Алексей Поляков

bhv

Методы и алгоритмы КОМПЬЮТЕРНОЙ ГРАФИКИ в примерах на Visual C++

- Математические основы компьютерной графики
- Векторная и растровая графика
- Цифровая обработка изображений
- Готовые решения

```
auxWireTorus()
glPixelTransferf(GL_ALPHA_SCALE, 1);
CPolygon::Show()
CMatrixFilter::TransformPix()
V = xi + yj + zk
CFilter
```

+ CD-ROM

МАСТЕР

Алексей Поляков

Методы и алгоритмы КОМПЬЮТЕРНОЙ ГРАФИКИ *в примерах на Visual C++*

Санкт-Петербург

«БХВ-Петербург»

2002

УДК 681.3.06 + 800.92Си
ББК 32.973.26-018
П54А

Поляков А. Ю.

П54А Методы и алгоритмы компьютерной графики в примерах на Visual C++. — СПб.: БХВ-Петербург, 2002. — 416 с.: ил.

ISBN 5-94157-136-4

В книге представлен курс компьютерной графики. Основное внимание уделяется вопросам прикладного программирования с использованием языка Visual C++ и библиотеки MFC. Рассмотрены математические основы компьютерной графики: преобразования на плоскости и в трехмерном пространстве, методы построения кривых и поверхностей, которые могут быть использованы для визуализации результатов научных расчетов. Значительный раздел книги посвящен работе с растровой графикой. Описывается формат BMP, функции сохранения и загрузки растровых изображений. Обсуждается создание и применение графических фильтров для обработки изображений. Отдельно рассматривается назначение библиотек OpenGL и DirectX, приводится пример использования OpenGL для визуализации трехмерной сцены. Весь изложенный материал иллюстрируется программами, написанными с использованием объектно-ориентированного стиля.

Для программистов

УДК 681.3.06 + 800.92Си
ББК 32.973.26-018

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Анатолий Адаменко*
Зав. редакцией	*Анна Кузьмина*
Редактор	*Леонид Кочин*
Компьютерная верстка	*Натальи Караваевой*
Корректор	*Татьяна Звертановская*
Дизайн обложки	*Игоря Цырульникова*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 23.05.02.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 33,54.
Тираж 3000 экз. Заказ №
"БХВ-Петербург", 198005, Санкт-Петербург, Измайловский пр., 29.

Гигиеническое заключение на продукцию, товар № 77.99.02.953.Д.001537.03.02
от 13.03.2002 г. выдано Департаментом ГСЭН Минздрава России.

Отпечатано с готовых диапозитивов
в Академической типографии "Наука" РАН
199034, Санкт-Петербург, 9 линия, 12.

ISBN 5-94157-136-4

СОДЕРЖАНИЕ

Моему брату

Предисловие

Эта книга представляет собой курс основ компьютерной графики. Здесь изложены алгоритмы и методы решения многих задач, возникающих при работе с векторными и растровыми изображениями. Основное внимание уделено вопросам прикладного программирования с использованием языка Microsoft Visual C++ и MFC. Материал книги накоплен за несколько лет чтения курса лекций "Компьютерная графика" в Томском государственном университете систем управления и радиоэлектроники (ТУСУР), а также в результате практической работы над несколькими научными и коммерческими проектами.

Основная цель книги — показать, как можно запрограммировать ту или иную задачу компьютерной графики: построить сплайновую кривую, нарисовать поверхность, отобразить на этой поверхности линии уровня, управлять несколькими объектами-фигурами в программе, сохранить изображение, считать растровое изображение с диска, обработать и записать обратно на диск, распечатать изображение на принтере.

Другая задача — продемонстрировать различные пути и способы достижения одинаковых результатов. Поэтому при изучении материала книги старайтесь всегда находить новые, более удачные варианты решения рассмотренных задач.

Весь теоретический материал книги иллюстрируется рабочими примерами программ. В качестве языка программирования выбран C++, поскольку он представляется наиболее подходящим для решения серьезных задач компьютерной графики. В примерах применяется объектно-ориентированный стиль программирования (ООП). Используя стиль ООП, легко представить программу в виде отдельных частей (модулей, деталей), взаимодействующих между собой, если же вы склонны к философствованию, то можно мыслить о программе, как о сообществе неких индивидуумов, которые могут рождаться, обретать какие-то свойства, общаться между собой и умирать.

Каждая программа, описанная в книге, представляет собой законченное приложение. Законченное в том смысле, что его можно откомпилировать, запустить, посмотреть, что оно делает, сохранить результаты работы. Для работы с примерами вам потребуется компьютер с операционной системой MS Windows 95 (или более поздней версией) и среда разработки MS Visual C++ 6.0.

Поскольку данная книга является учебником по компьютерной графике, а не справочным пособием по Windows API или GDI, то в ней, как правило, не приводится подробного описания параметров вызова API-функций или полного описания методов классов MFC. Эти сведения могут быть легко получены из электронной библиотеки Microsoft Developer Network (MSDN) Library или специальной литературы.

На кого рассчитана эта книга

Книга предназначена студентам и преподавателям, специализирующимся в области информатики и вычислительной техники. Предполагается, что читатель уже имеет определенный опыт программирования. Однако изложение в книге ведется "от простого к сложному", основные этапы создания программ описаны достаточно подробно. Для понимания теоретического материала требуются начальные знания из области линейной алгебры и тригонометрии. Необходимым условием для успешного усвоения практической части книги является знание языка программирования C++ и знакомство с основными идеями объектно-ориентированного программирования.

Структура книги

Книга состоит из 12 глав. Тематически ее можно разделить на четыре части. Первая часть — изучение основных понятий компьютерной графики и средств программирования (главы 1—3). Вторая часть — векторная графика (главы 4—8). Третья часть — растровая графика (главы 9—11). Четвертая часть — назначение графических библиотек (глава 12).

Глава 1 посвящена рассмотрению задач, решаемых с использованием компьютерной графики. В ней также приводятся основные понятия и определения.

В *главе 2* рассматриваются основные средства, которые будут использованы далее при программировании всех примеров книги. Среди них:

- особенности программирования "под Windows";
- основные понятия и принципы объектно-ориентированного программирования;
- назначение библиотеки MFC;
- процесс создания приложения в MS Visual C++.

В *главе 3* рассмотрен от начала и до конца пример создания "графического" приложения Painter, в котором пользователь сможет нарисовать свой первый рисунок, сохранить его в виде файла и вывести на печать.

В *главе 4* более детально рассматриваются используемые программные средства, развивается программа Painter, проектируется иерархия классов графических объектов.

Глава 5 посвящена рассмотрению математических основ преобразований на плоскости.

В *главе 6* материал пятой главы находит свое практическое применение, а также рассматривается реализация в программе Painter некоторых функций редактирования создаваемых изображений.

Глава 7 посвящена теории и практике преобразований в трехмерном пространстве. В программе Painter реализуется построение трехмерной поверхности $z = f(x, y)$ и линий уровня на поверхности.

В *главе 8* рассматриваются математические основы и программная реализация построения сплайновых кривых.

Глава 9 посвящена использованию растровых ресурсов (пиктограмм, курсоров, растровых картинок) в приложениях.

В *главе 10* рассматривается растровый формат BMP, в программе Painter реализуется экспорт изображений в BMP-файл.

Глава 11 целиком посвящена работе с растровыми изображениями. В ней рассматривается создание и применение графических фильтров для обработки изображений. Разрабатывается программа, позволяющая загрузить, обработать и сохранить растровые изображения. Эту программу вы сможете использовать на практике для улучшения качества своего электронного фотоархива, а, расширив ее возможности, добиться неповторимых эффектов.

В *главе 12* рассматривается назначение библиотек OpenGL и DirectX, приводится пример использования OpenGL для визуализации трехмерной сцены и вывода на экран растрового изображения.

Обратная связь

Легко предположить, что у вас могут возникнуть вопросы, замечания, пожелания, критика и предложения. Пожалуйста, направляйте их в "книгу жалоб и предложений" по адресу **graphics@elecard.net.ru**. Я, конечно, не беру на себя обязательство откликаться на все письма, но если мне найдется, что сказать, то непременно постараюсь ответить. Ответы на часто задаваемые вопросы будут публиковаться по адресу **http://www.elecard.com/graphics**.

Благодарности

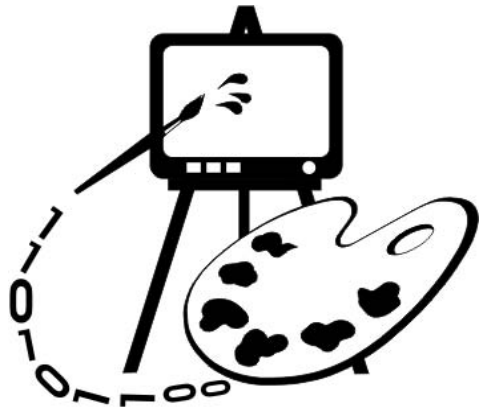
Я благодарен преподавателям и студентам кафедры компьютерных систем управления и проектирования ТУСУР за участие в обсуждении многих рассмотренных в книге тем. Хочу выразить отдельную признательность моему научному руководителю кандидату технических наук Леониду Ивановичу Бабаку. Спасибо студентам Елене Завадской и Сергею Цибенко за участие в деле построения поверхностей и линий уровня.

Я очень признателен руководству компании Элекард **http://www.elecard.com** за предоставленное в мое распоряжение необходимое оборудование, а своим коллегам по фирме Moonlight Russia за ценное общение.

Выражаю глубокую благодарность за поддержку своим родителям Анне Афанасьевне и Юрию Антониновичу.

Особое спасибо жене Марине за терпение, заботу и вдохновение.

Алексей Поляков

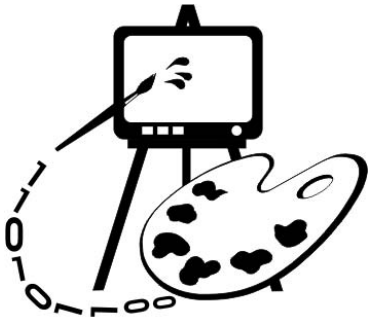


Часть I

ОСНОВЫ КОМПЬЮТЕРНОЙ ГРАФИКИ

Глава 1

Основные понятия машинной графики



В данной главе рассматриваются:

- цели и задачи машинной графики;
- основные понятия и определения.

1.1. Цели и задачи машинной графики

Понятие "компьютерная графика" объединяет довольно широкий круг операций по обработке графической информации с помощью компьютера. Причем наблюдается явная тенденция "компьютеризации" изображений циркулирующих в обществе. Стали обыденностью термины "цифровое фото" и "видео". Западные кинорежиссеры давно уже пытаются испугать нас ужасами будущего, захваченного компьютерными монстрами, подсовывающими людям виртуальный суррогат вместо прекрасной реальности. В виртуальных буднях грядущего компьютерной графике отводится огромная роль. Это связано с тем, что, по мнению ученых, исследующих проблемы мозга, зрительная система в иерархии мозговых структур человека занимает особое место. С восприятием и обработкой визуальной информации непосредственно связано примерно 20% мозга человека. Благодаря зрению мы получаем по разным оценкам от 70 до 90% сведений об окружающем мире. Следовательно, образный мир компьютерной графики является одним из глубинных проявлений человеческой природы.

В компьютерной графике можно выделить несколько основных направлений.

- Визуализация научных (расчетных или экспериментальных) данных. Большинство современных математических программных пакетов (например, Maple, MatLab, MathCAD) имеют средства для отображения графиков, поверхностей и трехмерных тел, построенных на основе каких-либо расчетов. Кроме того, графическая информация может активно

использоваться в самом процессе вычислений. Например, в системе Image, разработанной на кафедре компьютерных систем управления и проектирования Томского университета систем управления и радиоэлектроники, визуальные образы, выводимые на экран, являются основой для решения математических и проектных задач. Визуализация позволяет представить большой объем данных в удобной для анализа форме и широко используется при обработке результатов различных измерений и вычислений.

- Геометрическое проектирование и моделирование. Это направление компьютерной графики связано с решением задач начертательной геометрии — построением чертежей, эскизов, объемных изображений с помощью программных систем, получивших название CAD-системы (от английского Computer-Aided Design), например, AutoCAD.

  Существует большое количество специализированных CAD-систем в машиностроении, архитектуре и т. д.

- Распознавание образов. Способность человека распознавать абстрактные образы считают одним из важнейших факторов, определившим развитие его мыслительных способностей и выделившим его из животного мира[1]. Решение задачи распознавания и классификации графической информации является одной из ключевых и при создании искусственного интеллекта. Уже в наши дни компьютеры распознают образы повсеместно: системы идентификации футбольных хулиганов у входа на стадион; анализ аэро- и космических фотоснимков; системы сортировки, наведения и т. д. Возможно, самый известный пример распознавания образов — сканирование и перевод "фотографии" текста в набор отдельных символов, формирующих слова. Программное обеспечение многих современных сканеров позволяет выполнить такую операцию. Кроме того, существуют специализированные программы распознавания текста, например, FineReader.

- Изобразительное искусство. К этому направлению можно отнести разнообразную графическую рекламу от текстовых транспарантов и фирменных знаков до компьютерных видеофильмов, обработку фотографий, создание рисунков, мультипликацию и т. д. В качестве примера популярных

[1] "Символические изображения, будь то животные, нарисованные на стенах пещер, или лунные календари, вырезанные на кости животных, ассоциируются исключительно с нашим видом. Возможно, в результате произошедшего небольшого щелчка в устройстве нашего мозга вкупе с некоторым "переключением проводов" наши предки обрели тот тип мышления, который был недоступен неандертальцам, — а именно, способность к распознаванию абстрактных символов. И как только этот скрытый интеллектуальный потенциал начинал использоваться вследствие изобретения способов символьной коммуникации — произошедшего, скажем, 35 000 лет назад, его обладатели должны были быстро выйти на совершенно новый уровень технологий". — Майкл Л. Ротсчайльд. Биономика.

программ из этой области компьютерной графики можно назвать Adobe Photoshop (обработка растровых изображений), CorelDRAW (создание векторной графики), 3ds max (трехмерное моделирование).

- Виртуальная реальность. Реальность, даже виртуальная, подразумевает воздействия на всю совокупность органов чувств человека, но в первую очередь на его зрение. К компьютерной графике можно отнести задачи моделирования внешнего мира в различных приложениях: от компьютерных игр до тренажеров. Кроме того, не стоит забывать о компьютерах-злодеях, которые используют виртуальную реальность для захвата мира. Поэтому надо изучать компьютерную графику, чтобы не дать себя провести.
- Цифровое видео. Все более широкое распространение получают анимированные изображения, записанные в цифровом формате. Это прежде всего фильмы, передаваемые через компьютерные сети, а также видеодиски Digital Video Disk (DVD), цифровое кабельное и спутниковое телевидение.

Приведенная классификация сфер применения компьютерной графики является во многом условной. Возможно, найдутся задачи, которые нельзя отнести ни к одному из обозначенных направлений.

1.2. Основные понятия и определения

1.2.1. Графический формат

Графическим форматом называют порядок (структуру), согласно которому данные, описывающие изображение, записаны в файле.

Графические данные обычно разделяются на два класса: *векторные* и *растровые*. Изображения, в зависимости от типа описывающих их данных, называются векторными или растровыми.

Векторные данные используются для представления прямых, многоугольников, кривых и т. д. с помощью определенных в числовом виде *базовых* (опорных, контрольных, ключевых) точек. Программа, обрабатывающая векторные данные, воспроизводит линии посредством соединения базовых точек. Вместе с информацией о базовых точках хранятся атрибуты (цвет, толщина и другие параметры линий) и набор правил (соглашений) вывода (рисования). Пример векторного изображения приведен на рис. 1.1.

Растровые данные представляют собой набор числовых значений, определяющих яркость и цвет отдельных *пикселов*. Пикселами (или пикселями — от английского pixel) называются минимальные элементы (цветные точки), из которых формируется растровое изображение. Термин "растр" в компьютерной графике и полиграфии имеет несколько отличающихся значений.

Далее под растром будем понимать массив пикселов (массив числовых значений). Для обозначения массива пикселов часто используется термин *bitmap* (битовая карта). В *bitmap* каждому пикселу отводится определенное число битов (одинаковое для всех пикселов изображения). Это число называется *битовой глубиной* пиксела или *цветовой глубиной* изображения, т. к. от количества бит, отводимых на один пиксел, зависит количество цветов изображения. Наиболее часто используется пикселная глубина 1, 2, 4, 8, 15, 16, 24 и 32 бита.

Рис. 1.1. Векторный рисунок

Источниками растровых данных могут быть программы, формирующие изображение на растровом экране и различного рода устройства для ввода изображений (сканеры, цифровые камеры и др.).

Пример растрового изображения приведен на рис. 1.2.

Рис. 1.2. Растровый рисунок

Типы форматов графических файлов определяются способом хранения и типом графических данных. Наиболее широко используются растровый, векторный и метафайловый форматы.

Векторный формат наиболее удобен для хранения изображений, которые можно разложить на простые геометрические фигуры (например, чертежи или текст). Векторные файлы содержат математические описания элементов изображения. Наиболее распространенные векторные форматы: AutoCAD DXF и Microsoft SYLK.

Растровый формат используется для хранения растровых данных. Файлы такого типа особенно хорошо подходят для хранения реальных изображений, например, фотографий. Растровые файлы содержат пикселную карту изображения и ее спецификацию. Наиболее распространенные растровые форматы: BMP, TIFF, GIF, PCX, JPEG.

Метафайловый формат позволяет хранить в одном файле и векторные, и растровые данные. Примером такого формата являются файлы CorelDRAW — CDR.

Кроме того, существуют файловые форматы для хранения мультипликации (видеоинформации), мультимедиа-форматы (одновременно хранят звуковую, видео и графическую информацию), гипертекстовые (позволяют хранить не только текст, но и связи-переходы внутри него) и гипермедиа (гипертекст плюс графическая и видеоинформация) форматы, форматы трехмерных сцен, форматы шрифтов и т. д.

1.2.2. Элементы графического файла

Графические файлы состоят из последовательности данных или структур данных, называемых *файловыми элементами* или *элементами данных*. Эти элементы подразделяются на три категории: поля, теги и потоки.

- *Поле* — это структура данных в графическом файле, имеющая фиксированный размер. Для определения положения поля в файле обычно задают либо абсолютное смещение от начала или конца файла, либо смещение относительно другого поля.
- *Тег* — это структура данных, размер и позиция которой изменяются от файла к файлу. Позиция тега может задаваться абсолютно, либо относительно от другого файлового элемента. Теги могут содержать в себе другие теги или наборы связанных полей.
- *Поток* — набор данных, предназначенный для последовательного чтения. В отличие от полей и тегов поток не обеспечивает быстрого доступа к нужным данным, т. к. их положение в файле определяется в процессе чтения.

Как правило, в графических файлах применяются комбинации этих элементов данных.

1.2.3. Преобразование форматов

Часто возникает проблема преобразования формата данных, связанная с необходимостью обмена изображениями между различными программами. Для преобразования данных существуют специализированные программы, кроме того, распространенные графические редакторы позволяют читать и сохранять изображения в различных форматах. Поэтому преобразование растровых данных в одном формате в растровые данные в другом формате обычно не представляет сложности. Другое дело — преобразование векторных изображений в растровые. При таком преобразовании неизбежно теряется часть информации, т. к. происходит переход от "идеального" математического описания рисунка к его дискретному (растровому) представлению. Обратное же преобразование из растрового формата в векторный вообще является нетривиальной задачей, связанной с распознаванием образов.

1.2.4. Сжатие данных

Сжатие — процесс уменьшения физического размера блока данных. Так как изображения, как правило, описываются большим количеством данных, файлы изображений имеют большой размер. Поэтому графические данные часто подвергаются сжатию. Обычно каждый формат графического файла поддерживает какой-либо из методов (алгоритмов) сжатия. В большинстве случаев сжатие заключается в замене избыточной информации на ее более компактную форму. Сжатие бывает физическое и логическое. Различие между физическим и логическим сжатием заключается в методе получения более компактной формы данных. Физическое сжатие данных выполняется без учета содержащейся в них информации. Логическое же сжатие — напротив основано на логическом анализе информации. Примером логического сжатия может служить преобразование строки "Союз Советских Социалистических Республик" в аббревиатуру "СССР". Для графических данных логическое сжатие не применяется.

Методы сжатия бывают *с потерями* и *без потерь*. Когда данные сжимаются, а после восстанавливаются (распаковываются) и полученные данные полностью соответствуют исходной информации, то говорят, что имело место сжатие без потерь. Иначе говоря, при методе сжатия без потерь не должно происходить какого-либо изменения данных.

Методы сжатия с потерями предусматривают отбрасывание некоторой части данных изображения для достижения большей степени сжатия.

Некоторые наиболее распространенные методы сжатия.

- Упаковка пикселов. Метод заключается в компактной записи пикселов с глубиной 1, 2 и 4 бита компактно в 8-битовые байты соответственно по 8, 4 и 2 штуки.

- Групповое кодирование (Run-Length Encoding, RLE) — является общим алгоритмом кодирования и применяется в таких растровых форматах, как BMP, TIFF, PCX.
- Алгоритм Lempel-Ziv-Welch (LZW) применяется в форматах GIF, TIFF.
- Алгоритм JPEG, разработанный объединенной экспертной группой по фотографии, включает в себя целый набор методов сжатия. Базовая реализация JPEG применяет схему преобразования изображения по алгоритму дискретных косинус-преобразований с последующим кодированием методом Хаффмана.
- Фрактальное сжатие — математический процесс, используемый для кодирования растровых изображений в совокупность математических данных, которые описывают фрактальные (похожие, повторяющиеся) свойства изображения.

Сжатие в основном применяется к данным растровых изображений. В растровых файлах сжимаются только данные изображения, другие же данные (заголовок файла, таблица цветов и т. п.) остаются всегда несжатыми.

Векторные файлы сжимаются редко. Это связано с тем, что векторные форматы сами по себе хранят данные в очень компактной форме и сжатие не даст ощутимого эффекта.

Важно уяснить, что алгоритмы сжатия не задают какой-либо файловый формат, а определяют только способ кодирования данных.

1.2.5. Пикселы и цвет

Различают физические и логические пикселы.

Физические пикселы — реальные точки, отображаемые на устройстве вывода — наименьшие элементы на поверхности отображения, которыми можно манипулировать. При выводе на экран или принтер один физический пиксел обычно формируется из нескольких более мелких цветовых точек. Например, один цветной пиксел на мониторе формируется из трех более мелких точек красного, зеленого и синего цветов, яркость которых и определяет цвет пиксела.

Логические пикселы подобны чертям на кончике иглы[1] — они имеют местоположение и цвет, но не занимают физического пространства (то есть нельзя вычислить высоту и ширину такого пиксела). По сути логический пиксел — это всего лишь некоторое число, которое задает его цвет. Положение же логического пиксела (его координаты) определяется его местом в карте изображения и количеством пикселов на единицу измерения (разрешением).

[1] "Сколько чертей уместится на кончике иглы" — известный схоластический спор.

При отображении логических пикселов на физическом экране происходит преобразование численных данных, характеризующих яркость и цвет логических пикселов, в интенсивность свечения физических пикселов. При этом никак не обойтись без учета размера и расположения физических пикселов.

Количество цветов пиксела напрямую связано с отводимым для него количеством битов и равно 2^n, где n — количество битов. Изображения, каждому пикселу которого отводится один бит, называют монохромными — двухцветными. Такие изображения вполне подходят для чертежей и текста. Изображения с глубиной цвета 24 бита и более называют *truecolor* (истинные цвета). Каждый пиксел такого изображения может принимать более 16 миллионов цветов. Считается, что этого вполне достаточно, чтобы правильно отобразить окружающую нас действительность.

При отображении цветов, заданных для логических пикселов на устройстве визуализации, может возникнуть проблема согласования цветов. Например, если устройство вывода способно отобразить до 16 млн. цветов, а изображение имеет глубину цвета 8 бит (256 цветов), то проблем с его отображением не будет. Если же все наоборот, то программе визуализации придется потрудиться, выполняя преобразование цветов изображения к возможностям устройства вывода. В последнем случае неизбежна потеря части данных, и, как следствие, снижение качества изображения. Кроме того, возможно возникновение всякого рода побочных эффектов (муар, вторичные контуры — артефакты одним словом).

1.2.6. Палитры цветов

Палитра цветов, называемая также *картой* или *таблицей цветов*, представляет собой одномерный массив цветовых величин. С помощью палитры цвета задаются косвенно, посредством указания их позиции в массиве. При использовании палитры цветов, сведения о цветах пикселов записаны в файле в виде последовательности индексов. Использование палитр во многих случаях позволяет значительно сократить объем растровых данных.

Наиболее часто используются палитры из 16 и 256 цветов, соответственно глубина цвета 4 и 8 битов, но могут быть палитры и других размеров.

Например, каждый пиксел изображения с глубиной цвета в 4 бита может иметь 16 цветов (2^4). Эти 16 цветов определены в палитре, которая обычно включается в один файл с растровыми данными. Каждое пикселное значение рассматривается как индекс в этой палитре и содержит одно из значений от 0 до 15. Значения цветов в палитре задаются с максимально возможной точностью. Обычно элемент палитры занимает 24 бита (3 байта). Таким образом, элемент палитры может задавать один из 16 777 216 цветов (2^{24}). Программа, осуществляющая визуализацию изображения, читает из файла растровые данные — индексы в таблице цветов и использует соответствующие им цвета для окрашивания пикселов экрана (рис. 1.3). Палитры разных изображений чаще всего различаются.

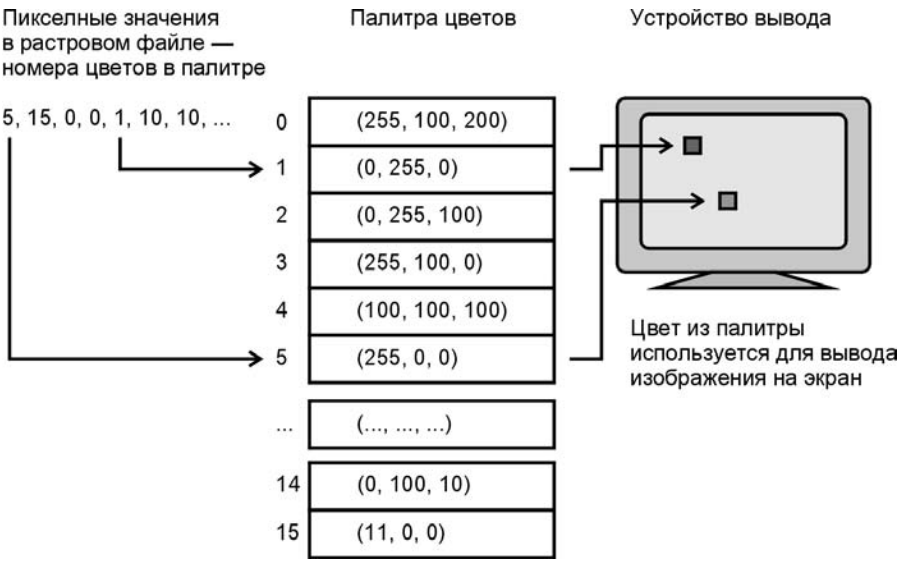


Рис. 1.3. Определение цвета с помощью палитры

1.2.7. Цвет. Цветовые модели

Для описания цветов применяют несколько различных математических систем (моделей). Ни одна из существующих систем представления цвета не является наилучшей. Для разных целей и задач служат различные системы.

В графических файлах обычно используют цветовые модели, основанные на трех основных цветах, которые не могут быть получены смешиванием других цветов. Все множество цветов, которые могут быть получены путем смешивания основных цветов, представляет собой *цветовое пространство*, или *цветовую гамму*.

Цветовые модели могут быть разделены на две категории: *аддитивные* и *субтрактивные*. В аддитивных моделях новые цвета получаются путем сложения основных цветов различной интенсивности с черным цветом. Чем больше интенсивность добавляемых цветов, тем ближе результирующий цвет к белому. Белый цвет получается при максимальных значениях интенсивности всех трех основных цветов, черный — при минимальных. Аддитивные модели формирования цвета применяются в самосветящихся устройствах (например, мониторах).

В субтрактивных моделях основные цвета вычитаются из белого. Чем больше интенсивность вычитаемых цветов, тем ближе результат к черному. Субтрактивные модели применяются при формировании цветных изображений на отражающих носителях, например, бумаге.

Наиболее распространенные цветовые модели.

- **Модель RGB** (Red-Green-Blue — красный-зеленый-синий) — модель наиболее широко используемая в графических форматах. RGB — аддитивная модель. Каждый пиксел представляется в виде трех числовых величин — трех интенсивностей красного, зеленого и синего цветов. Каждому цвету обычно отводится 8 битов, в которых может быть записано 256 уровней интенсивности. Таким образом, значение (0, 0, 0) представляет черный цвет, а (255, 255, 255) — белый.
- **Модель CMY** (Cyan-Magenta-Yellow — голубой-пурпурный-желтый) — субтрактивная модель, применяется для получения цветных изображений на белой поверхности. При освещении изображения, полученного с помощью модели CMY, каждый из основных цветов поглощает дополняющий его цвет: голубой поглощает красный; пурпурный — зеленый; желтый — синий. Теоретически наивысшая интенсивность всех трех основных цветов субтрактивной модели должна обеспечить черный цвет. На практике этого не происходит (так как реальные красители далеки от математических идеалов), поэтому в модель вводят четвертый компонент — черный цвет (Black), обозначаемый буквой "K". В результате получается модель CMYK, широко распространенная в полиграфии. При использовании субтрактивной модели изображение каждого пиксела (цветной точки) изображения состоит из четырех пятен основных цветов.

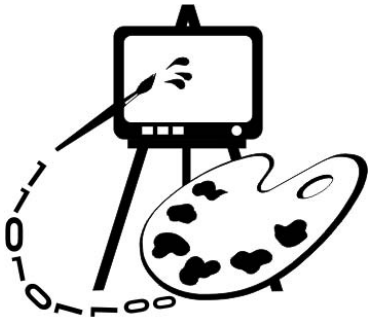
Существует и другие модели, не основанные на смешении цветов, например, HSV (Hue-Saturation-Value — оттенок-насыщенность-величина). Оттенок — это, по сути, цветовой тон, например, красный, оранжевый, синий и т. д. Насыщенность определяет количество белого в оттенке. Например, если в полностью насыщенном (100%) оттенке красного не содержится белого, такой оттенок считается чистым. Насыщенность 50% задает более светлый цвет, и в нашем примере, будет соответствовать розовому цвету. Величина (яркость) задает интенсивность свечения цвета.

1.3. Заключение

Итак, в этой главе мы коротко рассмотрели основные задачи, при решении которых используются средства компьютерной графики, и дали определения используемым терминам. В следующих главах на практике рассмотрим работу с векторной и растровой графикой. В качестве дополнительной литературы можно порекомендовать [6, 15].

Глава 2

Особенности программирования "под Windows"



В данной главе рассматриваются:

- особенности программирования "под Windows";
- основные понятия и принципы объектно-ориентированного программирования;
- создание приложения в MS Visual C++.

Сегодня, когда операционная система Win32 (имеются в виду все 32-разрядные разновидности Windows) получила широчайшее распространение, и о работе под DOS и Win16 уже стали забывать, программирование под Windows стало обыденностью. Однако не лишним будет подчеркнуть те концепции Windows, благодаря которым она стала столь популярной. Win32 предоставляет программисту такие возможности, о которых лет 10 назад можно было только мечтать. Пожалуй, основными отличиями операционных систем Win32 от их предшественниц является вытесняющая многозадачность и возможность прямой адресации до 4 Гбайт.

Вытесняющая многозадачность означает, что операционная система сама решает, какой из программ предоставить в распоряжение процессор. Каждая программа, отработав некоторое время, автоматически выгружается системой, и управление передается другой задаче. Возможно, кому-то не понравится такой "авторитарный" стиль управления. Зато такая организация позволяет, во-первых, создать иллюзию одновременного выполнения нескольких программ, а во-вторых, не допускает возможности полного захвата ресурсов компьютера какой-нибудь сбойной задачей и таким образом защищает систему от "зависания" (но мы-то знаем, что на практике это не всегда так). На мой взгляд, о вытесняющей многозадачности лучше всех сказал Омар Хайям:

Напрасно ты винишь в непостоянстве рок,
Что не в накладе ты, тебе и невдомек.
Когда б он в милостях своих был постоянен,
Ты б очереди ждать своей до смерти мог.

Вторая особенность Win32 — 32-разрядная адресация в полностью виртуальном адресном пространстве, т. е. любое приложение[1], независимо от того, сколько их одновременно выполняется, может распоряжаться 4 Гбайтами памяти. Причем, т. к. каждому приложению выделяется свое адресное пространство, они не могут случайно повредить данные друг друга. Получив в свое распоряжение столь большой ресурс памяти, программисты смогли наконец-то вздохнуть свободно. Интересно, далеко ли то время, когда 4 Гбайт не будет хватать для решения обычных задач.

Кроме перечисленных особенностей Win32, можно отметить также следующие:

- поддержка многозадачности на уровне процессов и потоков;
- возможность синхронизации потоков;
- существование файлов, проецируемых в память;
- наличие механизмов обмена данными между процессами.

В Win32 процессом называется каждая выполняющаяся программа. Каждый процесс имеет как минимум один поток выполнения. Термин "поток" (thread) можно определить как логическое направление выполнения программы. Если программу-процесс сравнить с рекой, то потоки можно сравнить с ее рукавами. Река может разделиться на несколько потоков-рукавов, которые будут течь параллельно друг другу. Например, когда программа должна выполнить какую-нибудь продолжительную операцию, ее целесообразно выделить в отдельный поток, основной же поток программы в это время может продолжать общаться с пользователем. Все потоки, принадлежащие одному процессу, выполняются в одном адресном пространстве и имеют общие с этим процессом код, ресурсы и глобальные переменные.

Для того чтобы несколько потоков слаженно решали поставленные задачи и не мешали друг другу при использовании каких-то общих ресурсов, применяется синхронизация потоков. Синхронизация может потребоваться, например, в случае, когда один из потоков должен дождаться завершения какой-то операции, выполняемой другим потоком, или, когда потоки работают с ресурсом, способным одновременно обслуживать лишь один из них. Поскольку потоки выполняются в условиях вытесняющей многозадачности, функции их синхронизации берет на себя операционная система. Для управления потоками в Windows используются специальные флаги, на которых основано действие нескольких механизмов синхронизации: *семафоры* (semaphores), *исключающие* (mutex) семафоры, *события* (event), *критические секции* (critical section).

Каждый процесс в Win32 имеет возможность прямой адресации до 4 Гбайт. Однако 4 Гбайт оперативной памяти пока не очень типичны для большинства персональных компьютеров. Поэтому программы работают с *виртуальными*

[1] Так называют свои программы настоящие Win32-программисты.

адресами, которые отличаются от физических адресов памяти. Для того чтобы предоставить в распоряжение программ такой большой объем памяти, Windows использует специальный механизм подкачки памяти с жесткого диска. Этот механизм называется страничной организацией памяти (paging). При такой организации логическое адресное пространство каждого процесса разбито на отдельные блоки, называемые *страницами*. Страницы могут располагаться как в оперативной памяти, так и на жестком диске. Операционная система оптимизирует выделение оперативной памяти таким образом, чтобы процессы могли получить быстрый доступ к часто используемым данным. Диспетчер виртуальной памяти отслеживает давно не используемые страницы памяти и отправляет их в страничный файл на диск.

Видимо, похожий механизм управления памятью используется при создании *файлов, проецируемых в память*, называемыми еще *отображаемыми файлами*. Проецируемый файл как бы отображает файл на диске в диапазон адресов в памяти. Это позволяет выполнять операции с файлом прямо в памяти. Операционная система же сама организует обмен данных между памятью и диском. Использование файлов, проецируемых в память, позволяет значительно упростить работу с файлами, а также дает возможность разделять данные между процессами. Разделение данных происходит следующим образом: два или более процессов создают в своей виртуальной памяти проекции одной и той же физической области памяти — объекта "проецируемый файл". Когда один процесс записывает данные в свою "проекцию" файла, изменения немедленно отражаются и в "проекциях", созданных в других процессах. Для организации такого взаимодействия все процессы должны использовать одинаковое имя для объекта "проецируемый файл".

В Win32 каждый процесс имеет свое адресное пространство, поэтому одновременно выполняемые приложения не могут случайно повредить данные друг друга. Однако часто возникают задачи обмена информацией между процессами. Для этой цели в Win32 предусмотрены специальные способы, позволяющие приложениям получать совместный доступ к каким-либо данным. Все способы основаны на механизме проецирования файлов, описанном выше.

Один из способов заключается в создании "Почтового ящика" (mailslot) — специальной структуры в памяти, имитирующей обычный файл на жестком диске. Приложения могут помещать данные в "ящик" и считывать их из "ящика". Когда все приложения, работающие с ящиком, завершаются, "почтовый" ящик и данные, находящиеся в нем, удаляются из памяти.

Другой способ заключается в организации коммуникационной магистрали — "канала" (pipe), соединяющего процессы. Приложение, имеющее доступ к каналу с одного его "конца", может связаться с приложением, имеющим доступ к каналу с другого его "конца" и передать ему данные. Канал может предоставлять приложениям односторонний или двусторонний доступ. При одностороннем доступе одно приложение может записывать данные

в канал, а другое считывать; при двустороннем — оба приложения могут выполнять операции чтения/записи. Существует возможность организовать канал, коммутирующий сразу несколько процессов.

Далее при реализации алгоритмов компьютерной графики мы постараемся использовать эти возможности, предоставляемые Windows.

2.1. Типы данных в Windows

В Windows-программах вместо стандартных для C и C++ типов данных (таких как `int` или `char*`) применяются типы данных, определенные в библиотечных файлах (например, в WINDOWS.H). Часто используются следующие типы:

- HANDLE — 32-разрядное целое в качестве дескриптора (числового идентификатора какого-либо ресурса);
- HWND — дескриптор окна (32- разрядное длинное целое);
- BYTE — 8-разрядное беззнаковое символьное значение;
- WORD — 16-разрядное беззнаковое короткое целое;
- DWORD, UINT — 32-разрядное беззнаковое длинное целое;
- LONG — 32-разрядное длинное целое со знаком;
- BOOL — целое, используется для обозначения истинности (1 — TRUE) или ложности (0 — FALSE);
- LPSTR — 32-разрядный указатель на строку символов.

Кроме перечисленных, существует еще много других типов, обозначающих дескрипторы различных ресурсов, указатели, структуры и т. д. Различия многих из них заключаются лишь в том, что они обозначают. В Windows-программах можно также использовать и все традиционные типы данных.

2.2. Структура Windows-приложения

Работа Windows-программ основана на обработке сообщений, которые поступают от пользователя, операционной системы и других программ. Структура приложений обязательно включает в себя главную функцию `WinMain`, одинаково устроенную для всех приложений. С этой процедуры начинается выполнение всех Windows-программ. `WinMain` должна выполнить следующие основные действия:

1. Определить и зарегистрировать класс окна в Windows.
2. Создать и отобразить окно, определяемое данным классом.
3. Запустить цикл обработки сообщений.

При определении класса окна указывается специальная *функция окна*, которая должна реагировать на поступающие окну сообщения. Каждое приложение имеет свою очередь сообщений — ее создает Windows и помещает туда все сообщения, адресованные окну приложения. После создания и отображения окна запускается основной цикл обработки сообщений, в котором сообщения проходят предварительную обработку, и возвращаются обратно Widows. Затем Windows вызывает функцию окна программы с очередным сообщением в качестве аргумента. Анализируя сообщение, функция окна инициирует соответствующие операции. Сообщения, обработка которых не предусмотрена функцией окна, передаются обратно Windows, которая выполняет обработку "по умолчанию". Ниже приведена минимальная программа для Windows (листинг 2.1).

Листинг 2.1. Минимальная программа для Windows

```
// Минимальное приложение для Win32
#include <windows.h>
// Прототип функции окна
LRESULT CALLBACK WinFunction (
HWND hwnd,        // Указатель на окно
UINT message,     // Идентификатор сообщения
WPARAM wparam,    // Параметры
LPARAM lparam);   // сообщения

// Имя класса окна
char szWindowName[]="MyClass";

// Функция WinMain
int WINAPI WinMain(
   HINSTANCE hThisInst,  // Дескриптор экземпляра приложения
   HINSTANCE hPrevInst,  // В Win32 всегда NULL
   LPSTR lpszArgs,       // Указатель на строку с аргументами
   int nWinWode)         // Способ визуализации окна при запуске
{
   // Будет содержать указатель главного окна программы
   HWND hwnd;
   // Структура-буфер для хранения сообщений
   MSG msg;
   // Структура для определения класса окна
   WNDCLASSEX wcl;
   wcl.hInstance=hThisInst;           // Дескриптор приложения
```