\; -c \;\; 1)$ получится аналогично из

$$\left(\frac{x}{1-t/c} \qquad \frac{y}{1-t/c} \qquad \frac{ct}{t-c} \qquad 1\right)$$

при $t \to \infty$.

Тем самым бесконечно удаленный центр $(0 \; 0 \; 1 \; 0)$ пучка параллельных оси OZ линий переводится в точку $(0 \; 0 \; -c \; 1)$ оси OZ. Эта точка называется точкой схода.

В общем случае каждый несобственный пучок прямых (т.е. совокупность прямых, параллельных заданному направлению), непараллельный картинной плоскости, под действием преобразования [PP] переходит в собственный пучок.

Центр получаемого пучка и есть точка схода.

Главные точки схода - это точки схода для пучков, параллельных координатным осям, их три.

Для матрицы [PP] (вернее, преобразования с ее помощью) существует лишь одна точка схода, на рис. 9 дана иллюстрация в виде куба, изображенного с помощью перспективной проекции при наличии одной точки схода на оси z.

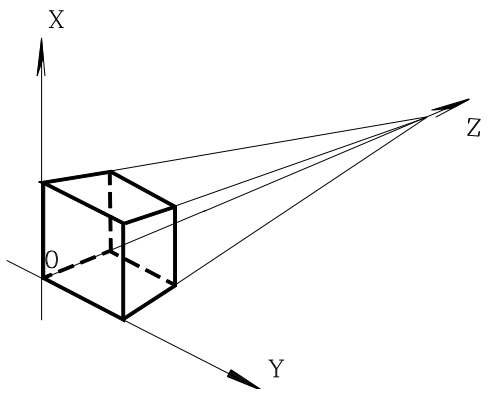


Рис. 9. Изображение куба при наличии одной точки схода

В общем случае таких точек три - когда оси координатной системы не параллельны плоскости экрана. Матрица соответствующего преобразования имеет вид

$$[PP_3] = \begin{vmatrix} 1 & 0 & 0 & -1/a \\ 0 & 1 & 0 & -1/b \\ 0 & 0 & 1 & -1/c \\ 0 & 0 & 0 & 0 \end{vmatrix}.$$

Пучки прямых, параллельных OX (1 0 0 0) и OY (0 1 0 0) переходят соответственно в пучки прямых с центрами

(1 0 0 -1/a), (0 1 0 -1/b),

Или

(- a 0 0 1) , (0 -b 0 1).

Если есть две точки схода, то куб будет выглядеть следующим образом:

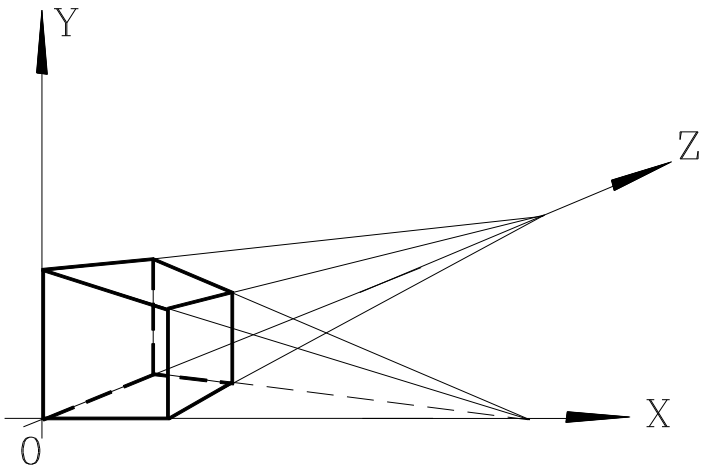


Рис. 10. Изображение куба при наличии двух главных точек схода на осях OX и OZ

Изображение куба при наличии трех главных точек схода (т.е. расположенных на осях) выглядит следующим образом

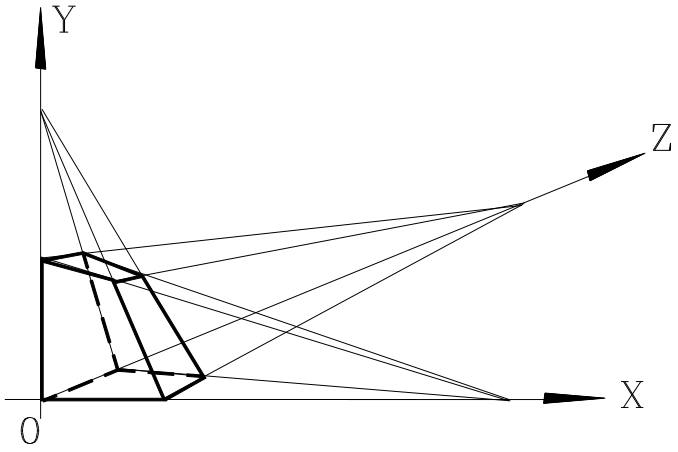


Рис. 11. Изображение куба при наличии трех главных точек схода

3.3 Особенности проекций гладких изображений

При изображении объектов на плоскости используется геометрическая операция проецирования. Это делается с помощью параллельного или центрального пучка прямых. При этом через каждую точку объекта проводится прямая из пучка, а на плоскости изображения (картинной плоскости) отыскивается пересечение с этой прямой. Как уже было отмечено выше, при центральном проецировании все проецирующие прямые проводятся из одной точки, называемой центром; при параллельном проецировании «центр» пучка (в данном случае условный) лежит на бесконечности.

При проецировании искривленных объектов (поверхностей) на плоскость возникают некоторые эффекты - вне зависимости от того, какое проецирование - параллельное или центральное.

Далее принимаем, что картинная плоскость - это X = 0, а пучок проецирующих прямых ей перпендикулярен.

Оказывается, существует три принципиально отличных случая.

1. Проецируем поверхность $z = x$ на плоскость $x = 0$. Если построить систему осей OXZ, то в этой системе уравнение $z = x$ дает прямую линию. В пространственном случае добавится третья ось OY, а прямая линия при любом значении y остается в том же положении, т.е. описывает плоскость. Уравнение этой плоскости $x - z = 0$. Из аналитической геометрии известно, что для поверхности, описываемой уравнением

$$f(x, y, z) = 0,$$

координаты нормального к ней вектора получаются как

$$\left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}\right).$$

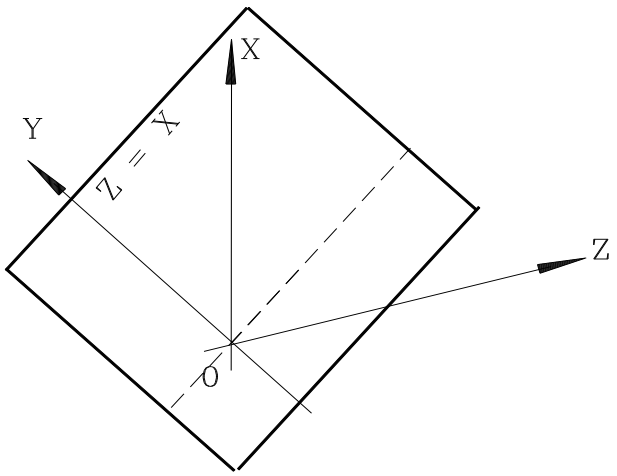
В нашем случае координаты нормали к поверхности $N = (1, 0, -1)$.

Проецирование ведем вдоль вектора $L = (1, 0, 0)$, т.е. вдоль оси OX. Т.к. $(N, L) = 1 > 0$, то вектор проецирования и нормальный к поверхности вектор не перпендикулярны ни в одной точке. Здесь выражение вида (N, L) означает скалярное произведение векторов $N(x_1, y_1, z_1)$ и $L(x_2, y_2, z_2)$ и считается по формуле

$$(N, L) = x_1 \cdot x_2 + y_1 \cdot y_2 + z_1 \cdot z_2.$$

Следует помнить, что при скалярном перемножении результатом является число (скаляр - отсюда и название произведения). Знак этого числа говорит о взаимной ориентации векторов – если между ними острый угол, произведение – положительное число; если прямой угол, произведение обращается в 0; если угол тупой – отрицательное число.

Говорят в таком случае, что полученная проекция не имеет никаких



особенностей (рис. 12).

Рис. 12. К проектированию поверхности без особенностей

2. Проектируем на ту же плоскость поверхность $z = x^2$. В осях OXZ (в плоском, или двумерном, случае) это уравнение параболы. При добавлении оси OY в пространственном случае парабола при изменении y не меняется (в уравнение ее не входит величина y), и поверхность

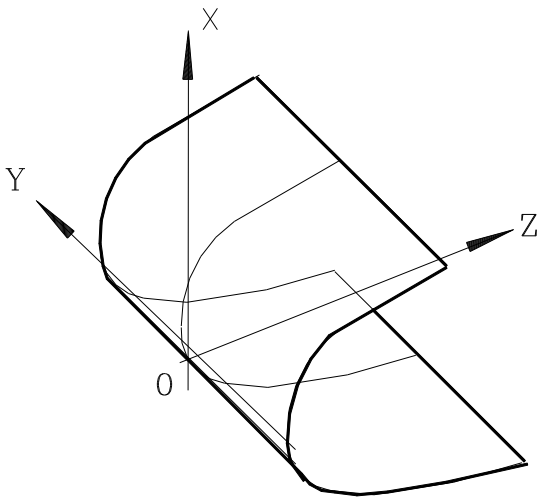


Рис. 13. Проектирование поверхности с особенностью типа складка

получается как след движения параболы вдоль оси OY – это так называемый параболический цилиндр (рис. 13). Иначе уравнение его можно записать в виде $x^2 - z = 0$.

Нормальный вектор к этой поверхности будет $N = (2x, 0, -1)$, и в точках $x = 0$, т.е. на оси y, выполняется соотношение $(N, L) = 0$, т.е. нормальный к поверхности вектор перпендикулярен вектору проецирования.

На плоскости $x = 0$ возникают три разных группы точек:

1) точки $z > 0$ - для них на исходной проецируемой поверхности существует по два прообраза (т.е. исходной проецируемой поверхности);

2) $z = 0$ - для этих точек один прообраз;

3) $z < 0$ - для этих точек прообраза нет вообще.

Особенность такого типа называется складкой (см. рис. 13).

Прямая $x = 0, z = 0$ - особая; для нее векторы N, L ортогональны.

3. Рассмотрим поверхность

$$z = x^3 + xy \qquad \text{или} \qquad x^3 + xy - z = 0.$$

Нормальный вектор этой поверхности

$$N = (3x^2 + y, x, -1).$$

Построим эту поверхность с помощью метода сечений, придавая различные значения величине y.

$y = 1$: $z = x^3 + x$;

$y = 0$: $z = x^3$;

$y = -1$: $z = x^3 - x$;

Соответствующие кривые приведены на рис. 14.

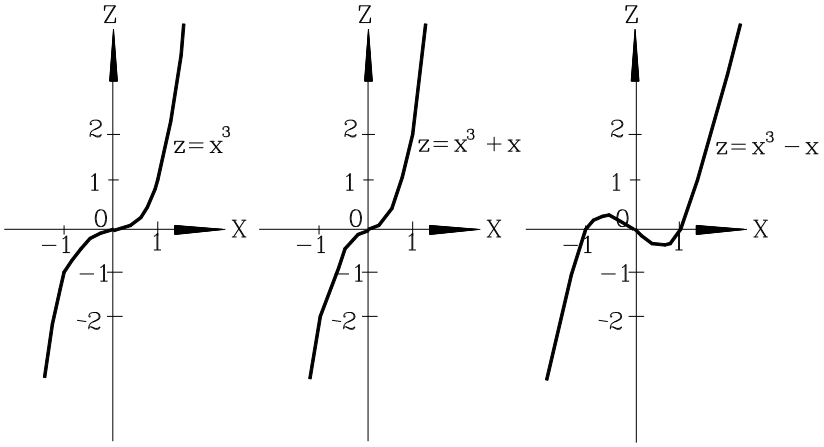


Рис. 14. К построению поверхности $z = x^3 + xy$

Можно затем построить и всю поверхность (рис.15).

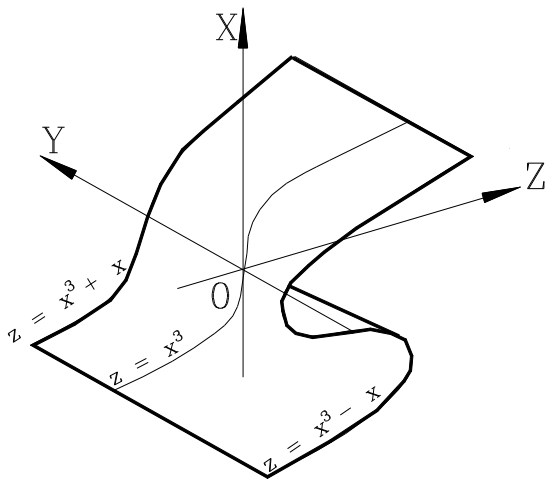


Рис. 15. Общий вид поверхности с особенностью типа сборка

Из условия $(N, L) = 3x^2 + y = 0$ следует, что вдоль кривой, лежащей на поверхности, с уравнениями $y = -3x^2$, $z = -2x^3$ векторы L и N перпендикулярны. Исключая x, получаем, что на этой кривой

$$\left(-\frac{y}{3}\right)^3 = \left(-\frac{z}{2}\right)^2.$$

Это уравнение полукубической параболы на плоскости $x = 0$.

Эта кривая делит точки плоскости на три группы:

1) - точки на острие, у каждой из них на поверхности ровно два прообраза;

2) - внутри острия, для каждой точки три прообраза;

3) - вне острия, для каждой точки один прообраз.

В целом особенность такого типа называется сборкой.

Хотя полукубическая парабола имеет точку заострения на плоскости $x = 0$, ее прообраз на исходной поверхности

$$x = x, y = -3x^2 , z = - 2x^3$$

является регулярной кривой.

В теории особенностей (которая еще носит название теории катастроф) доказывается, что при проецировании на плоскость гладкой поверхности возможны только три рассмотренных типа проекций:

1) обыкновенная; 2) складка; 3) сборка.

В принципе могут быть и более сложные случаи, но при малом изменении либо направления проецирования, либо взаимного расположения плоскости и проецируемой поверхности эти все сложные случаи переходят в перечисленные выше.

3.4. Интерполирующие и сглаживающие кривые

Если попытаться описать всю достаточно большую и сложную поверхность в целом, то простыми формулами не обойтись. Применение же простых формул предпочтительно во многих случаях.

На практике берется небольшой участок поверхности и небольшое число так называемых опорных точек на ней, и через эти точки строится поверхность. Для таких поверхностей и нужно их аналитическое представление с помощью сплайнов.

Визуализация таких кривых и поверхностей сразу дает возможность оценить, что получается в результате проектирования, и внести при необходимости изменения в описание поверхностей.

Рассмотрим задачи геометрического моделирования при проектировании кривых и поверхностей.

Типичные задачи, решаемые в этом случае: по массиву точек на плоскости или в пространстве построить кривую, сводятся к двум основным классам:

1) построение кривой, проходящей через эти точки (интерполяция);
2) построение кривой, проходящей вблизи этих точек (сглаживание).

В качестве примера на рис.16 для одного и того же набора точек приведены две кривые: верхняя кривая – интерполирующая, нижняя – сглаживающая.

При решении той и другой задачи сразу возникают минимум две проблемы:

- в каком классе кривых искать нужную кривую;
- как искать эту кривую.

Остановимся на первой проблеме.

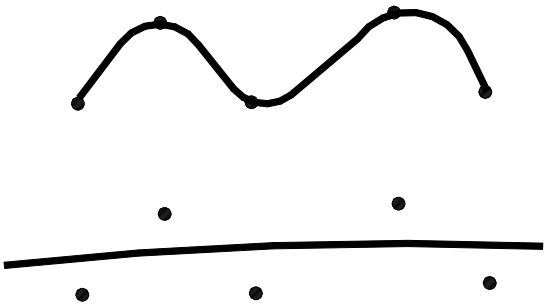


Рис. 16. Примеры интерполирующей (верхней) и сглаживающей (нижней) кривых для одного и того же набора точек

Одно из первоочередных требований к построению искомых кривых – это единственность решения задачи. Второе требование сводится к тому, что построенная кривая должна быть плавной, или гладкой, т.е. описываться дифференцируемыми функциями.

Пусть на плоскости задан набор точек (x, y), $i = 0, 1, ..., m,$ причем

$$x_0 < x_1 < x_2 <...< x_m ,$$

и будем искать интерполирующую кривую в классе многочленов (полиномов).

3.4.1 Интерполяционный полином Лагранжа

Известен полином, носящий имя Лагранжа. Он определяется соотношением

$$y(x) = \frac{(x-x_1)(x-x_2)...(x-x_m)}{(x_0-x_1)(x_0-x_2)...(x_0-x_m)} \bullet y_0 + \frac{(x-x_0)(x-x_2)...(x-x_m)}{(x_1-x_0)(x_1-x_2)...(x-x_m)} \bullet y_1 +$$

$$... + \frac{(x-x_0)(x-x_1)...(x-x_{m-1})}{(x_m-x_0)(x_m-x_1)...(x_m-x_{m-1})} \bullet y_m .$$

То, что эта функция является полиномом, непосредственно видно из ее записи. Все значения x и y с индексами – это просто заданные числа, а функциональная зависимость задается произведением ряда сомножителей, где всюду фигурирует выражение типа $(x - x_0)$ и т.д. Нетрудно увидеть, что этот полином по построению таков, что кривая, описываемая им, проходит через

все заданные точки. В самом деле, если принять $x = x_0$, то первая дробь (множитель перед y_0) обращается в единицу, а остальные – в 0. В итоге при этом значении $x = x_0$ получим $y(x_0) = y_0$. Аналогично при $x = x_1$ получается $y(x_1) = y_1$, и т.д.

Если нужно построить интерполяционный полином Лагранжа всего лишь для двух точек $(x_0, y_0), (x_1, y_1)$, а такая задача тоже может ставиться, то выражение его будет

$$y(x) = \frac{x - x_1}{x_0 - x_1} \bullet y_0 + \frac{x - x_0}{x_1 - x_0} \bullet y_1 .$$

Это не что иное, как уравнение прямой, проходящей через заданные точки $(x_0, y_0), (x_1, y_1)$. Для случая трех точек получится квадратная парабола, для четырех точек – кубическая парабола, и т. д. Если число заданных точек, пронумерованных от 0 до m, составляет $m + 1$, то степень соответствующего полинома Лагранжа на единицу меньше и равна m.

Достоинства полинома Лагранжа:

1) полином просто строится, и соответствующее выражение при необходимости легко программируется;

2) полином однозначно определяется, т.е. при заданном наборе значений x_i и y_i его выражение всегда получается одним и тем же;

3) как и для любого полинома, для полинома Лагранжа существует производная любого порядка (неважно, что с порядка производной, на единицу выше степени полинома, эти производные обращаются в нули – все равно они существуют).

Недостатки полинома Лагранжа:

1) с ростом числа точек растет и степень полинома (m -1 при m точках), и растет степень уклонения кривой от аппроксимируемой зависимости;

2) изменение или добавление хотя бы одной точки приводит к изменению всего полинома.

Но наиболее неприятным является следующий недостаток полинома Лагранжа. Если построить кривую, проходящую через все заданные точки, то она получится при большом числе точек сильно осциллирующей, т.е. сильно уклоняющейся от среднего значения между двумя соседними точками. И хотя при этом кривая проходит через все заданные точки, производные (до любого порядка существующие) могут иметь любые значения между двумя точками, а не те, что отвечают общему характеру зависимости $y = y(x)$. В этом отношении характерны кривые на рис.16: верхняя интерполирующая кривая

проходит через все заданные точки, как и должно быть в соответствии с определением. Однако если взять производную – а ее величина определяется как тангенс угла наклона касательной к кривой – то на всем интервале наклон соответствующей касательной меняется от точки к точке очень заметно. Ясно, что «среднее» значение производной гораздо лучше получится, если его определить по сглаживающей кривой.

Второй, в известном отношении противоположный, способ аппроксимации - с помощью ломаной линии. В этом случае на отдельных участках будет идеальная линейная аппроксимация, зато в угловых точках, совпадающих с точками x_i, первые производные претерпевают разрыв, т.е. имеют разные значения справа и слева от точки.

Компромисс достигается аппроксимацией кривой (полиномом) на отдельных участках, т.е. не всей кривой сразу, а последовательно, участок за участком. Затем проводится плавное сопряжение отдельных участков путем соответствующего подбора коэффициентов полиномов. В этом и заключается центральная идея сплайн-аппроксимаций.

3.4.2 Кубические сплайны. Случай одной переменной

Термин «сплайн» произошел от английского слова spline - так называлась гибкая полоска стали, с помощью которой проводятся гладкие кривые через заданные точки. Этот способ построения кривых использовался при строительстве лодок и корпусов кораблей с гладкими обводами.

Теория сплайнов – один из самых молодых разделов математики, первая работа в этом направлении опубликована Шенбергом в 1946г.

На первом этапе это было решением проблемы приближения функций в теоретическом и практическом плане. Затем обнаружилось, что сами сплайны весьма разнообразны и применимы в различных областях: численные методы, системы автоматизированного проектирования, автоматизация научных исследований, КГ и т.д.

При появлении компьютеров с помощью теории сплайн-аппроксимаций стало возможным описывать достаточно сложные поверхности простыми формулами.

Результат такой аппроксимации называется сплайн-функцией или просто сплайном. Аналогия с чертежными сплайнами достаточно очевидна. Более того, если взять гибкую стальную линейку и с ее помощью провести кривую через ряд опор, оказывается, что кривая будет представлять собой график функции $y(x)$ на каждом участке между опорами - полином третьей степени. На всем промежутке $x_0, x_1, \ldots, x_m$ этот полином – непрерывно

дифференцируемая функция. Такие функции носят название интерполяционных кубических сплайнов.

Определение:

Интерполяционным кубическим сплайном называется функция $S(x)$, для которой

1. $S(x_i) = y_i$, $i = 0, 1, ..., m$;

2. На каждом отрезке $[x_i, x_{i+1}]$, $i = 0, 1, ..., m - 1$ функция имеет вид

$$S(x) = \sum_{j=1}^{3} a_{ij} (x - x_i)^j .$$

3. На всем протяжении $[x_0, x_m]$ у функции $S(x)$ есть непрерывная вторая (по меньшей мере) производная.

Так как на каждом отрезке полином третьей степени определяется 4 числами (коэффициентами), а участков m, то в итоге нужно найти $4m$ чисел.

Сформулируем условия, из которых можно однозначно определить эти коэффициенты. Для этого потребуется $4m$ соотношений, так как число неизвестных в системе уравнений должно быть равно числу неизвестных. Из третьего условия требуем непрерывности сплайна во всех внутренних узлах, т.е.

$$S(x_i - 0) = S(x_i + 0), \; i = 1, 2, ..., m - 1)$$

- это дает $m - 1$ условие. Обратим внимание, что в крайних точках эти условия нельзя сформулировать. Для первой производной тоже $m - 1$ условий, и для второй. Если учесть еще первое требование – $m + 1$ условие - то всего получается $4m - 2$ условия. Два недостающих условия получаются, если задать, например, значения первых производных в граничных точках при $i = 0$ и $i = m$ (это своего рода граничные условия)

$$S'(x_0) = l_0; \quad S'(x_m) = l_m;$$

и тогда задача решается однозначно. Последние два условия по существу означают, что с их помощью мы можем задать направления «входа» в соответствующую кривую при $x = x_0$ и выхода из нее при $x = x_m$.

3.4.3 Бикубические сплайны (случай двух переменных)

Если нужно построить функцию двух переменных по массиву точек в пространстве, то используется понятие интерполяционного бикубического сплайна.

Пусть каждой точке на плоскости

$$(x_i, y_j),\ i = 0, 1,...,m;\ j = 0, 1,..., n,\ (x_0 < x_1 <... < x_m, y_0 < y_1 <... < y_n)$$

отвечает точка z_{ij}; тогда существует массив

$$(x_i, y_j, z_{ij}),\ i = 0, 1, ..., m;\ \ j = 0, 1, ..., n.$$

Определим функцию, графиком которой будет поверхность, проходящая через эти точки.

Определение

Интерполяционным бикубическим сплайном двух переменных называется функция $S(x, y)$, для которой

1. $S(x_i, y_j) = z_{ij};\ \ i = 0,1,..., m;\ j = 0, 1, ..., n;$

2. На каждом частичном прямоугольнике

$$[x_i, x_{i+1}] \cdot [y_j, y_{j+1}],\ (i = 0, 1, ..., m - 1;\ j = 0, 1, ... , n - 1)$$

эта функция имеет вид

$$S(x, y) = \sum_{k=1}^{3} \sum_{l=1}^{3} a_{ijkl} (x - x_i)^k (y - y_j)^l .$$

3. На всем прямоугольнике

$$[x_0, x_m] \cdot [y_0, y_n]$$

есть непрерывные первые и вторые производные у функции $S(x, y)$.

В каждом прямоугольнике - 16 коэффициентов, всего их будет $16 \cdot m \cdot n$. Отыскиваются они из тех же соображений, что и выше для одномерного случая.

Достоинства сплайн-аппроксимаций:

1. Полученные аппроксимации функций дают графики (в пространственном трехмерном случае это уже не кривые, а поверхности), проходящие через все заданные точки, причем строятся эти кривые (поверхности) однозначно.

2. Для решения линейных систем относительно искомых коэффициентов есть эффективные методы, тем более что системы достаточно просты.

Недостатки сплайн-аппроксимаций:

1. При изменении положения хотя бы одной точки нужно пересчитывать все - что не всегда и нужно на практике. Более того, поскольку некоторые величины на практике всегда определяются с известной погрешностью, строить поверхность или кривую, строго проходящую через все точки, зачастую нецелесообразно.

2. Интерполирующие поверхности могут быть сильно осциллирующими при значительном числе точек. Это означает, что производные в любой точке – а их наличие является одной из целей построения гладкой кривой или поверхности – не отражают общего хода кривой, или общих закономерностей, которые получаются исследованием производных функции. В этом смысле гладкая кривая становится мало информативной.

От этих недостатков свободны некоторые из методов сглаживания - далее будем их рассматривать, расширив класс объектов, в которых ведется поиск соответствующих кривых и поверхностей.

3.5 Сглаживающие кривые Безье

Как уже отмечалось выше, сглаживающая кривая не должна (можно сказать точнее «не обязана») проходить через заданные точки, в отличие от интерполирующей кривой. Тем не менее, для нее необходимо иметь некоторый алгоритм построения, по возможности определяющий такую кривую однозначно. Рассмотрим класс такого рода кривых, носящих имя Безье.

Во многих случаях используется так называемое параметрическое представление кривой. Вообще известны следующие варианты задания кривой (для простоты рассмотрим случай плоской кривой в системе координат OXY).

Наиболее привычна форма записи уравнения кривой в явной форме

$$y = y(x).$$

Неявной формой задания кривой называется уравнение вида

$$f(x, y) = 0.$$

Наконец, если уравнение кривой задано в виде

$$x = x(t), y = y(t),$$

говорят о параметрической форме записи уравнения кривой.

С формальной точки зрения все эти способы равноправны, поскольку допускают, как правило, преобразование из одной формы задания кривой в другую. Но в конкретных случаях преимущества разных форм записи могут быть значительными. Рассмотрим достаточно простую кривую – окружность радиуса R, заданную уравнением

$$x^2 + y^2 = R^2.$$

Это вторая из указанных выше форм задания кривой – в неявном виде. Преобразование этого уравнения к явной форме даст соотношение

$$y = \pm\sqrt{R^2 - x^2}$$

Знаки « + » и « – » перед радикалом определяют соответственно верхнюю и нижнюю части окружности, и для определенности выбора той или иной части и соответствующего знака нужно формулировать дополнительное условие.

В то же время можно задать уравнение окружности в параметрическом виде:

$$x = R\cdot\cos t, \quad y = R\cdot\sin t.$$

Очевидно, что, возведя первое и второе соотношения в квадрат и суммируя, получим исходную запись уравнения окружности, далее можно перейти и к явной форме. Но преимущества параметрического представления очевидны: каждому конкретному значению параметра t отвечает вполне определенная точка окружности. Если рассмотреть более сложные кривые, например, типа спиралей, когда одному значению x отвечает множество значений y (иногда теоретически бесконечное), то для них параметрическое представление становится единственно приемлемым.

Определение

Параметрически заданной кривой называется множество точек $M(x, y, z)$, для которых

$$x = x(t), \quad y = y(t), \quad z = z(t), \quad a <= t <= b, \quad (*)$$

где $x(t), y(t), z(t)$ – непрерывные на отрезке $[a, b]$ функции.

Далее будем принимать, что $a = 0, b = 1$, т.е. параметр t меняется в пределах от 0 до 1. Этого всегда можно добиться заменой параметра t на u по соотношению

$$u = (t - a) / (b - a),$$

и далее роль параметра t будет играть величина u, меняющаяся уже в пределах от 0 до 1. При $t = a$ будет $u = 0$, при $t = b$ соответственно $u = 1$.

Иногда три соотношения (*) заменяют одним равенством векторного вида

$$r = r(t),$$

причем вектор $r(t)$ имеет составляющие $x(t), y(t), z(t)$, что записывается в виде

$$r(t) = (x(t), y(t), z(t)).$$

Параметр t задает порядок прохождения точек пространства при монотонном изменении t; говорят, что тем самым задается ориентация параметризованной кривой.

Определение

Кривая γ называется регулярной, если в каждой ее точке существует касательная, определяемая единственным образом, и ее ориентация меняется непрерывно по мере движения точки вдоль кривой.

По существу это определение гладких кривых, т.е. кривых, не имеющих угловых и других особых точек. При этом самопересекающиеся кривые могут быть регулярными, лишь бы при изменении параметра t направление движения вдоль кривой в точке пересечения определялось однозначно.

Далее будем говорить о регулярных кривых. Для них, в соответствии с данным выше определением, можно всегда построить касательную в любой точке однозначно ориентированную, причем ориентация касательной определяется ее единичным вектором по формуле

$$\bar{T}(t) = \frac{\bar{r}'(t)}{\left|\bar{r}'(t)\right|}$$

В этой формуле надчерком обозначаются векторные величины.

Далее рассмотрим плоские кривые на плоскости OXY. Для них, естественно, справедливы все утверждения и определения, сделанные выше для общего пространственного случая. И обратно, все дальнейшие рассуждения справедливы для пространственного случая, поскольку рассматривается плоский случай для простоты и наглядности.

Пусть на этой плоскости задан набор точек V_0, V_1, …, V_m. Индексы обозначают номера точек, причем нумерация определяется сразу и далее не меняется. В соответствии с этой нумерацией последовательно соединим точки отрезками и получим ломаную линию. Важно отметить, что соединение точек идет именно в соответствии с их нумерацией. Если поменять номера двух или любого другого числа точек, вид ломаной изменится, но и сглаживающая кривая тогда станет иной. Эта линия для заданного набора или массива точек называется **контрольной ломаной** и обозначается как

$$V = V[V_0, V_1, \ldots, V_m].$$

При этом каждая из точек V_i определяется парой чисел на плоскости – своими координатами:

$$V_i = V_i\,(x_i, y_i).$$

Зададимся задачей: построить сглаживающую кривую для массива точек V. Рассмотрим один из наиболее распространенных алгоритмов построения сглаживающих кривых, который связывается с именем Безье.

Определение:

Кривая Безье для массива точек V определяется уравнением

$$r(t) = \sum C_m^i t^i (1-t)^{m-i} V_i, \qquad 0 \le t \le 1, \qquad (*)$$

Здесь коэффициенты определяются так же, как коэффициенты в разложении бинома Ньютона:
число сочетаний из m по i. При вычислении этих коэффициентов нужно помнить, что

$$1! = 0! = 1.$$

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

Кривая Безье, которую можно теперь построить на основе данного определения, обладает следующими свойствами.

1. Кривая гладкая, что означает, как уже отмечалось, наличие касательной в каждой ее точке, определенной однозначно.

2. Кривая начинается в точке V_0 и заканчивается в точке V_m, при этом она касается отрезка V_0V_1 на выходе из первой точки, и касается отрезка $V_{m-1}V_m$ при подходе к последней.

3. Кривая Безье всегда лежит внутри оболочки – ломаной (контрольной ломаной).

Последнее свойство является следствием того, что функциональные коэффициенты в (*)

$$C_m^i t^i (1-t)^{m-i}$$

представляют собой так называемые многочлены Бернштейна. Они неотрицательны, а сумма их равна единице.

Порядок нумерации точек очень важен для построения сглаживающей кривой Безье.

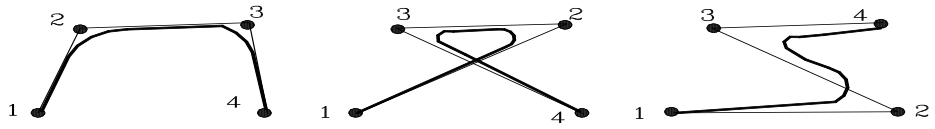


Рис. 17. Примеры различных кривых Безье для одного и того же набора точек с различной нумерацией

На рис. 17 показано влияние нумерации на вид кривых Безье. Как видно, сглаживающая кривая “пытается повторить” контрольную ломаную, причем в зависимости от порядка нумерации точек можно считать варианты сглаживания “удачными” или нет.

Недостатки кривой Безье можно сформулировать следующим образом.

1. Степень функциональных коэффициентов растет вместе с увеличением числа точек массива V_i.

2. При добавлении или изменении хотя бы одной точки кривая меняется. Для построения новой кривой требуется провести перерасчет параметрической зависимости (*).

Можно отметить, что эти недостатки «перекликаются» с аналогичными недостатками, которые отмечались при использовании интерполяционного полинома Лагранжа. Очевидно, что и преодоление их может быть сделано в известном отношении аналогичным способом. Если интерполяционные кривые строились в виде отдельных полиномиальных зависимостей невысокой степени, и потом соответствующие кривые «сшивались» между собой, то и при построении сглаживающих кривых решение следует искать на этом же пути. Один из способов решения заключается в использовании так называемых базовых сплайнов или B-сплайнов (Base-spline). Проблема сводится к другому способу определения коэффициентов в зависимости типа (*).

3.6 Базовые сплайны

Для построения сглаживающих кривых введем специальные функции, так называемые базовые сплайны. Для этого сначала разобьем отрезок $0 \le t \le 1$ на участки $0 = t_0 < t_1 < t_2 < \ldots < t_{m-1} < t_m = 1$.

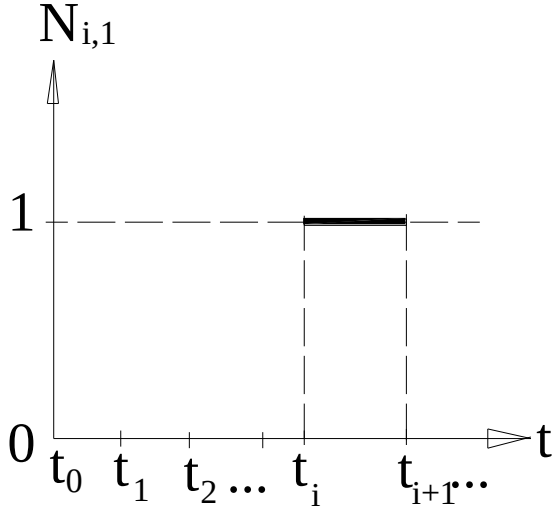


Рис.18. График функции $N_{i,1}(t)$

Первую из специальных функций (см. рис. 18) введем в виде

$$N_{i,1}(t) = \begin{cases} 1, & t_i \le t \le t_{i+1}, \\ 0, & t < t_i \ \textit{или} \ \ t > t_{i+1} \end{cases},$$

и далее остальные функции по соотношениям вида

$$N_{i,q}(t) = \frac{t - t_i}{t_{i+q-1} - t_i} \bullet N_{i,q-1}(t) + \frac{t_{i+q} - t}{t_{i+q} - t_{i+1}} \bullet N_{i+1,q-1}(t). \qquad (*)$$

Это так называемая рекуррентная форма задания функций. Для $q = 1$ график функции приведен на рис. 18 в соответствии с ее определением. Как следует из определения, в интервале от t_i до t_{i+1} эта функция – просто постоянная величина, равная единице. При $q = 2$ зависимость от параметра t становится линейной (а не постоянным значением). Рассмотрим выражение (*) более внимательно. В первом слагаемом первый сомножитель, дробь, есть не что иное, как параметрическое задание прямой (именно прямой, так как параметр t в него входит в первой степени), проходящей при $q = 2$ через точки $(t_i, 0)$, $(t_{i+1}, 1)$. Из этой прямой второй сомножитель

$$N_{i,q-1}(t) = N_{i,1}(t),$$

не равный нулю только внутри интервала $[t_i , t_{i+1}]$, «вырезает» отрезок прямой, так как вне интервала он обращает в 0 все первое слагаемое. Тот же самый результат можно получить из анализа второго слагаемого в (*). В итоге график функции $N_{i,2}(t)$ можно представить в виде, представленном на рис. 19. Он содержит два линейных участка и

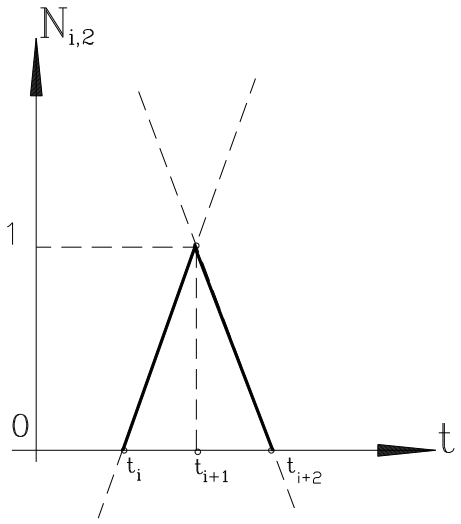


Рис. 19. Вид функции $N_{i,2}(t)$

строится по трем точкам. Если теперь построить график функции $N_{i,3}(t)$, то она будет уже квадратичной относительно параметра t, и для ее построения будут

использованы 4 точки, и т.д. Для построения функции и соответственно графика $N_{i,q}$ для любого q потребуется $q + 1$ точка, а степень полинома будет $q - 1$.

Свойства этих функций:

1. $N_{i,q}(t) > 0$ на интервале (t_i , t_{i+q}).
2. $N_{i,q}(t) = 0$ вне этого интервала.
3. На всей области задания функция $N_{i,q}(t)$ $(q > 3)$ имеет непрерывные производные до порядка q - 2 включительно.

При построении кубического сплайна должна быть использована функция $N_{i,4}(t)$, для которой потребуется пять узлов разбиения

$$t_i , t_{i+1} , t_{i+2}, t_{i+3}, t_{i+4}$$

отрезка [0, 1]. Если узлов не хватает, их набор определенным образом расширяют, например, полагая

$$t_{-3} = t_{-2} = t_{-1} = 0, \quad t_{m+1} = t_{m+2} = t_{m+3} = 1.$$

Вводимые дополнительные "отрезки" имеют нулевую длину, а первоначально первый $t = 0$ и последний $t = 1$ узлы становятся кратными.

Для введенных функций сохраняется равенство

$$\Sigma N_{i,q}(t) = 1.$$

Это значит, что кривая, заданная векторным уравнением

$$r(t) = \Sigma N_{i,4}(t) * V_i , \qquad (*)$$

всегда принадлежит выпуклой оболочке вершин массива. Кроме того, оказывается, что она выходит из точки V_0 и приходит в точку V_m , касаясь соответствующих отрезков.

Сохраняется и достаточная гладкость кривой: при $q \geq 4$ все функциональные коэффициенты имеют непрерывные вторые производные, что достаточно для большинства практических задач. Далее принимается $q = 4$.

Изменение положения одной вершины в массиве уже не приводит к полному изменению всей кривой; в силу свойств функциональных коэффициентов пересчет коснется лишь пяти слагаемых.

Определяемый соотношением (*) отрезок кривой лежит внутри их выпуклой оболочки - четырехугольника в плоском случае и тетраэдра в пространственном.

Иногда построенная кривая по каким-то соображениям не устраивает нас, и мы хотим подправить результат. Это можно сделать с помощью параметров,

вводимых в уравнение кривой. Для этого используются так называемые бета-сплайны, при этом используется несколько иной подход - с так называемыми составными кривыми.

3.7 Бета-сплайны

При построении составной регулярной кривой необходимо наряду с непрерывным переходом одного участка кривой в другой обеспечить и гладкость этого сопряжения.

Пусть γ_1 и γ_2 - регулярные кривые, заданные соответственно параметрическими уравнениями

$$r = r_1(t), \quad 0 \le t \le 1, \quad r = r(t), \quad 0 \le t \le 1$$

и имеющие общую точку

$$r_1(1) = r_2(0).$$

Для того, чтобы кривая γ, составленная из γ_1, γ_2, была регулярной, потребуем совпадения в общей точке единичных касательных векторов

$$\frac{r_1'(1)}{|r_1'(1)|} = \frac{r_2'(0)}{|r_2'(0)|},$$

и векторов кривизны сопрягаемых кривых γ_1 и γ_2. Штрих означает производную по параметру t.

Оказывается, если радиусы-векторы кривых γ_1 и γ_2 связаны условиями геометрической непрерывности

$$\begin{aligned} r_2(0) &= r_1(1), \\ r_2'(0) &= \beta_1 r_1'(1), \\ r_2''(0) &= \beta_1^2 r_1''(1) + \beta_2 r_1'(1), \end{aligned} \qquad (1)$$

где $\beta_1 > 0$, $\beta_2 \ge 0$ - числовые параметры, то каждое из условий регулярности составной кривой будет выполнено.

Рассмотрим набор из $m + 1$ точек V_0, V_1, ..., V_{m-1}, V_m, заданных своими радиус-векторами.

Будем искать сглаживающую составную регулярную кривую при помощи частичных кривых $r_i(t)$, описываемых уравнениями вида

$$r_i(t) = \Sigma\ b_j(t) * V_{ij}\ ,\ 0 \le t \le 1, \qquad (2)$$

где

$$b_j(t) = \Sigma\, C_{kj}(\beta_1, \beta_2)\, t,\ j = -2, -1, 0, 1 \qquad (3)$$

- не зависящие от i весовые функциональные коэффициенты.

Для того, чтобы найти эти весовые коэффициенты, потребуем, чтобы векторы $r_i(t)$, $r_{i+1}(t)$ в точке сопряжения удовлетворяли условиям непрерывности (1). С учетом (2) это можно записать:

$$\sum_{j=-2}^{1} b_j(0) V_{i+j+1} = \sum_{j=-2}^{1} b_j(1) V_{i+j},$$

$$\sum_{j=-2}^{1} b_j'(0) V_{i+j+1} = \beta_1 \sum_{j=-2}^{1} b_j'(1) V_{i+j}, \qquad (4)$$

$$\sum_{j=-2}^{1} b_j''(0) V_{i+j+1} = \beta_1^2 \sum_{j=-2}^{1} b_j''(1) V_{i+j} + \beta_2 \sum_{j=-2}^{1} b_j'(1) V_{i+j}$$

что позволяет найти все функциональные коэффициенты

$$b_j(t),\ j = -2, -1, 0, 1.$$

Расписав, например, первое из равенств (4) подробнее

$$b_{-2}(0)V_{i-1} + b_{-1}(0)V_i + b_0(0)V_{i+1} + b_1(0)V_{i+2} =$$

$$b_{-2}(1)V_{i-2} + b_{-1}(1)V_{i-1} + b_0(1)V_i + b_1(1)V_{i+1},$$

и приравнивая коэффициенты при одинаковых векторах, получим

$$b_{-2}(1) = 0;\ b_{-2}(0) = b_{-1}(1);\ b_{-1}(0) = b_0(1);\ b_0(0) = b_1(1);\ b_1(0) = 0.$$

Подобным образом из двух других равенств (3) получаются соотношения, связывающие значения в точках 0 и 1 первых и вторых производных весовых коэффициентов. В итоге после привлечения соотношения (3) получается линейная система для искомых чисел C_{kj}, определитель которой

$$\delta = 2\beta_1^3 + 4\beta_1^2 + 4\beta_1 + \beta_2 + 2 > 0.$$

Вычислим коэффициенты и подставим их в (3), тогда найденные выражения для весовых функций будут

$$b_{-2}(t) = \frac{2\beta_1^3}{\delta}(1-t)^3,$$

$$b_{-1}(t) = \frac{1}{\delta}\left[2\beta_1^3 t(t^2 - 3t + 3) + 2\beta_1^2(t^3 - 3t^2 + 2) + 2\beta_1(t^3 - 3t + 2) + \beta_2(2t^3 - 3t^2 + 1),\right]$$

$$b_0(t) = \frac{1}{\delta}\left[2\beta_1^2 t^2(3-t) + 2\beta_1 t(3 - t^2) + \beta_2 t^2(3-2t) + 2(1-t^3)\right],$$

$$b_1(t) = \frac{2t^3}{\delta}.$$

причем они годятся для всей конструкции. Подставляя их в (2), получаем значения векторных функций $r_i(t)$.

Чтобы составная кривая проходила через вершины V_0, V_m, касаясь отрезков V_0V_1, $V_{m-1}V_m$ контрольной ломаной, следует к полученному набору вектор-функций добавить еще четыре:

$$r_0(t) = \left(1 - \frac{2t^3}{\delta}\right)V_0 + \frac{2t^3}{\delta}V_1,$$

$$r_1(t) = \left[b_{-2}(t) + b_{-1}(t)\right]V_0 + b_0(t)V_1 + b_1(t)V_2,$$

$$r_m(t) = b_{-2}(t)V_{m-2} + b_{-1}(t)V_{m-1} + \left[b_0(t) + b_1(t)\right]V_m,$$

$$r_{m+1}(t) = \frac{2\beta_1^3}{\delta}(1-t)^3 V_{m-1} + \left[1 - \frac{2\beta_1^3}{\delta}(1-t)^3\right].$$

Тем самым искомая составная кривая γ полностью найдена.

На рис. 20 показано, как выбор параметров β_1, β_2 влияет на форму результирующей кривой.

По аналогии с кривыми можно построить и сглаживающие поверхности. При этом возникает понятие поверхности Безье и т.д.

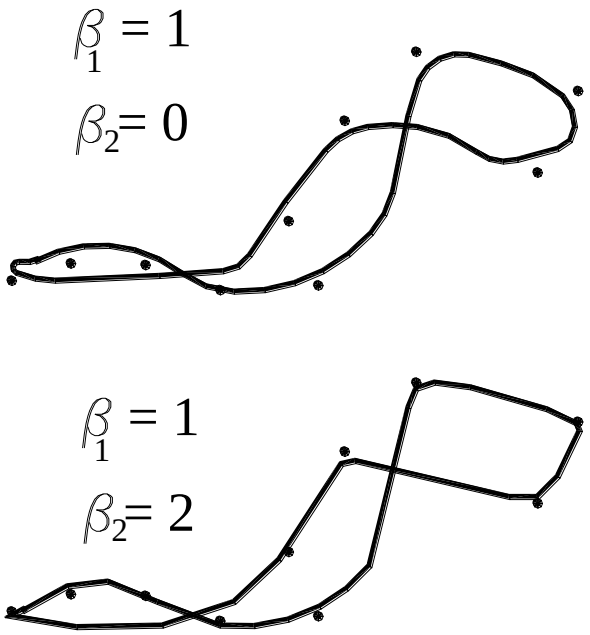


Рис. 20. Сглаживающие кривые, построенные для одного
и того же набора точек с помощью бета-сплайнов
при разных значениях параметров β_1 и β_2

3.8 Сплайновые поверхности

Регулярной называется поверхность, представляющая собой множество точек M(x, y, z), которые определяются соотношениями

$$x = x(u,v), \quad y = y(u,v), \quad z = z(u,v), \quad (u,v) \in D$$

где величины x, y, z – гладкие функции своих аргументов (параметров). При этом требуется выполнение соотношения

$$rang\begin{pmatrix} x_u(u,v) & y_u(u,v) & z_u(u,v) \\ x_v(u,v) & y_v(u,v) & z_v(u,v) \end{pmatrix} = 2;$$

D – некоторая область значений параметров u, v. Равенство, приведенное выше, означает, что в каждой точке поверхности существует единственная касательная плоскость, и эта плоскость при непрерывном перемещении точки касания поворачивается непрерывно.

Такая форма записи уравнения поверхности называется параметрической. Уравнение поверхности можно записать и в векторном виде

$$r = r(u,v), \quad (u,v) \in D,$$
$$r(u,v) = (x(u,v), y(u,v), z(u,v)).$$

Далее для простоты считаем, что параметры u, v меняются в пределах от 0 до 1.

Рассмотрим в пространстве набор точек вида

$$V_{ij}, \quad i = 0,1,...,m; \quad j = 0,1,...,n.$$

Соединяя эти точки прямолинейными отрезками в порядке нумерации, получим контрольный многогранник (или опорный граф) заданного массива точек V (рис. 21).

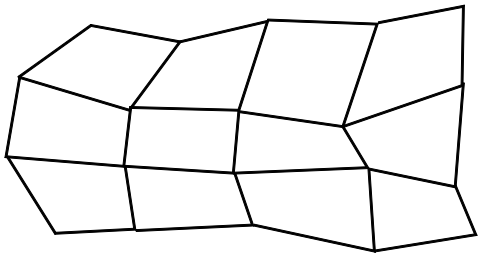


Рис. 21

После этого сглаживающая поверхность строится относительно просто – в виде так называемого тензорного произведения, которое можно записать в параметрическом виде:

$$r(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} a_i(u) b_j(v) * V_{ij},$$

$$\text{где} \quad \alpha \le u \le \beta, \quad \gamma \le v \le \delta.$$

Если приведенное выше уравнение переписать в следующей форме

$$r(u,v) = \sum_{i=0}^{m} a_i(u) r_i(v),$$

$$\text{где} \quad r_i(v) = \sum_{j=0}^{n} b_j(v) V_{ij}, \quad i = 0,1,...,m,$$

то становится возможным перенести на двумерный случай свойства, которые были отмечены при построении кривых. То есть построенные таким образом поверхности будут «унаследовать» многие свойства одноименных кривых. В этом и заключается преимущество способа построения поверхности в виде тензорного произведения.

Следует отметить, что построение поверхностей (а не кривых), или повышение размерности задачи по пространственным переменным, может

породить большое число новых проблем. Поэтому указанная выше общность в свойствах кривых и поверхностей связана с тем, что в качестве опоры для строимых поверхностей берутся наиболее простые наборы точек.

3.8.1. Поверхности Безье

Что касается построения сглаживающих поверхностей Безье, начнем, как и в случае кривых, с описания уравнений отдельных частей их.

Ограничимся бикубическим случаем. Именно такие сплайновые поверхности наиболее употребительны и удобны для применения в задачах компьютерной графики и в прикладных исследованиях. В этом случае функциональные коэффициенты представляют собой многочлены третьей степени относительно соответствующих переменных (кубические полиномы). Для заданного набора из 16 точек V_{ij}, i = 0, 1, 2, 3, j = 0, 1, 2, 3 запишем параметрические уравнения элементарных фрагментов (участков) поверхностей, считая, что параметры u, v меняются в пределах от 0 до 1.

Начнем с элементарной бикубической поверхности Безье. Параметрические уравнения участка этой поверхности имеют следующий вид:

$$r(u,v)=\sum_{i=0}^{3}\sum_{j=0}^{3}C_3^iC_3^ju^i(1-u)^{3-i}v^j(1-v)^{3-j}V_{ij},$$

$$0\le u\le 1,\quad 0\le v\le 1,$$

Или в матричной форме

$$\begin{pmatrix}x(u,v)\\ y(u,v)\\ z(u,v)\end{pmatrix}=(1\ u\ u^2\ u^3)M^T\begin{pmatrix}V_{00}&V_{01}&V_{02}&V_{03}\\ V_{10}&V_{11}&V_{12}&V_{13}\\ V_{20}&V_{21}&V_{22}&V_{23}\\ V_{30}&V_{31}&V_{32}&V_{33}\end{pmatrix}M\begin{pmatrix}1\\ v\\ v^2\\ v^3\end{pmatrix}.$$

Здесь

$$M=\begin{pmatrix}1&-3&3&1\\ 0&3&-6&3\\ 0&0&3&-3\\ 0&0&0&1\end{pmatrix}$$

- так называемая базисная матрица Безье; знаком T обозначена операция транспонирования.

Свойства элементарной бикубической поверхности Безье (они во многом наследуют свойства кривой Безье):

- поверхность лежит в выпуклой оболочке порождающих ее точек;
- гладкая;
- упирается в точки V_{00}, V_{30}, V_{03}, V_{33}, касаясь исходящих из них отрезков контрольного (опорного) графа.

Из элементарных участков поверхностей Безье можно построить далее составные поверхности. При этом возникает вопрос о гладкости таких составных бикубических поверхностей.

Из таких элементарных кусочков поверхности можно строить теперь составную поверхность – по аналогии с кривыми в одномерном случае.

Рассмотрим условия гладкости таких составных поверхностей Безье.

Пусть имеются две элементарные бикубические поверхности. Уравнение первой определяется заданием

$$r^{(1)}(u,v), \quad 0 \le u \le 1, \quad 0 \le v \le 1,$$

а второй

$$r^{(2)}(u,v), \quad 0 \le u \le 1, \quad 0 \le v \le 1.$$

Эти уравнения порождены наборами точек

$$V_{ij}^{(1)}, \quad i,j = 0,1,2,3; \qquad V_{ij}^{(2)}, \quad i,j = 0,1,2,3$$

соответственно. Если при этом выполняется условие

$$V_{3j}^{(1)} = V_{0j}^{(2)},$$

то это означает, что эти кусочки поверхностей имеют общую граничную кривую.

Поверхность, составленная из этих двух кусочков, будет иметь общую непрерывную касательную плоскость, если каждая тройка точек вида

$$V_{2j}^{(1)}, \quad V_{3j}^{(1)} = V_{0j}^{(2)}, \quad V_{1j}^{(2)}$$

лежит на одной прямой, а отношения

$$\frac{\left|V_{2j}^{(1)}V_{3j}^{(1)}\right|}{\left|V_{0j}^{(2)}V_{1j}^{(2)}\right|}$$

не зависят от j.

3.8.2. Бикубические B-сплайновые поверхности

Начнем с векторного параметрического уравнения элементарного участка бикубической B-сплайновой поверхности, которая строится по 16 опорным точкам

$$V_{ij}, \quad i, j = 0,1,2,3.$$

Это уравнение имеет вид

$$r(u,v) = \sum_{i=0}^{3}\sum_{j=0}^{3} n_i(u)n_j(v)V_{ij}, \quad 0 \le u \le 1, \quad 0 \le v \le 1,$$

где функциональные коэффициенты n_0, n_1, n_2, n_3 **определяются равенствами**

$$n_0(u) = \frac{(1-u)^3}{6}, \quad n_2(u) = \frac{3u^3 - 6u^2 + 4}{6},$$

$$n_2(u) = \frac{-3u^3 + 3u^2 + 3u + 1}{6}, \qquad n_3(u) = \frac{u^3}{6}.$$

В матричной форме это же уравнение можно записать в виде

$$r(u,v) = U^T M^T W M V, \qquad 0 \le u, v \le 1.$$

В этой записи

$$r(u,v)=\begin{pmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{pmatrix}, \quad U=\begin{pmatrix} 1 \\ u \\ u^2 \\ u^3 \end{pmatrix}, \quad V=\begin{pmatrix} 1 \\ v \\ v^2 \\ v^3 \end{pmatrix},$$

$$W=\begin{pmatrix} V_{00} & V_{01} & V_{02} & V_{03} \\ V_{10} & V_{11} & V_{12} & V_{13} \\ V_{20} & V_{21} & V_{22} & V_{23} \\ V_{30} & V_{31} & V_{32} & V_{33} \end{pmatrix}.$$

Через M по-прежнему обозначена базисная матрица кубического B-сплайна.

Элементарная бикубическая B-сплайновая поверхность наследует многие свойства элементарной кубической B-сплайновой кривой:

- гладкая;
- лежит в выпуклой оболочке порождающих ее 16 вершин;
- «повторяет» контрольную многогранную поверхность.

Построение составной бикубической B-сплайновой поверхности на прямоугольнике

$$[0,m]\times[0,n]$$

с равномерными узлами проводится аналогично одномерному случаю.

IV Растровые алгоритмы

4.1 Основные понятия

Пусть на плоскости есть сетка с единичным шагом, причем ее узлы - центры соответствующих единичных квадратов. Когда такая сетка является регулярной, она может служить моделью растра, с помощью которого строятся изображения. Дадим ряд определений, которые нам понадобятся в дальнейшем.

Точки называются 4-соседями, если у них отличаются только x или только y на единицу. Это так называемые непосредственные соседи, для точки (квадрата) с номером 1 (на схеме ниже) это будут соответственно точки или квадраты с номерами 2, 3, 4, 5.

Точки называются 8-соседями или косвенными соседями, если у них отличаются x или y не более чем на единицу.

7	2	6
3	1	5
8	4	9

Для точки 1 это все окружающие точки с номера 2 по номер 9. Таким образом, непосредственный сосед - еще и косвенный, но не наоборот.

У непосредственных соседей всегда есть общая сторона, у косвенных соседей общей может быть как сторона, так и вершина.

Сильносвязным путем (4-путем) называется множество точек A_1, A_2, ..., A_n, для которых любые точки A_i, A_{i+1} являются непосредственными соседями при $i = 1, 2, ..., n - 1$.

Слабосвязным путем (8-путем) называется множество точек A_1, A_2, ..., A_n , для которых A_i , A_{i+1} - слабые соседи при $i = 1, 2, ..., n - 1$.

Если $A_1 = A_n$, т.е. последняя точка совпадает с первой, путь называется замкнутым.

Если любые две точки множества можно связать сильносвязным путем, это множество - сильносвязное.

В случае слабосвязного пути соответственно называется и тип связанного им множества точек.

Простой кривой на плоскости называется множество точек, в котором все точки, за исключением двух, имеют ровно двух соседей, а эти
две – по одному соседу.

В этом определении две исключенные точки – не что иное, как начальная и конечная точки кривой.

Простой замкнутой кривой на плоскости называется множество точек, в котором все точки имеют ровно двух соседей. Таким образом, под это определение не попадают самопересекающиеся кривые.

В геометрии сформулированы и доказаны два следующих утверждения:

1. Простая замкнутая слабосвязная кривая разбивает плоскость на два сильносвязных множества.

2. Простая замкнутая сильносвязная кривая разбивает плоскость на два слабосвязных множества.

Растром называется целочисленная решетка на плоскости. Таким образом, это определение включает в себя понятие технического растра, реализованного на экране монитора в виде набора пикселей, как частный случай. Важно, что каждому элементу растра можно поставить в соответствие целое число (номер).

Растровое представление объекта – представление геометрической фигуры в виде множества элементов растра на целочисленной плоскости.

Такое представление не является однозначным. Любую фигуру можно изобразить различными способами. Представьте, что перед вами лежит лист бумаги в клетку, и ваша задача – изобразить некоторую фигуру, например, треугольник, с помощью закрашивания клеток, которые в данном случае и представляют собой модель растра. В группе студентов – это можно с уверенностью утверждать – выполнение такого задания будет порождать разные варианты. Но для программы компьютера никакой неоднозначности не может и не должно быть. В дальнейшем будем считать, что нужно разработать алгоритм выбора последовательности точек, которые нужно инициализировать на растре, чтобы получить растровое представление объекта в целом.

Существуют стандартные процедуры генерации отрезка, дуги, эллипса, закраски многоугольников и т.д. – для всех примитивов.

Знание проблем растровой генерации полезно в следующих случаях:

- при наличии (или задании) зависимости атрибутов пиксела от каких-либо условий, например, от положения пикселя на границе или внутри многоугольника;
- при необходимости изменить структуру соответствующего алгоритма для ускорения его работы.

4.2 Растровая развертка отрезка. Алгоритм Брезенхема

Растровая развертка отрезка - это процесс последовательной инициализации множества пикселей экрана, изображающего отрезок. Можно отметить, что растровое изображение отрезка довольно очевидно для трех случаев – когда он ориентирован строго горизонтально, вертикально или под углом 45^o к этим направлениям. Тогда на квадратной сетке элементов растра никаких проблем с выбором элементов, которые следует «инициализировать» (т.е. заставить изменить цвет и/или яркость, чтобы получить изображение на общем фоне экрана), не возникает. В общем случае, при рисовании отрезка произвольной ориентации, дело обстоит сложнее.

Если даже ограничить класс растровых представлений простыми растровыми кривыми, то уже возможны два типа представлений - 8- и 4-связные (или слабо- и сильносвязные).

Рассмотрим так называемое «простое» решение задачи.

Чтобы выбрать промежуточные пиксели (точки растра), которые наименее удалены от идеального изображаемого отрезка, можно, например, последовательно инициализировать все точки растра, окрестности которых пересекаются с этим отрезком.

Можно предложить следующий алгоритм последовательной генерации точек.

Если $M_1(x_1, y_1)$, $M_2(x_2, y_2)$ - границы отрезка, то его уравнение:

$$y = y_1 + k*(x - x_1), \qquad x_1 \le x \le x_2, \qquad k = \frac{y_2 - y_1}{x_2 - x_1},$$

причем далее принимаем $0 \le k < 1$. То, что это уравнение прямой, очевидно. При этом соответствующая прямая проходит через заданные точки: при $x = x_1$ будет $y = y_1$, а при $x = x_2$ получим $y = y_2$.

Процедура генерации растровой развертки (здесь и далее в таких случаях используется так называемый псевдоалгол, или учебный алгол, для записи алгоритмов) может быть записана в виде следующего алгоритма. Принимаем, что имеется программа *PutPixel(x, y, ...),* которая инициализирует пиксель с координатами x, y, а остальные параметры программы, наличие которых обозначено троеточием, определяют цвет и/или яркость (атрибуты) соответствующего пикселя. Тогда нужная программа рисования отрезка представляется в виде

```
 Dx := 1; Dy := abs((y2 - y1)/(x2 - x1));
X := x1; y := y1; L := x2 - x1;
for i := 0 to L - 1 do begin
PutPixel(x, Round(y));
X := x + Dx; y := y + Dy; end.
```

Работа этого алгоритма происходит следующим образом: начинаем с начальной точки (x_1, y_1), «рисуем» ее, инициализируя соответствующий пиксель. Затем получаем очередную точку на отрезке и инициализируем пиксель, ближайший к ней.

Условие $k < 1$ необходимо, чтобы при построении растровой развертки не пропустить ни одной точки: по координате x шагаем с шагом $Dx = 1$, по y с меньшим шагом, округляя значения y до целых значений. Если $k > 1$, то в этом алгоритме нужно поменять местами переменные x и y.

В записанном алгоритме целочисленная абсцисса точки изменяется на каждом шаге на единицу, а целочисленная ордината изменяется лишь в том случае, когда при накоплении приращений Dy ордината точки окажется в окрестности 0.5 соседнего уровня по оси ординат.

С учетом этого наблюдения можно несколько изменить алгоритм по форме без изменения основной схемы:

```
x := x1; y := y1; n := x2 - x1; m:= y2 - y1; d := m/n; e := 0;
 for i := 1 to n do begin {шаг по x и вычисление отклонения}
 x := x + 1; e := e + d;
```

{если отклонение по оси ординат от текущего значения y больше 1/2, то нужно увеличить y на 1 и скорректировать величину e от нового значения y}

```
 if e  > 0.5 then begin y := y + 1; e := e - 1; end;
PutPixel(x,y);
end;
```

В этом случае получается 8-связное представление отрезка, т.к. переход к следующей точке осуществляется на одну из соседних 8 клеток.

Более общие алгоритмы такого рода (в частности, без ограничения $k < 1$) для 8- и 4-связных разверток называются алгоритмами Брезенхема.

4.3 Отсечение отрезка. Алгоритм Сазерленда-Кохена

Необходимость отсечь часть выводимого изображения встречается достаточно часто. Например, при пользовании командой «Обрежь» или «Рамка». В последнем случае отсечение проводится по всем границам области – прямоугольника (рамки).

Ниже рассматривается простой и эффективный алгоритм отсечения отрезков на границе произвольного прямоугольника. Он заключается в разбиении всей плоскости, на которой построено изображение (плоскости экрана) на 9 областей четырьмя прямыми, образующими прямоугольник. В каждой из этих областей все точки по отношению к прямоугольнику расположены одинаково. Если мы определим, в какие области попадают концы произвольного отрезка, то легко поймем, где необходимо отсечение. Для этого каждой области присваивается код (4-битовый), который имеет значения:

0 – для точек левее, в двоичном представлении 00
1 – выше, 01
2 – правее, 10
3 – для точек ниже прямоугольника, 11

Работа алгоритма сводится к тому, что любой точке отрезка в зависимости от ее координат присваивается признак как произведение кодов, указанных выше для внешних по отношению к прямоугольнику (рамке) полей. Если полученный признак не совпадает с признаком, которым можно характеризовать внутренность прямоугольника, то соответствующая точка отрезка находится вне рамки, и ее не нужно изображать. После работы такого алгоритма изображение строится только для той части отрезка, которая попала внутрь рамки.

Обобщение работы этого алгоритма на другие геометрические объекты сводится к тому, что проверке на попадание внутрь или вне рамки надо проверять все точки (элементы растра), с помощью которых построено растровое представление этого объекта. Поэтому если процедура «Обрежь» иногда выдает диагностику «Не могу обрезать этот примитив» (например, текст или штриховку), то это связано с тем, что она работает не с растровым представлением объекта, а с его векторным представлением (математическим описанием). Процедура «Рамка» в этом отношении универсальна, и при попадании в рамку части текста или штриховки она работает уже с растровым представлением объекта.

4.4 Тест принадлежности точки многоугольнику

Многоугольник – фигура, ограниченная на плоскости простой (несамопересекающейся) ломаной замкнутой линией.

Ломаная задается своими вершинами $A_i(x_i, y_i)$, $i = 1, 2, ..., n$.

Соседние точки с номерами i и $i + 1$ - смежные вершины.

Задача: получить растровую развертку многоугольника, т.е. провести инициирование его внутренних точек. Для начала следует решить следующую задачу: если задана произвольная точка $A(x, y)$, нужно определить, принадлежит она данному многоугольнику или нет.

Теорема Жордана:

Простая замкнутая плоская ломаная разбивает плоскость на две связные компоненты: 1) ограниченную область, или внутренность многоугольника; 2) неограниченную внешнюю часть.

Эта теорема дает основание считать, что любые из рассмотренных далее алгоритмов закрашивания многоугольников будут работать ограниченное время, так как в них рассматривается всегда конечное число элементов растра, отвечающих внутренности многоугольника.

Алгоритм должен отличать внутренние и внешние точки.

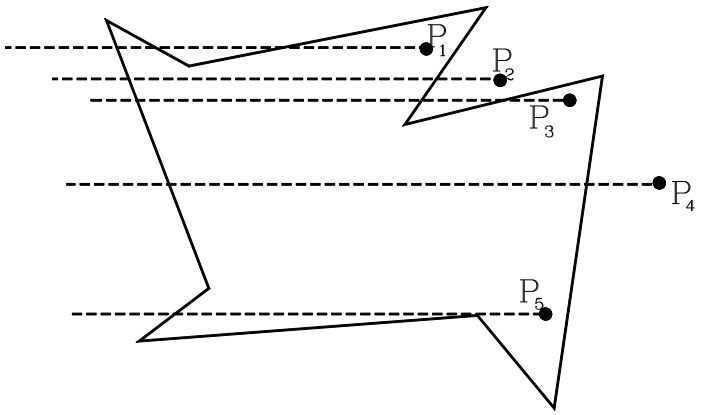


Рис. 22

Обозначим ребра многоугольника $E_i : [A_i , A_{i+1}]$, $i = 1, 2, ..., n$ (при этом вершина с номером $n + 1$ совпадает с вершиной номер 1).

Пусть $P_i(x, y)$ - некоторая точка плоскости, не принадлежащая ломаной (границе многоугольника), см. рис. 22. Нужно определить, находится ли она внутри многоугольника.

Из точки P_i ведем горизонтальную полупрямую влево (т.е. точка P_i - правый конец полупрямой). При достаточно большом удалении конечной (условно конечной, так как полупрямая теоретически бесконечна) точки этой полупрямой Q от точки P_i варианты:

- нет пересечений полупрямой с границей многоугольника, тогда P_i - внешняя точка;
- есть четное число пересечений полупрямой с границами многоугольника; тогда P_i – внешняя для этого многоугольника точка;
- есть нечетное число пересечений; P_i - внутренняя точка.

Пересечение называется существенным, если P_iQ пересекает ломаную, а не касается одной из вершин.

Правила:

1) пересечения отрезка с горизонтальными ребрами игнорируются;

2) пересечение игнорируется, если точкой пересечения является вершина ребра, и засчитывается в любом другом случае. Т.е. в точках максимума пересечение не считается, а в точках минимума считается дважды; при этом четность не нарушается.

4.5 Алгоритм определения точки пересечения произвольного луча с геометрическими объектами

Для исключения лишней работы при анализе сцены желательно определить лучи, идущие из центра проектирования на картинную плоскость, которые вообще не пересекаются с объектами. Это означает, что в соответствующих точках картинной плоскости будет изображен фон.

Один из приемов - поместить объект произвольной формы внутрь сферы или многогранника, и далее из анализа исключить те лучи, что не пересекают их. Что касается поиска пересечения со сферой или параллелепипедом, то ниже показано, что это можно сделать с помощью относительно простых алгоритмов.

Только после этого для оставшихся лучей имеет смысл проводить более детальный анализ.

4.5.1 Пересечение луча со сферой

А. Аналитическое (алгебраическое) решение

Пусть луч выходит из точки О, радиус вектор которой $R_0\ (x_0, y_0, z_0)$, в направлении, определяемом вектором L $(l, m, n) \neq 0$. Здесь l, m, n – косинусы углов, которые образует вектор, направленный вдоль луча, с осями декартовой системы координат, или проекции соответствующего единичного вектора на эти оси.

Параметрическое уравнение такого луча можно записать в виде

$$R(t) = R_0 + L^* t, \qquad t > 0,$$

или

$$x(t) = x_0 + l^*t,\ y(t) = y_0 + m^*t,\ z(t) = z_0 + n^*t \qquad (*)$$

Если направляющий вектор единичный ($l^2 + m^2 + n^2 = 1$), то смысл параметра t - расстояние от начала луча до текущей точки.

Пусть имеется сфера радиуса r, центр которой расположен в точке с координатами x_c, y_c, z_c. Ее уравнение тогда можно записать:

$$(x - x_c)^2 + (y - y_c)^2 + (z - z_c)^2 = r^2.$$

Точки пересечения луча со сферой будут, если x, y, z одновременно принадлежат лучу и сфере, т.е. удовлетворяют соответственно уравнениям луча и сферы одновременно.

Подставим (*) в уравнение сферы; тогда получим

$$at^2 + 2bt + c = 0 \qquad (**)$$

Здесь $a = l^2 + m^2 + n^2 = 1$ (далее для единичного направляющего вектора L);

$$b = l^*(x_0 - x_c) + m^*(y_0 - y_c) + n^*(z_0 - z_c);$$
$$c = (x_0 - x_c) + (y_0 - y_c) + (z_0 - z_c) - r\ .$$

Если существуют значения t, при которых выполняется (**), это и есть точки пересечения.

Корни уравнения (**) $\quad t = - b \pm \sqrt{b^2 - c}$

Если $b^2 - c < 0$, решения нет, и это означает, что луч идет мимо сферы;

$b^2 - c = 0$ - одно решение – луч касается сферы;

$b^2 - c > 0$ - два решения – луч "протыкает" сферу.

Поскольку мы принимали $t > 0$, то ближайшая точка пересечения отвечает меньшему положительному корню (**); обозначим ее через t_1. Тогда точка пересечения определяется координатами

$$x = x_0 + lt_1,\ y = y_0 + mt_1,\ z = z_0 + nt_1.$$

Нормаль к сфере - единичный вектор - определится тогда соотношением

$$N = (x - x_c, y - y_c, z - z_c) / r.$$

Таким образом, весь алгоритм включает в себя расчет

1) a, b, c из соотношения (*);
2) $b^2 - c$;
3) $t_{\min}$ и $t_{\max}$;
4) t_1, выбираемое из $t_{\min}$ и $t_{\max}$; определяется точка пересечения;
5) расчет N.

Б. Геометрическое решение

Существует значительное число тестов, с помощью которых можно определить, пересекает ли данный луч сферу или не пересекает. Их использование позволяет избежать хотя и несложных, но временами громоздких вычислений, рассмотренных, например, выше.

Для того, чтобы определить, лежит начальная точка луча внутри сферы или нет, достаточно вычислить длину вектора от этой начальной точки O до центра сферы C. Если эта длина меньше радиуса сферы (OC < R) – начальная точка луча лежит внутри сферы. При равенстве OC = R начальная точка лежит на сфере, иначе, при OC > R – вне сферы.

В любом случае следующий шаг заключается в определении расстояния от начальной точки луча до точки на луче, ближайшей к центру сферы. Очевидно, такой точкой будет точка пересечения луча с перпендикулярной ему плоскостью, проходящей через центр сферы.

Получение такой точки эквивалентно проектированию вектора, соединяющего начало луча и центр сферы, на луч. Проекция при этом получается как скалярное произведение, а координата t^0 получается

$$t^0 = (OC * L).$$

Если при этом получается значение $t^0 < 0$, это означает, что центр сферы находится «позади» начальной точки луча.

Для лучей, исходящих из точек внутри сферы, это несущественно – в любом случае луч пересечет сферу. Если же луч начинается вне сферы, то для таких «точек старта» пересечения со сферой не будет, и в этом случае тестирование можно завершить.

Зная длину вектора

$$OC = sqrt(OC*OC)$$

и величину проекции этого вектора на луч

$$sqrt(OC*L),$$

по теореме Пифагора можем получить квадрат расстояния от центра сферы до луча:

$$d^2 = (OC*OC) - (OC*L)^2,$$

а затем разность

$$t^2 - d^2.$$

Если эта разность отрицательна, то луч проходит мимо сферы (разумеется, при этом начальная точка луча находится вне сферы). Если эта разность положительна, то остается найти значения параметра t, отвечающие пересечениям луча со сферой. Поскольку точка на луче, ближайшая к центру, нам уже известна (это значение t^0), то для луча, исходящего из точки вне сферы, ближайшая точка пересечения со сферой отвечает минимальному значению t:

$$t^* = t^0 - \sqrt{R^2 - d^2}.$$

Если же начальная точка луча находится внутри сферы, то единственное пересечение его со сферой определяется значением

$$t^* = t^0 + \sqrt{R^2 - d^2}.$$

Шаги соответствующего алгоритма можно записать в следующей последовательности:

1. Определяется квадрат расстояния между начальной точкой луча и центром сферы.
2. Вычисляется квадрат расстояния от центра сферы до луча.
3. Выясняется, лежит ли луч вне сферы.
4. Находится разность квадратов радиуса сферы и кратчайшего расстояния.
5. Определяется знак этой разности.
6. Определяется расстояние до ближайшей точки пересечения со сферой.

7. Находится точка пересечения со сферой.
8. Определяется (при необходимости) единичный вектор нормали к сфере в этой точке.

Выбор алгебраического или геометрического способов решения задачи о пересечении луча со сферой – в известном отношении дело вкуса пользователя. Более того, можно наверняка придумать новые способы решения этой задачи.

4.5.2. Пересечение луча с плоскостью

Пусть уравнение плоскости

$$ax + by + cz + d = 0;$$

тогда $N(a, b, c)$ - вектор нормали к ней. Если при этом N - единичный вектор ($a^2 + b^2 + c^2 = 1$), то d в уравнении плоскости означает расстояние от начала координат до плоскости.

Пересечение луча с плоскостью будет, если вместо x, y, z в уравнение плоскости подставим (*):

$$x = x_0 + lt, \quad y = y_0 + mt, \quad z = z_0 + nt$$

Тогда получается линейное относительно параметра t уравнение:

$$a*(x_0 + lt) + b*(y_0 + mt) + c*(z_0 + nt) + d = 0.$$

Решение этого уравнения

$$t_1 = -\frac{ax_0 + by_0 + cz_0 + d}{al + bm + cn}. \qquad \text{(a)}$$

Если знаменатель в этом выражении равен нулю, то $t_1 \to \infty$, а это означает, что луч параллелен плоскости и ее не пересекает.

Если $t_1 < 0$ - пересечения луча с плоскостью нет;

если $t_1 > 0$ - есть пересечение в этой точке, координаты которой определяются из соотношений

$$x = x_0 + lt_1, \; y = y_0 + mt_1, \; z = z_0 + nt_1.$$

Вектор нормали к плоскости в точке пересечения ее лучом выбирается так, чтобы угол между ним и направляющим вектором луча был тупым. Это значит, что вектор нормали находится в том же полупространстве, что и начальная точка луча.

Алгоритм расчета состоит из 4-х шагов:

1. Вычисляем знаменатель в выражении (а)
2. Сравниваем t_1 с нулем.
3. Определяем точку пересечения луча с плоскостью.
4. Определяем нужное нам направление нормали.

4.5.3. Пересечение луча с выпуклым многоугольником

Выпуклый n - угольник определяется однозначно набором своих вершин

$$x_i,\ y_i,\ z_i \quad (i = 1,2,...,n)$$

Считаем, что нумерация сделана так, что соседние по номеру вершины примыкают к одной стороне многоугольника.

При определении пересечения луча с многогранником определяется возможность его пересечения с каждой гранью - многоугольником.

Для этого определяется:

1) пересекает ли луч плоскость многоугольника;
2) попадает ли точка пересечения внутрь многоугольника.

О первом вопросе – о пересечении луча с плоскостью – речь уже шла выше. Ясно, что при отсутствии такого пересечения вопрос о попадании луча внутрь многоугольника снимается.

Пусть луч пересекается с плоскостью многоугольника

$$ax + by + cz + d = 0$$

в некоторой точке с координатами

$$(x^*,\ y^*,\ z^*).$$

Исключим из рассмотрения случай, когда эта точка попала на границу многоугольника. Тогда по существу осталось определить, лежит ли эта точка внутри многоугольника. Для решения этого вопроса и многоугольник, и точка пересечения проектируются на какую-либо координатную плоскость, например, $z = 0$. Если проекция точки пересечения находится внутри проекции многоугольника, то и сама точка – внутри многоугольника.

Проектирование вершин многоугольника на эту плоскость приводит к обнулению координаты z_i , а координаты проекций вершин будут (x_i, y_i). Соответственно координаты точки пересечения луча с плоскостью многоугольника будут (x^*, y^*). Если последняя точка лежит внутри n–угольника, образовавшегося на плоскости z, то и исходная точка – внутри исходного n-угольника.

Один из приемов заключается в следующем. Система координат передвигается так, чтобы точка пересечения (x^*, y^*) оказалась в начале координат. При таком преобразовании координаты вершин спроектированного n–угольника становятся равными соответственно

$$x_i^* = x_i - x^*, \qquad y_i^* = y_i - y^*.$$

После этого есть ряд способов определить, находится ли этот центр внутри многоугольника.

1-й случай

Пусть все $x_i^* > 0$ *или* $x_i^* < 0$ одновременно, тогда центр (0, 0) находится вне многоугольника, и анализ прекращается. В противном случае анализ продолжается с использованием более сложных рассуждений.

2-й случай

Рассмотрим два ребра n–угольника с вершинами

$$(x_i^*, y_i^*), \quad (x_{i+1}^*, y_{i+1}^*),$$
$$(x_j^*, y_j^*), \quad (x_{j+1}^*, y_{j+1}^*),$$

где $i < j$, причем

$$x_i^* \cdot x_{i+1}^* < 0, \quad x_j^* \cdot x_{j+1}^* < 0.$$

Это означает, что знаки (x) у координат вершин разные, т.е. вершины лежат в положительной и отрицательной полуплоскостях (вдоль оси Ox).

Если теперь выполняется условие

$$\left(y_i^* - \frac{y_{i+1}^* - y_i^*}{x_{i+1}^* - x_i^*} \cdot x_i^* \right) \cdot \left(y_j^* - \frac{y_{j+1}^* - y_j^*}{x_{j+1}^* - x_j^*} \cdot x_j^* \right) < 0,$$

то интересующая нас точка лежит внутри многоугольника. Это означает, что и исходная точка луча попадает внутрь многоугольника.

В каждом из сомножителей слева – координаты точки пересечения границ четырехугольника (т.е. прямых линий, соединяющих попарно левые и

правые точки отрезков – ребер n–угольника). Приведенное неравенство означает, что эти линии пересекают ось Ox по разные стороны от начала координат.

4.5.4 Пересечение с прямоугольным параллелепипедом

Иногда для уменьшения объема вычислений при определении пересечения луча с каким-либо объектом его помещают в объемлющий параллелепипед, и если луч не пересекает этот параллелепипед, то заведомо не будет и пересечения с исходным объектом.

Если стороны параллелепипеда параллельны координатным плоскостям, то сам он однозначно определяется любыми двумя своими вершинами, примыкающими к одной из диагоналей этого параллелепипеда. Например, если известны координаты вершин, причем

$$x_1 < x_2,\ y_1 < y_2,\ z_1 < z_2,$$

то параллелепипед строится однозначно.

Рассмотрим луч, исходящий из точки x_0, y_0, z_0 по направлению, определяемому вектором (l, m, n), причем

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

Опишем алгоритм, который определяет, пересекает ли этот луч заданный выше своими вершинами параллелепипед.

Рассмотрим, например, пару плоскостей, параллельных координатной плоскости Oyz, которые определяются уравнениями

$$x = x_1 \quad \text{и} \quad x = x_2.$$

Если

$$l = 0, \qquad (*)$$

то данный луч параллелен этим плоскостям, и при

$$x_0 < x_1 \quad или \quad x_0 > x_2$$

он не пересекает параллелепипед.

Если же условие (*) не выполняется, то можно найти отношения

$$t_{1x} = \frac{x_1 - x_0}{l}, \quad t_{2x} = \frac{x_2 - x_0}{l},$$

причем можно принять, что $t_{1x} < t_{2x}$ (иначе их можно просто поменять местами). Обозначим теперь

$$t_{near} = t_{1x},\ t_{far} = t_{2x}.$$

Совершенно также при m ≠ 0 находим величины

$$t_{1y} = \frac{y_1 - y_0}{m},\ t_{2y} = \frac{y_2 - y_0}{m},$$

и если

$$t_{1y} > t_{near},$$

то принимается

$$t_{near} = t_{1y},$$

а если

$$t_{2y} < t_{far},$$

то принимается

$$t_{far} = t_{2y}.$$

Сравниваем теперь полученные значения t, по существу отвечающие расстоянию от центра луча до пересечения с плоскостями, содержащими в себе грани параллелепипеда. Если

$$t_{near} > t_{far} \quad или \quad t_{far} < 0,$$

то пересечения нет, т.е. луч идет мимо параллелепипеда.

Анализ продолжается затем для последней пары плоскостей $z = z_1$ и $z = z_2$.

Если после проведения всех сравнений получим

$$0 < t_{near} < t_{far} \quad или \quad 0 < t_{far},$$

то луч обязательно «протыкает» параллелепипед.

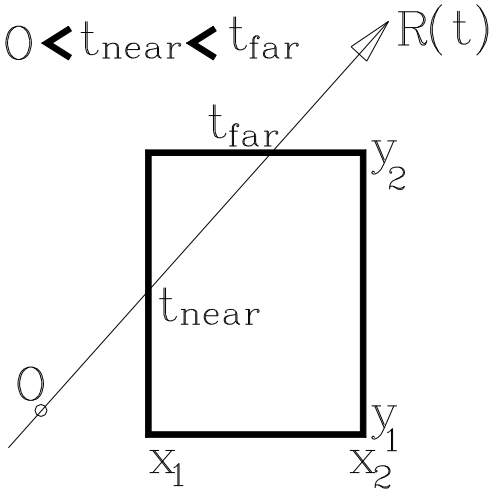


Рис. 23

На рис. 23 приведена иллюстрация к случаю, когда луч проходит через параллелепипед.

На рис. 24 показан случай, когда луч проходит мимо параллелепипеда.

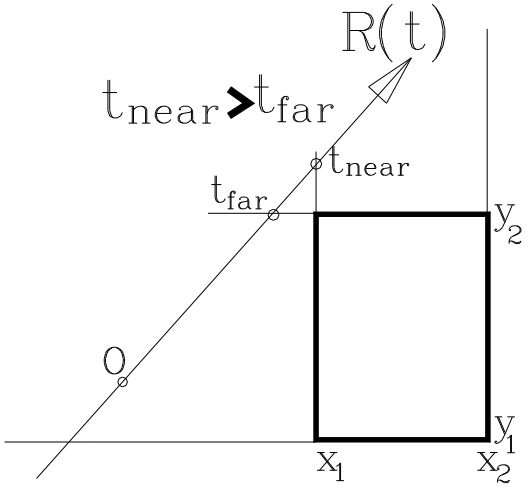


Рис. 24

V. Удаление невидимых линий и поверхностей

5.1 Постановка проблемы и терминология

При построении реалистичных изображений трехмерных объектов одна из сложнейших проблем компьютерной графики – удаление невидимых линий и частей поверхностей. К этой проблеме примыкают вопросы прозрачности, отражения, построения теней и т.д. Раньше эта проблема обсуждалась в начальной части курса при историческом обзоре существовавших технических средств, в частности, экранов мониторов со сплошным слоем люминофора и с реализованным техническим растром. Далее будут обсуждаться проблемы, связанные с программной частью решения этого вопроса.

Считаем, что наблюдатель находится в центре проектирования, а при ортогональном проектировании этот центр находится на бесконечности.

Будем говорить, что точка P_1 поверхности S загораживает точку P_2, если на картинной плоскости проекции P_1 и P_2 совпадают, а P_1 находится между P_2 и центром проектирования (наблюдателем).

Точка P называется видимой, если она не загораживается никакой другой.

Универсальных алгоритмов удаления невидимых линий нет.

Существуют **два основных подхода** к решению проблемы загораживания и удаления невидимых линий.

В первом случае для каждого пиксела определяется тот объект, который вдоль направления проектирования является ближайшим к наблюдателю (центру проектирования). Именно характеристики этого объекта и определят цвет и светимость пиксела. Этот подход существенным образом использует растровые свойства дисплея, а работа по такой схеме называется работой в пространстве картинной плоскости (картинные методы). Время работы таких алгоритмов связано как с количеством изображаемых объектов n, так и числом точек растра N (не забываем, что на разных мониторах величина пиксела, а, следовательно, и их количество, различны).

Таким образом, $t \sim n*N$, где N - число точек растра.

Во втором случае непосредственно сравниваются объекты друг с другом, и выясняется, какие части каких объектов являются видимыми. В этом случае работа ведется в исходном пространстве объектов и не привязана к растровым свойствам дисплея (объектные методы). Время работы таких алгоритмов пропорционально величине m , где m - число объектов сцены и их разрешение.

Условно можно считать, что существует еще и третий тип алгоритмов – их называют смешанными, когда при анализе используются и первый, и второй подходы.

1. Пусть сцена разбита на две части A и B. Если самые дальние точки A ближе, чем самые близкие B, то B не может загородить A. Поэтому сначала строим B, затем A, и получим верное изображение. Такого рода алгоритмы возникли еще в самом начале развития компьютерной графики, но сразу же стала очевидной их нерациональность. Если на первом плане находится объект, загораживающий все остальные, то вся работа по рисованию дальних объектов получается напрасной. Представьте, что мы рисуем полный стадион, тщательно добиваясь портретного сходства для каждого из зрителей, а потом на первом плане рисуем забор, который все загораживает. Ясно, что такой алгоритм обречен на выполнение лишней работы.

2. Пусть можно построить плоскость такую, что объект A находится с одной стороны от нее, объект B - с другой (рис. 25). Тогда, если

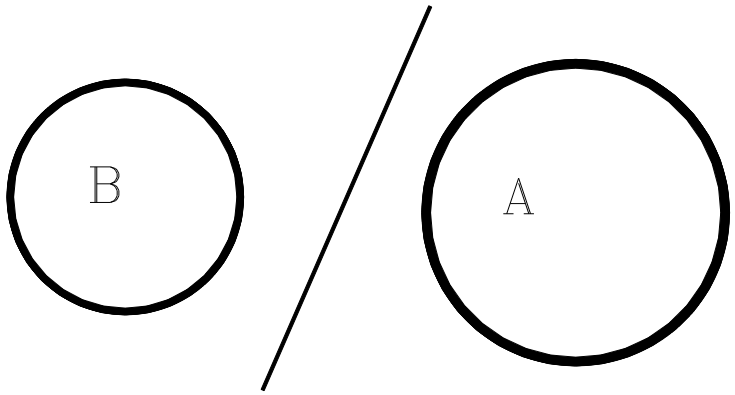


Рис. 25

наблюдатель находится со стороны объекта A, то объект B никогда не загораживает для него объект A. Проще всего это иллюстрируется примером двух непересекающихся шаров. Их разделяет плоскость, перпендикулярная отрезку, соединяющему центры шаров A и B. В зависимости от того, где находится наблюдатель, смотрится и вся сцена.

В случае выпуклых непересекающихся объектов все эти рассуждения остаются в силе. Если объекты невыпуклы, их можно попытаться поместить в непересекающиеся выпуклые тела, например, в минимального размера шары, – и тогда решение проблемы становится очевидным.

3. В предыдущих случаях по существу устанавливается последовательность обработки частей изображения, гарантирующая

отсутствие ошибок. Части изображения могут разрабатываться независимо друг от друга, т.е. каждую проекцию объектов A и B можно обрабатывать независимо.

Общая задача о пересечении проекций весьма сложна; можно использовать простые достаточные условия непересечения.

Например: описав вокруг проекций прямоугольники со сторонами, параллельными осям координат, и убедившись в их непересечении (это просто ряд неравенств), можем утверждать, что исходные фигуры тоже не пересекаются.

4. Если вся сцена сложна, ее разбивают на части A и B; затем каждую из этих частей еще на A_1, A_2, B_1, B_2 и т.д. - до тех пор, пока все не упростится до применимости простых правил. Такой подход позволяет вместо решения одной сложной задачи рассматривать ряд простых задач.

5. Вместо разбиения сцены можно разбивать изображение (картинную плоскость) на части и их обрабатывать, рассматривая только те объекты, которые попадают на эти части картинной плоскости. Принцип реализуется тот же, что и в предыдущем случае, только речь идет о разбиении картины на части, а не сцены.

6. Т.к. свойства сцены связаны со свойствами изображения, их следует учитывать при выборе метода решения задачи загораживания. Можно, например, отметить, что есть так называемые "хорошо организованные" объекты. Они отличаются таким свойством, как когерентность по видимости, например, выпуклые многогранники. Для них характер видимости меняется некоторым регулярным образом, т.е. грань либо полностью видима, либо полностью невидима.

Для невыпуклых многогранников такого свойства нет, но и в этом случае изменение видимости при переходе от одних элементов к другим носит регулярный характер: либо элемент полностью видим, либо полностью невидим; если же он частично видим, а частично нет, то его проекция на картинную плоскость обязательно пересекается с проекцией другого ребра многогранника, через которое проходит складка проектирования поверхности многогранника на картинную плоскость. Складка - последовательность ребер, каждое из которых является смежным с двумя гранями, разным образом ориентированными по отношению к наблюдателю (нормали к этим граням направлены в разные стороны относительно луча проектирования).

Это свойство многогранников и гладких аналогичных поверхностей положено в основу методов количественной невидимости в задачах загораживания.

5.2.Способы представления поверхности

В компьютерной графике, прежде чем вести речь о построении изображения некоторой поверхности, необходимо определиться, о каком представлении этой поверхности идет речь. Различаются три основных типа представления поверхности.

1. Аналитический: поверхность задается при помощи аналитического выражения, обычно для простых (их называют каноническими) случаев - сфера, цилиндр, конус и т.п.

2. Полиэдральный: поверхность как совокупность многоугольных граней; количество граней - основная характеристика полиэдральной поверхности;

3. Параметрический: в виде набора частей, каждая из которых представляет собой параметрически заданную поверхность; нередко при помощи триангуляции такие поверхности заменяются представлениями в виде многогранников.

Ребра многогранника и линии сетки параметрического представления называют каркасными линиями, а соответствующее **изображение - каркасным.**

Если поверхность изображается с помощью полутоновой закраски ее элементов, то **изображение называется полутоновым.** Построение таких изображений – самостоятельная проблема компьютерной графики.

По способу визуализации алгоритмы делятся на две группы:

1. Алгоритмы, дающие каркасное изображение.
2. Алгоритмы, создающие полутоновое изображение.

Далее эта вторая группа алгоритмов рассматриваться и анализироваться не будет.

Глубиной элемента поверхности называется расстояние от этого элемента до картинной плоскости.

В алгоритмах загораживания используются две специфические процедуры: тест глубины и тест принадлежности.

5.2 Некоторые подходы к решению задачи загораживания

Исторически первыми возникли методы переборного типа. Они предполагают прямой перебор элементов сцены. Авторы алгоритмов

стремились точно решить задачу о том, какие именно части элементов сцены должны быть удалены. Поэтому такие алгоритмы лучше работают при получении каркасных изображений.

При работе такого алгоритма рассматривается каждое ребро каждого из объектов сцены, и анализируется его взаимное расположение со всеми гранями каждого из объектов сцены.

При этом возникают следующие варианты:

A. Ребро полностью находится в том же полупространстве относительно грани, что и наблюдатель - тогда оно полностью видимо.

B. Ребро полностью находится в полупространстве, не содержащем наблюдателя. Это не означает, против ожидания, что оно невидимо, так как грань имеет конечные размеры (это не плоскость, а лишь ее относительно небольшой «кусочек»).

Выявляется, закрыто ли ребро гранью. Если проекция ребра находится вне контура проекции грани, то ребро видимо. Если внутри - невидимо. Если проекция ребра находится частично внутри проекции грани, то для видимой части нужно проверить все сначала с другой очередной гранью.

C. Ребро пересекается с плоскостью грани. Тогда точкой пересечения выделяются две части, одна из которых находится в полупространстве наблюдателя, вторая в другом полупространстве. После этого для каждой части ребра анализ проводится в последовательности, описанных в пунктах A или B.

Время реализации таких алгоритмов пропорционально произведению $m \cdot n$, где m - число ребер, n - число граней.

5.3 Удаление нелицевых граней многогранника

Пусть в пространстве задан многогранник, не обязательно выпуклый. Необходимо построить его изображение либо с использованием центральной проекции (центр проектирования P), либо ортогонального проектирования с направляющим вектором l.

Пусть F - некоторая грань многогранника.

Плоскость, несущая эту грань, разбивает все пространство на две части, при этом внешняя нормаль определяет т.н. положительное полупространство.

При центральном проектировании грань F называется лицевой, если P лежит в положительном полупространстве.

При ортогональном проектировании F лицевая, если l и n образуют острый угол.

Для выпуклого многоугольника удаление всех нелицевых граней полностью решает задачу визуализации.

Если многоугольник произвольный (невыпуклый), то удаление нелицевых граней позволяет уменьшить число подлежащих анализу граней.

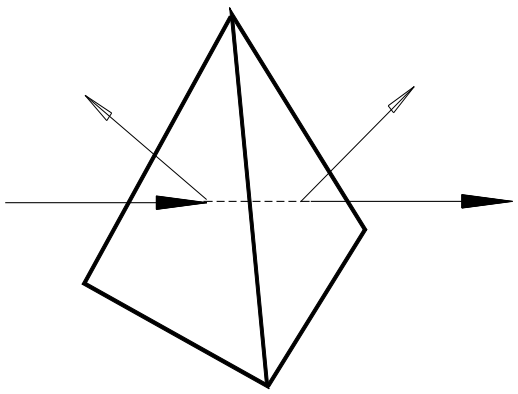


Рис. 26. К вопросу об удалении нелицевых граней многогранников

5.4 Удаление невидимых линий. Алгоритм Робертса. Алгоритм Аппеля

Даже если грань сама по себе невидима, ребро (или даже несколько ребер) этой грани все равно могут быть видимы. В том случае, когда каждая из граней является выпуклым многоугольником, применяется **алгоритм Робертса**.

Сначала выбрасываются все ребра, которые определяются пересечением невидимых граней. Такие ребра заведомо невидимы и не включаются в список подлежащих проверке на видимость.

Для оставшихся ребер проводится проверка на закрывание их всеми гранями многогранника. При этом возникают варианты:

- грань не закрывает ребро;
- грань полностью закрывает ребро, и тогда это ребро удаляется из списка анализируемых на видимость;
- грань частично закрывает ребро, тогда само ребро разбивается на несколько частей, из которых видимыми могут быть не более двух частей. Ребро исключается из списка, но те его части, которые не закрыты, включаются в список;

Время работы такого алгоритма связано с числом граней. Так, если число граней n, то время определяется как $O(n^2)$.

Можно значительно сократить время работы с помощью следующего простого приема. Картинная плоскость разбивается на равные клетки, и для каждой клетки составляется список тех граней, проекции которых в нее попадают. Тогда при проверке произвольного ребра сначала определяются

клетки, в которые оно попадает, и проверка на закрывание делается для списка граней, отвечающих этим клеткам.

Хотя этот прием требует некоторых затрат на разбиение картинной плоскости клетками, все равно при удачном выборе такого разбиения значительно сокращается и составляет величину порядка $O(n)$.

Более эффективным является **алгоритм Аппеля**.

Вводится понятие так называемой количественной невидимости точки. Это количество лицевых граней, закрывающих данную точку. Ясно, что очка видима только в том случае, если ее количественная невидимость равна нулю.

Рассмотрим, как меняется количественная невидимость вдоль ребра.

Количественная невидимость точек ребра меняется на единицу при прохождении ребра позади так называемой контурной линии. Эта линия состоит из тех ребер, для которых одна из граней является лицевой, а другая – нелицевой. Например, для многогранника на рисунке такой линией будет ломаная ABCIJDEKLGA (рис. 27).

Для определения видимости ребер произвольного многогранника берется одна из вершин, и ее видимость определяется непосредственно (определением количественной невидимости).

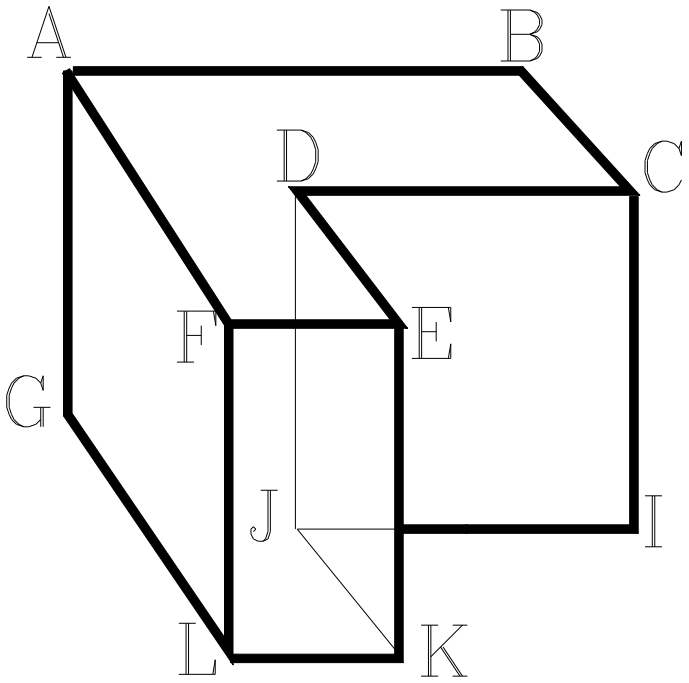


Рис. 27

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

На следующем шаге определяется количественная невидимость ребер, выходящих из новой вершины, и т. д.

В результате определяется количественная невидимость связной компоненты сцены, содержащей исходную вершину, и при этом она рисуется.

В случае, когда рассматривается изменение количественной невидимости вдоль ребра, выходящего из вершины, принадлежащей контурной линии, нужно проверить, не закрывается ли это ребро одной из граней, выходящей из этой вершины. Например, для примера на рисунке грань DEKJ закрывает ребро DJ.

Так как для любого реального объекта количество ребер, входящих в контурную линию, намного меньше общего числа ребер, то алгоритм Аппеля более эффективен, чем алгоритм Робертса. Более того, для дальнейшего повышения его эффективности можно использовать разбиение картинной плоскости на части.

5.5 Метод Z - буфера

Этот метод удобен для аппаратной реализации ввиду простоты алгоритма и используемого в нем набора операций. Временные характеристики этого метода линейно зависят от числа точек растра и "глубинной сложности" сцены, т.е. среднего числа граней, взаимно закрывающих друг друга - но не зависят от числа граней поверхности многогранника.

Этот метод привлекается для сложных сцен с применением ортогонального проектирования на картинную плоскость.

Для использования метода организуются и используются две области памяти: буфер глубины (Z - буфер) и буфер кадра (К – буфер), хранящий информацию о состоянии пикселей экрана компьютера.

Z - буфер (глубины) - для хранения координаты *z* («глубины») каждого видимого на данной стадии анализа изображения пикселя картинной плоскости;

К - буфер (кадра) - для запоминания и хранения атрибутов (яркость и цвет) соответствующего пикселя.

В начальный момент в Z - буфере хранится число, обозначающее глубину фона, обычно это возможно большое число. В К - буфере хранятся атрибуты фона.

При работе проекция очередной грани разлагается в растр. После этого анализируется глубина (значение *z*) каждого нового пикселя изображения путем сравнения ее с глубиной того пикселя, который имеет ту же проекцию на картинную плоскость и уже занесен в буфер глубины.

Если *z* нового пикселя меньше *z* старого, то рассматриваемый элемент ближе к картинной плоскости. Подправляется координата *z* соответствующего элемента Z-буфера, а атрибуты нового пикселя заносятся в К-буфер.

В противном случае не делается никаких действий.

Формальное описание метода:

1. Весь буфер кадра инициализируется фоновыми значениями (интенсивности, цвета).
2. Буфер глубины инициализируется значениями глубины фона.
3. Для каждой грани сцены:
   3.1. Преобразуются проекции грани в растровую форму.
   3.2. Для каждого пикселя проекции вычисляется его глубина $z = z(x, y)$.
   3.3. Сравнивается значение $z(x, y)$ с соответствующим значением буфера глубины $Z(x, y)$.
   3.4. Если $z(x, y) < Z(x, y)$, то
      3.4.1. Записывается атрибут этого пикселя в буфер кадра.
      3.4.2. Записывается z вместо Z.
   3.5. Иначе никаких действий.

Алгоритм метода Z-буфера легко модифицируется для получения сечений поверхности параллельными плоскостями z_1 ,z_2 .

Это можно получить, если в пункте 3.4 поставить условие

Если $z < Z(x, y)$ и $z_1 < z(x, y) < z_2$.

В итоге на картинной плоскости останутся только те части сцены, которые расположены между секущими плоскостями z_1, z_2 .

5.6 Алгоритмы упорядочения

Идея метода заключается в таком упорядочении граней, чтобы при их выводе получилось корректное изображение. Для этого, например, достаточно выводить дальние грани раньше, чем более близкие. Но существуют случаи, когда грани таким образом упорядочить не удается (рис. 28). В подобных случаях необходимо разбиение одной или

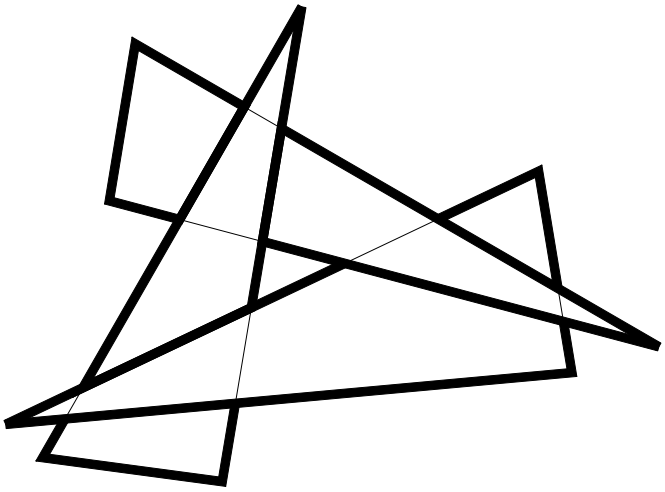


Рис. 28

нескольких граней на части, чтобы получившиеся после разбиения грани можно было упорядочить.

5.6.1 Метод сортировки по глубине

Самый простой способ сортировки заключается в том, чтобы упорядочить грани по их расстоянию до картинной плоскости вдоль направления проектирования. Рисование затем надо проводить в порядке приближения.

Это метод хорошо работает, например, при изображении достаточно простых непересекающихся тел. Однако иногда просто сортировка по расстоянию до картинной плоскости не обеспечивает правильного упорядочения граней. Поэтому желательно после такой сортировки проверить порядок вывода граней. Для определенности далее считаем, что рассматривается параллельное проектирование вдоль оси Oz. Перед выводом грани P следует убедиться, что никакая другая грань Q, проекция которой на ось Oz пересекается с проекцией грани P, не может закрываться гранью P. Лишь при выполнении этого условия грань P может быть выведена раньше.

Проверка может быть проведена в порядке возрастания сложности по следующим пяти тестам.

1. Пересекаются ли проекции этих граней на ось Ox?
2. Пересекаются ли проекции этих граней на ось Oy?
3. Находится ли грань P по другую сторону от плоскости, проходящей через грань Q, чем наблюдатель?
4. Находится ли грань Q по ту же сторону от плоскости грани P, что и наблюдатель?
5. Пересекаются ли проекции этих граней на картинную плоскость?

Если хотя бы на один из этих вопросов можно дать отрицательный ответ, то можно считать, что грани P и Q упорядочены верно. Тогда грань P сравнивается со следующей гранью. Если же все ответы положительны, то надо поменять грани P и Q местами, для чего проверяются следующие тесты:

3′. Находится ли грань Q по другую сторону плоскости P, чем начало координат ?

4′. Находится ли грань P по ту же сторону от плоскости Q, что и начало координат?

В случае, если ни один из этих тестов не дает возможности однозначно судить, какая грань должна выводиться раньше, то одна из них разбивается плоскостью, проходящей через другую грань. После этого вопрос об упорядочении оставшейся грани и частей разбитой грани решается уже просто.

5.6.2 Метод двоичного разбиения пространства

Существует другой способ упорядочения граней, который можно назвать элегантным.

Рассмотрим некоторую плоскость в объектном пространстве, которая не пересекает ни одну из граней. Эта плоскость разбивает все грани на два непересекающихся множества граней (так называемых кластера), лежащих по разные стороны плоскости. Очевидно, что ни одна из граней, лежащих в полупространстве, не содержащем наблюдателя, не может закрывать собой ни одну из граней другого множества. Поэтому нужно сначала изображать грани дальнего кластера, а уже потом ближнего.

Подобная техника может применяться и внутри каждого кластера. В кластере проводится своя плоскость, и далее такой процесс разбиения можно повторить – до тех пор, пока в каждом получившемся кластере останется не более одной грани.

Обычно в качестве разбивающей плоскости рассматривается плоскость одной из граней. При этом множество всех граней разбивается на 4 класса:

1) лежащие на плоскости;
2) пересекающие плоскость;
3) лежащие в положительном полупространстве;
4) лежащие в отрицательном полупространстве относительно этой плоскости.

Все грани, пересекаемые плоскостью, разобьем вдоль этой плоскости.

В итоге возникает дерево разбиений пространства, узлами которого являются грани. Процесс построения дерева заключается в выборе грани, проведении через нее плоскости и разбиении множества всех граней. В этом процессе существует некоторый произвол в очередности выбора граней. После построения дерева можно строить изображение в соответствии с выбранным видом и центром проектирования.

Одним из преимуществ этого метода является его полная независимость от положения центра проектирования. Поэтому метод удобен для построения серии изображений одной и той же сцены из разных точек наблюдения (проектирования).

5.7 Метод построчного сканирования

Этот метод является примером метода, работающего в пространстве картинной плоскости. Вместо задачи удаления невидимых граней для проекций объектов на картинную плоскость решается серия простых одномерных задач.

На картинной плоскости любое изображение строится как ряд горизонтальных (или вертикальных) линий пикселов. Рассмотрим сечение сцены плоскостью, проходящей через такую линию и центр проектирования (эта плоскость всегда может быть построена). Пересечением этой плоскости с объектами сцены будет множество непересекающихся отрезков (за исключением их концов), которые и необходимо спроектировать. Задача удаления невидимых частей для такого набора отрезков решается тривиально. Рассматривая задачу удаления невидимых граней для каждой такой линии пикселей, тем самым разбиваем исходную задачу на набор гораздо более простых задач.

Подобные алгоритмы используются при создании компьютерных игр.

Пусть сцена представляет собой прямоугольный лабиринт с постоянным расстоянием от пола до потолка и набором вертикальных сцен (на рис. 29 показан вид сверху).

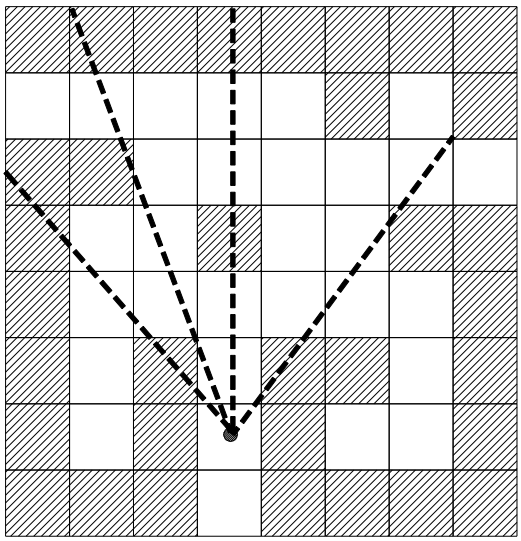


Рис. 29

Изображение сцены разложено в ряд вертикальных линий. Каждая из них определяет вертикальную полуплоскость, проходящую через линию и точку наблюдения. Ясно, что из всех пересечений этой полуплоскости со стенами лабиринта видимым будет только ближайшее. По существу, задача поиска пересечений сводится к задаче в плоскости *Oxy*, что равносильно решению задачи о поиске пересечений луча, идущего из центра наблюдения, с набором отрезков, которые представляют собой проекции стен лабиринта.

После построения таких пересечений можно найти проекции стен на эту линию, пользуясь свойствами центрального проектирования. Каждая вертикальная линия изображения состоит из трех частей – пол, часть стены, потолок. Поэтому после определения части линии, занимаемой проекцией стены (это будет отрезок), оставшиеся части линии заполняются цветом пола и потолка.

5.8 Алгоритм Варнака

Этот алгоритм основан на разбиении картинной плоскости на части, после чего для каждой из них изображение строится относительно просто.

Например, разобьем картинную плоскость на 4 равные части (рис. 30).

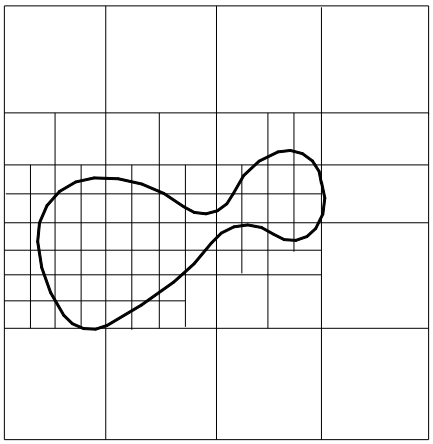


Рис. 30

Если какая-либо часть полностью накрывается проекцией ближайшей грани или вообще ничем не закрывается, вопрос о закрашивании тривиален.

Если ни одно из этих условий не выполнено для конкретной части, данная часть разбивается на четыре части, и для нее проверяется снова выполнение (или невыполнение) этих условий, и т.д. Такой процесс разбиения продолжается до тех пор, пока размер части не станет равным размеру пикселя. Разбиение прекращается, а для части размером в пиксель находится ближайшая грань явным образом, и проводится закрашивание.

5.9 Триангуляция

Триангуляцией называется регулярное разбиение плоскости на треугольники. По принципу триангуляции построены, например, опорные пункты геодезической сети, которая и носит название триангуляционной сети.

Вообще плоскость можно разбить регулярным образом только на 3-, 4-, 5- и 6-угольники. Это следует из того, что угол α при вершине правильного n – угольника вычисляется по формуле

$$\alpha = (n - 2)*\pi/n.$$

Если стыковать многоугольники в узлах по схеме «вершина к вершине», то начиная с семиугольника сумма углов в таком узле будет более 2π. Если в отдельно взятом узле и можно свести три неправильных семиугольника, то регулярно накрыть плоскость семиугольниками – правильными или неправильными, все равно – не получится именно по этой причине.

5.9.1 Приближение функции на нерегулярной сетке

Задача: пусть известны значения функции в N произвольным образом расположенных точках, и требуется найти ее приближенное значение в некоторой новой точке, не совпадающей ни с одной из заданных.

Эта задача очень часто встречается на практике. Например, сеть метеостанций расположена нерегулярным образом, а прогноз погоды нужно делать для каждого населенного пункта. Часто в технике по ограниченному количеству датчиков нужно ясно представлять распределение температуры или каких-либо других параметров по некоторому объему или на некоторой поверхности, и т.д.

Один из подходов к решению задачи заключается в следующем. По точкам и значениям функции в них строится поверхность, в простейшем варианте кусочно-линейная, состоящая из треугольников. Следует заметить, что для построения кусочно-линейной поверхности применимы только треугольники, т.к. по трем точкам всегда строится плоскость. Уже по четырем точкам в общем случае построить плоскость невозможно. На плоскости хОу необходимо по заданным точкам создать систему из непересекающихся треугольников. Проекция каждой точки пространства на плоскость принадлежит лишь одной из треугольных граней, и значения функции $f(x, y)$ аппроксимируются кусочно-линейной функцией, принимающей заданные значения в опорных точках. **Процесс триангуляции есть создание сети непересекающихся треугольников с вершинами в заданных точках**. По теореме Эйлера количество треугольников не превышает удвоенного числа исходных точек.

Триангуляция имеет ряд преимуществ перед разбиением на прямоугольники; например, это подстройка сети под заданную систему точек, что прямоугольники не позволяют сделать. У прямоугольной сети есть два направления, в общем случае не согласованные с начальными данными, и при повороте решетки нужно сильно мельчить сеть, чтобы не уйти далеко от исходной формы (границы) поверхности.

Недостатки треугольного разбиения тоже очевидны. Так, если для прямоугольной сети легко определяются координаты вершин, то для треугольников это сделать сложнее.

Для определения "качества" сети есть различные критерии. Например, иногда формулируется требование минимальности суммарной длины ребер триангуляции - это т.н. минимальная взвешенная триангуляция. Есть метод так называемой "жадной триангуляции" - когда не отменяется никакое ранее сделанное построение. Этот метод последовательно порождает ребра, и процесс завершается, когда возникает необходимое их число, определяемое множеством точек и его выпуклой оболочкой. Если при этом по прежнему выполняется требование минимизации суммарной длины ребер, то просто применяется локальный критерий. На каждом шаге добавляется самое короткое из возможных ребер, совместимое с теми, что уже построены ранее, т.е. не пересекающее ни одно из них.

В приложениях чаще всего встречается триангуляция Делоне. Она однозначно строит сеть из наиболее правильных треугольников, что удобно в расчетах. Этой триангуляции естественным образом соответствует кусочно-линейная поверхность в пространстве, состоящая из треугольников.

5.9.2 Алгоритм построения триангуляции

Итак, рассматривается следующая задача: на плоскости есть N точек, и их нужно соединить непересекающимися отрезками, чтобы каждая область внутри выпуклой оболочки этого множества была треугольником.

В результате мы должны получить список ребер, образующих триангуляцию.

Рассмотрим **алгоритм триангуляции Делоне**. Начинается процесс с разбиения плоскости на так называемые **области Вороного.**

Пусть S - множество из N точек. Будем считать, что никакие 4 точки не лежат на одной окружности.

Областью Вороного V для точки $p \in S$ называется множество таких точек плоскости, расстояние от которых до p меньше, чем для любой другой точки из S.

Полученные таким образом N областей образуют разбиение плоскости. Объединенная граница этих областей образует сеть - диаграмму Вороного.

Если в S только две точки, то перпендикуляр к средине соединяющего их отрезка и дает деление плоскости на две области Вороного. В общем случае область Вороного точки p - это пересечение всех таких линий.

Поэтому каждая область V - выпуклая ломаная, либо замкнутая, либо нет.

Ребрами получившейся решетки являются отрезки срединных перпендикуляров. Пересечение срединных перпендикуляров к отрезкам AB, BC

есть центр описанной окружности; но тогда и AC тоже делится перпендикуляром посредине.

Оказывается, что каждая вершина диаграммы Вороного есть пересечение трех ребер диаграммы.

Теперь рассмотрим граф, двойственный диаграмме Вороного, т.е. граф, полученный в результате соединения отрезками каждой пары точек множества S, многоугольники Вороного которых имеют общие ребра.

В итоге получаем разбиение области треугольниками - это и есть триангуляция Делоне области S.

Свойства этой триангуляции:

1. От любой локальной перестройки триангуляции Делоне треугольники становятся более неправильными. Это значит, что при переходе от разбиения (рис. 31) ABC, ABD к ACD, BCD (это называется

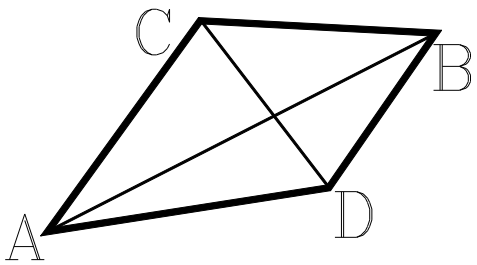


Рис. 31

"флип") минимальный из шести углов в паре треугольников не увеличится, т.е. он может только уменьшиться.

2. Внутри окружности, описанной внутри любого треугольника, нет больше ни одной вершины триангуляции Делоне.

Процесс построения триангуляции ведется по индукции. Пусть для определенного этапа триангуляция построена. На следующем шаге в нее включается новая точка, после чего производится локальная перестройка триангуляции.

Для новой точки:

1. Ищется треугольник, в который она попала, если она лежит внутри выпуклой оболочки триангуляции, построенной на предыдущем шаге; либо

2. Ищутся вершины, с которыми нужно соединить новую точку, если она вне оболочки.

В первом случае новая вершина соединяется с вершинами треугольника, в который она попала; если нужно, делается флип.

Во втором случае нужно новую точку соединить с последним ребром, которое пересек отрезок AB. Чтобы сохранить свойство выпуклости триангуляции, нужно еще соединить новую точку B со всеми вершинами, лежащими между опорными прямыми.

5.10 Построение линий уровня функции двух переменных

Использование линий уровня широко распространено в топографии, в физике (изобары, изотермы и т.д.), в метеорологии, в механике и т.п.

Часто на практике поле величин не является сплошным - есть дискретные значения в отдельных его точках. Если эта сетка редкая, необходимо использовать интерполяцию. Этот процесс более или менее прост для прямоугольных сеток.

В принципе от прямоугольной или четырехугольной сетки всегда можно перейти к треугольникам - достаточно разделить четырехугольники пополам.

При помощи интерполяции можно заменить график функции кусочно-линейной поверхностью; при этом вершины вдоль оси z проектируются в узлы сетки.

Таким образом, исходная задача сводится к задаче построений линий уровня получившейся кусочно-линейной функции, определенной на объединении треугольников. Грани этой поверхности - это треугольники с вершинами в точках P (x, y, f) - где (x, y) - узлы прямоугольной сетки, f - значения $f(x, y)$ в соответствующих узлах.

Линией уровня этой функции, соответствующей значению уровня h, является объединение проекции на плоскость $z = 0$ отрезков, получающихся в результате пересечения плоскости $z = h$ с треугольными гранями поверхности многогранника.

Таким образом, для решения нужно иметь процедуру нахождения пересечения треугольной грани с горизонтальной плоскостью. Применив эту процедуру последовательно к каждой грани, получим кусочно-линейную аппроксимацию линий уровня функции.

Возможны варианты:

1. Треугольник и плоскость не пересекаются, т.е. все вершины треугольника находятся по одну сторону от плоскости.
2. Треугольник касается плоскости одной вершиной, а две другие находятся по одну сторону от плоскости.
3. Треугольник касается плоскости ребром, т.е. две вершины находятся в плоскости, одна - вне ее.
4. Треугольник полностью в плоскости - все его вершины.
5. Треугольник пересекается с плоскостью - т.е. есть пара вершин, лежащих по разные стороны от плоскости.

Когда значения функции заданы в точках нерегулярного множества, можно поступить следующим образом. Одним из способов получить триангуляцию с вершинами в точках этого множества. После разбиения области определения функции на треугольники построить кусочно-линейную

интерполяцию функции с вершинами в точках исходного множества. Далее, находя пересечения плоскости с треугольниками, получим линии уровня функций - для разных значений *h* при необходимости.

5.11 Метод плавающего горизонта

Более точно этот метод иногда называется методом всплывающего горизонта, и причина этой модификации названия становится ясной, если разобраться в сути метода. Она сводится к следующему.

Пусть необходимо построить поверхность $z = f(x, y)$ с изображением только видимой ее части. Пусть при этом область изменения переменной y – от 0 до 1 (можно показать, что если пределы изменения переменной y будут другими, принципиально ничего не изменится). В декартовой системе координат Oxyz зафиксируем плоскость $y = 0$, она пройдет прямо по осям Ox и Oz. В этой плоскости теперь очень просто можно построить кривую $z = f(x, 0)$ (рис.32).

Далее рассматриваем плоскость 1 (см. рисунок), отстоящую от исходной на некоторое расстояние dy, и в этой плоскости строим кривую $z = f(x, dy)$, причем сплошными линиями изображаем только ту ее часть, что видна из-за первой линии. После этого на плоскости 2, отстоящей на расстояние $2 \cdot dy$ от исходной (или на dy от плоскости 1), строим кривую $z = f(x, 2 \cdot dy)$. Показываем опять только ту ее часть, что не закрывается двумя первыми линиями. Процесс продолжается до тех пор, пока не будет построена последняя линия $z = f(x, 1)$. Каждый раз новая линия строится как видимая лишь в той части, что не скрыта предыдущими линиями. Огибающая этих линий и представляют собой тот самый «всплывающий горизонт», о котором идет

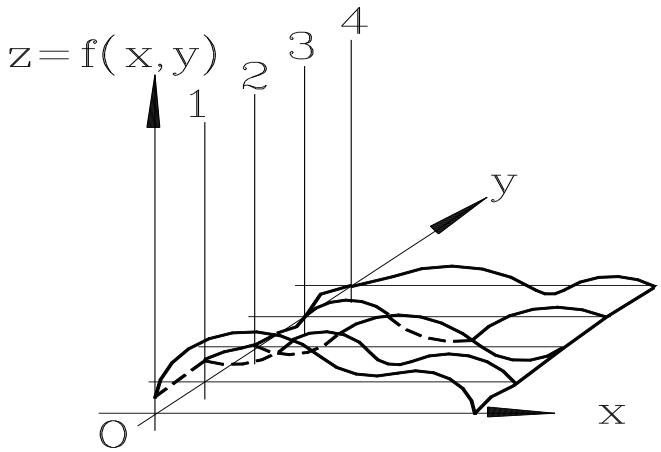


Рис. 32

речь в названии метода. Изображение поверхности получается как совокупность линий, построенных последовательно в указанном порядке.

ЗАКЛЮЧЕНИЕ

В настоящем пособии отражены наиболее важные с точки зрения автора сведения, которые необходимо знать любому квалифицированному пользователю. Эти сведения в основном изложены без привязки их к какому-либо конкретному варианту Автокада или другой графической системы. В тексте оговаривалось, что все графические системы в своей основе построены на одних и тех же принципах, что соответствует наличию общепринятых стандартов создания таких систем. Пользователь, имеющий дело с какой-либо конкретной версией известного или нового графического пакета, может убедиться, что их построение подчиняется одним и тем же требованиям.

В то же время круг вопросов, которые традиционно относятся к компьютерной графике, много шире того, что нашло отражение в настоящем пособии. Так, это вопросы трехмерной графики, связанные с учетом наличия источников света при построении реалистических изображений, с учетом рельефа или текстуры поверхностей тел, с построением геометрических форм различного рода моделей объектов. Это вопросы математических методов синтеза изображений. Это вопросы, которые относятся к прикладным – архитектурное проектирование, распознавание видеообразов, автоматизация проектирования в машиностроении и других технических областях. Это задачи восстановления формы объекта по нескольким его проекциям и сечениям, что важно, например, в медицинских исследованиях. Это видеотренажеры и имитаторы сложных сцен в играх и обучающих системах. Это использование компьютерной графики в рекламе, игровых и мультипликационных фильмах. Это перечень можно продолжить, но из сказанного уже ясно, что многие эти и другие области применения и соответствующие проблемы компьютерной графики в настоящем пособии даже не упомянуты.

И все же можно уверенно сказать, что информация, нашедшая отражение в настоящем пособии, безусловно, полезна читателю, впервые сталкивающемуся с предметом компьютерной графики. Во всяком случае, дело обстоит так, если пользователю захочется пользоваться соответствующими прикладными пакетами и графическими системами не вслепую, а с пониманием сущности происходящих в компьютере преобразований информации, которая и служит основой построения видеообразов.

В настоящее время компьютерная графика представляет собой бурно развивающуюся часть компьютерных технологий. Это развитие происходит в соответствии с принципами построения графических систем, которые в свое время были разработаны авторами соответствующего стандарта – ядра графической системы. Хотя время вносит коррективы в любые стандарты, основные принципы их оказались достаточно удачными и сохраняют свое значение и сейчас.

Процесс развития компьютерной графики, как и любой компьютерной технологии, содержит две составляющие – техническую, или аппаратную, и программную. Развитие их идет параллельно, заставляя каждую из составляющих поддерживать темп, чтобы не отставать от другой. Прогресс в этой области за последние 10 лет настолько разителен, что возможности, воспринимаемые в начале 90-х годов как выдающиеся, сейчас вызывают у современных слушателей курса ироническое отношение, явно незаслуженное.

Главной целью курса является не столько обучение пользователя приемам работы с конкретными графическими пакетами, сколько пробуждение интереса к этой области науки и техники, а также получение представления об основных направлениях ее развития в настоящее время и в ближайшей перспективе.

Автор будет считать, что свою задачу он хотя бы отчасти выполнил, если среди читателей учебного пособия найдутся такие, у которых возникнет устойчивый профессиональный интерес к этому увлекательному занятию – компьютерной графике.

ЛИТЕРАТУРА

1. Грайс Д. Графические средства персонального компьютера. Пер. с англ. – М.: Мир, 1989. – 376 с.
2. Шикин Е.В., Боресков А.В., Зайцев А.А. Начала компьютерной графики. – М.: ДИАЛОГ-МИФИ, 1993. – 138 с.
3. Шикин Е.В., Боресков А.В. Компьютерная графика. Динамика, реалистические изображения. – М.: ДИАЛОГ-МИФИ, 1995. – 288 с.
4. Иванов В.П., Батраков А.С. Трехмерная компьютерная графика. – М.: Радио и связь, 1995. – 224 с.
5. AutoCAD. Полезные рецепты / Под ред. М.И. Кнеллера. – М.: Радио и связь, 1994. – 208 с.