万水计算机编程技术与应用系列

GDI+

程序设计实例

周鸣扬　曾洁玫　等编著

中国水利水电出版社
www.waterpub.com.cn

万水计算机编程技术与应用系列

GDI+程序设计实例

周鸣扬　曾洁玫　等编著

中国水利水电出版社

内 容 提 要

本书详细介绍了与GDI+编程相关的大部分技术细节：GDI+编程规则、GDI+编程基础、GDI+高级应用等。

全书共分为11章，第1章和第2章介绍了GDI+程序的开发基础及GDI+程序的快速入门。第3章～第5章介绍了GDI+中的画笔、画刷、文本、字体、路径、区域的基本使用。第6章和第7章介绍了GDI+的坐标变换与色彩变换，以及矩阵的基本编程和使用。第8章和第9章介绍了GDI+中对图像文件的基本使用及对图像的色彩调整处理。第10章介绍了在GDI+中处理图像文件的编码与解码、图像文件格式的转换以及对图形文件属性的修改。第11章介绍了GDI+在图像特技处理中的高级编程应用。

本书适合于能够熟练使用C#语言进行程序开发的中高级程序设计人员。不论对GDI或GDI+编程熟悉与否，这本书都会从各个角度让读者全面掌握GDI+编程的每一个技术细节。同时，对于使用其他语言（如C++、Visual Basic、Delphi）进行程序开发的读者朋友，本书也可以使其对GDI+的基本原理及高级应用有一个全面的认识。

本书的源代码可以从中国水利水电出版社网站（www.waterpub.com.cn）上下载。

图书在版编目（CIP）数据

GDI+程序设计实例 / 周鸣扬等编著. —北京：中国水利水电出版社，2004
（万水计算机编程技术与应用系列）
ISBN 7-5084-2076-4

Ⅰ. G… Ⅱ. 周… Ⅲ. C语言－程序设计 Ⅳ. TP312

中国版本图书馆CIP数据核字（2004）第031956号

书　　名	GDI+程序设计实例
作　　者	周鸣扬　曾洁玫　等编著
出版　发行	中国水利水电出版社（北京市三里河路6号　100044） 网址：www.waterpub.com.cn E-mail：mchannel@263.net（万水） sales@waterpub.com.cn 电话：（010）63202266（总机）、68331835（营销中心）、82562819（万水）
经　　售	全国各地新华书店和相关出版物销售网点
排　　版	北京万水电子信息有限公司
印　　刷	北京蓝空印刷厂
规　　格	787mm×1092mm　16开本　27.5印张　620千字
版　　次	2004年5月第1版　2004年5月第1次印刷
印　　数	0001—5000册
定　　价	38.00元

凡购买我社图书，如有缺页、倒页、脱页的，本社营销中心负责调换

前　言

Windows XP 操作系统的推出，为广大计算机用户带来了全新的用户界面，这其中最主要的原因是采用了全新的图形界面引擎 GDI+。GDI+是一种新型的图形设备接口，主要特点是它能够创建全新的用户桌面体系、能够轻易地完成二维或三维图形的处理，为桌面带来一种新型的数字化图片。同时 GDI+也提供了增强的图形处理技术，如常见的 Alpha Blending、纹理、贴图、增强的文本及图片显示技术。GDI+是一个应用编程接口，通过一组 C#类来提供接口的功能。

Windows XP 或 Windows Server 2003 中的 GDI 是为了与现有的应用程序兼容，但是开发新应用程序的程序员应当使用 GDI+以满足自己所有的图形需求，因为 GDI+将 GDI 的很多功能进行了优化，而且还提供了增强的功能。实际上，GDI+主要的特色就在于强调通过图形加速卡等硬件加速手段来达到良好的视觉感受！

基于 GDI+技术的方便、快捷、高效特性，国内已有不少程序开发人员在程序设计中开始使用 GDI+技术来取代传统的 GDI 编程。本书详细、全面地介绍了 Windows GDI+提供的各种功能，包括位图、画笔、画刷、颜色、坐标空间和坐标变换、填充形状、线条和曲线、文本与字体、图元文件、绘图与绘画、路径、矩形、区域、图像的编码与解码、图像特技处理等。本书的内容几乎涵盖了所有 GDI+的技术细节，旨在帮助读者用较短的时间快速熟练地掌握 GDI+编程技术。

本书中的所有例子程序在 Windows XP Professional、Visual Studio.NET、Office XP 环境下全部编译通过，程序开发语言使用的是 Visual C#。本书适合于希望了解如何使用 GDI+编写 Windows Forms 程序的中高级 C#程序员和其他熟悉面向对象语言的开发人员阅读。

本书是集体合作的结果，周鸣扬、曾洁玫和陈培负责全书的策划和大部分内容的编写，田丽镧同志负责本书的资料整理和收集。其他对本书成稿做出贡献的人员还有：龚波、田军、张莉、张溟、冯军、李志、张巧莉、龚志翔、李红玲、白红利、于自跃、牛献忠、于冰、田飞、王强、王均、陈磊、蔡丽、吴秋丽、邓欣、韩存兵等。

由于时间仓促及编者经验和水平有限，书中难免有不妥之处，恳肯广大读者批评指正。

编者

2003 年 11 月

目　　录

第 1 章　C#程序设计基础

本书中关于 GDI+编程的介绍，使用的编程语言是 C#。所以，在开始对 GDI+编程讲述之前，有必要先对编程语言进行简单的介绍。如果读者朋友已经很熟悉 C#，可以直接跳过本章。

本章的目的不是让读者去全面地掌握 C#语言的每一个技术细节及使用技巧，而只是对 C#语言的一些基础知识进行讲述。本章的内容对于仅熟悉 C++程序开发的人员很重要，因为，尽管 C#和 C++相类似，但是在细节上，两者却有着一些质的区别，本章的重点也是引领 C++程序开发人员迅速进入到 GDI+程序设计当中，简而言之，本章是让程序员不至于因为对 C#语法的理解障碍而影响到对 GDI+程序开发的掌握。

本章主要内容：

- C#语言基本特点。
- C#语言的基本数据类型。
- C#语言的流程控制。

1.1　C#语言概述

1.1.1　C#引出

在最近的一段时间里，C 和 C++一直是最有生命力的程序设计语言。这两种语言为程序员提供了丰富的功能、高度的灵活性和强大的底层控制能力，而这一切都不得不在效率上作出不同程度的牺牲，更为令人头痛的是，它们并不总是与当前的 Web 应用结合得很好。

理想的解决方案是，将快速的应用开发与对底层平台所有功能的访问紧密结合在一起。程序员们需要一种环境：它与 Web 标准完全同步，具备与现存应用间方便地进行集成的能力。除此之外，程序员们喜欢在需要时能够使用底层代码。

针对该问题，微软的解决方案是一种称为C#的程序语言。C#是一种现代的、面向对象的程序开发语言，它使得程序员能够在新的微软.NET 平台上快速开发种类丰富的应用程序。.NET 平台提供了大量的工具和服务，能够最大限度地发掘和使用计算及通信能力。

由于 C#是从 C 和 C++中派生出来的，因此具有 C++的功能。同时，由于是 Microsoft 公司的产品，它又同 Visual Basic 一样简单。对于 Web 开发而言，C#很像 Java，同时具有 Delphi 的一些优点。

Microsoft 宣称：C#是开发.NET 框架应用程序的最好语言。C# 是.NET 的关键性语言，它是整个.NET 平台的基础。与 C#相比，.NET 所支持的其他语言显然是配角。比如，Visual

Basic .NET 的存在主要是对千万个 Visual Basic 开发人员负责。对于 JScript.NET 和 Managed C++也同样可以这么说，后者只是增加了调用.NET 类的 C++语言。C#是惟一没有在设计思路中加入前辈语言的某种遗传的新事物。

C#还让你调用无管理的代码，也就是在 CLR 引擎控制之外的代码。这种不安全的模式允许操作原始指针来读和写内置碎片收集控制以外的内存。

由于 C#一流的面向对象的设计，从构建组件形式的高层商业对象到构造系统级应用程序，你都会发现，C#将是最合适的选择。C#语言设计的组件能够用于 Web 服务，这样，通过 Internet 可以被运行于任何操作系统上，被任何编程语言所调用。

下面进一步讨论 C#语言的基本特点。

1.1.2 C#和 Java

自 C#诞生之日起，关于 C#与 Java 之间的论战便此起彼伏。从技术上讲，C#与 Java 都是对传统面向对象程序设计在组件化软件时代的革新之果，可谓殊途同归。虽说两个语言有着“90%的重叠”，但那另外“10%的较量”也往往能够左右天平的方向。

（1）C#和 Java 都提出了对传统 C++艰深、晦涩的语法语义进行的改进。

①在语法方面，两者都摈弃了 C++中函数及其参数的 const 修饰、宏代换、全局变量和全局函数等许多华而不实的地方。在继承方面，两者都采用了更易于理解和构建的单根继承和多接口实现的方案。

②在源代码组织方面，都提出了声明与实现于一体的更好的逻辑封装。

③在类型系统方面，两个语言都在中间语言 IL 或字节代码的基础上提出了映射（Reflection）这样的概念，彻底革新了传统 C++运行时类型鉴别的问题。但在大刀阔斧地对 C++进行改革的同时，C#显得更为保守，它对很多原来 C++中很好的性质予以了保留，如基于栈分配的轻量级的结构类型、枚举类型、引用（ref）、输出（out）、数组（params）修饰的参数传递方式等，这些在 Java 中都被很可惜地丢掉了。在基本类型和单根继承的对象之间的类型统一方面，C#提出的 box/unbox 要比 Java 的包装类显得高明，效率也要好。

（2）对 C++不安全的指针及内存的分配方式，C#和 Java 都提出了托管执行环境。效率问题是托管执行环境一直以来令人诟病的地方，Java 虚拟机（JVM）的解释执行方式曾经让很多开发者觉得“慢得不可忍受”。C#的 JIT 编译方式为 C#在这块战场上赢得赞声一片，某些 C#托管代码甚至比传统 C++代码都快。虽然现在各厂商实现的 Java 平台也都一致地采取了 JIT 编译方式，但 C#在这方面的优势非常明显——C#的目标编译语言 IL 从设计初始就把效率摆在了重要的地位，而 Java 的字节代码的设计却有些鲁莽。数组的索引越界检查、类型安全在 C#和 Java 中都被提到了相当的高度。在异常处理方面，不管从内置支持，还是从执行效率来讲，C#都较 Java 略胜一筹。

（3）“一次编程，多处执行”是程序设计一直以来的一个诉求，尤其是在现代互联网络时代。在跨平台方面，Java 的支持和实现都是为人称道的，虽然 JVM 的速度仍然让人备感头疼。而 C#虽然在底层构造方面对移植性进行了充分的考虑，但至少目前还没有成熟的经过检验的产品。C#在跨平台方面似乎更热衷于 XML Web Services 互操作，而不是跨

平台编程。但 C#通过其基础语言构造（CLI）对二十多种主流语言的对象级的互操作支持又极大地提升了 C#的技术地位，和 COM 组件廉价的互操作也为 C#挣到不少分数——保持一个兼容的体系对现代软件工业非常重要，也是对广大开发人员负责任的表现。

（4）面向组件无疑是当代软件开发的主流。C#对组件编程甚至到了“迷恋”的地步，这与 6 年前就出道的 Java 不可同日而语。C#通过属性、索引器、委派、事件、操作符重载、特征、版本等实现了其对组件编程的直接支持。虽然这些在 Java 中都可以通过方法、接口或适配器来间接实现，但软件发展历史告诉我们，这样做无论对编程效率还是逻辑设计都是一种极大的损伤——高级语言首先面对的是人，而不是机器。除去这些语言层面的组件支持机制，.NET 平台还为组件的配置、运行、管理等提供了一揽子解决方案，而为组件开发量身定做的 Visual Studio.NET 更是令人兴奋，这都为 C#的组件编程开辟了广阔的天地。在其他技术方面 Java 的微弱劣势尚且可以忽略不计，但在组件编程方面 Java 相较于 C#却有着不可治愈的硬伤。尤其对于从 C++和 Visual Basic 背景过来的开发人员，C#在这方面有着不可抵挡的魅力和诱惑。

（5）在 XML Web Services 的操作方面，.NET 平台直接在 IL 中间语言中的内置 XML 支持使得 C#与生俱来地成为下一代 Web 服务的首选，这是通过 API 集来支持 Web 服务的 Java 所不能比的。在 C#中，XML、SOAP、UDDI、WSDL 等底层协议被构建成了面向开发人员的组件，而 Java 中这些仍然是 JAX（Java XML API）等底层协议的操作函数。当然这种局面可能仅仅是时间问题，一个强大的高效的 Web Services 组件模型对 Java 来说并不是不可逾越的鸿沟。

（6）在语言标准化方面，微软也史无前例地做出了令人赞赏的举动。目前 C#及.NET 平台基础构造已递交欧洲计算机制造商协会 ECMA，经过标准化后的 C#将可以由任何厂商在任何平台上实现其开发工具及其支持软件，这为 C#的发展提供了强大的驱动力。而 Java 在这方面虽有动作——JCP（Java Community Process），但无疑只能是准标准化。在组件化软件时代拥有一门像 C#一样的标准化语言，对软件界尤其是广大开发人员非常重要。

对于任何一种语言来讲，后端平台（C# for .NET、Java for J2EE）、其编程框架的支持、语言相关工具的实现以及现有的系统基础等都对程序设计语言的发展产生相当的影响。从纯技术角度来讲，C#无疑较 Java 更具竞争力。

1.2　C#语言的特点

C#是专门为.NET 应用而开发出的语言。这从根本上保证了 C#与.NET 框架的完美结合，在.NET 运行库的支持下，.NET 框架的各种优点在 C#中表现得淋漓尽致。让我们先来看看 C#的一些突出的特点吧：

- 语法简洁。
- 面向对象。
- 与 Web 紧密结合。
- 完整的安全性与错误处理。

- 版本控制。
- 灵活性与兼容性。

1.2.1 语法简洁

在默认情况下，C#代码在.NET 框架提供的可控环境下运行，不允许直接对内存操作，最大特色是取消了指针。相对于 C++，C#在类操作符方面也作了重大改进。对于我们来说，现在需要理解的仅仅是名字嵌套而已。

C#用使用关键字换掉了会给活动模板库（Active Template Library，ALT）和 COM 的使用带来麻烦的伪关键字，如 OLE_COLOR、BOOL、VARIANT_BOOL、DISPID_XXXXX 等。每种 C#类型在.NET 类库中都有了新名字。

语法冗余是 C++中的常见问题，比如 const 和#define、各种各样的字符类型等，C#对进行了简化，只保留了常见的形式，取消了其他冗余语法形式。

1.2.2 面向对象

也许你会说，从 Smalltalk 开始，面向对象就始终是任何一种现代程序设计语言的关注焦点。的确，C#具有面向对象语言所应有的一切特性：封装、继承与多态，这并不奇怪。然而通过精心的面向对象设计，可以说 C#是开发通用组件的绝佳选择。

在 C#的类型系统中，每种类型都可以看作一个对象。C#提供了名为“装箱”（boxing）与“拆箱”（unboxing）的机制来完成这种操作，而不给用户带来麻烦。我们将在以后的章节中详细介绍。

C#只允许单继承，即一个类不会有多个基类，从而避免了类型定义的混乱。在后面的学习中，你很快会发现：C#没有全局函数、全局变量和全局常量。一切的一切都必须封装在类中。代码将具有更好的可读性，并且减少了发生命名冲突的可能。

整个 C#类模型是建立在.NET 虚拟对象系统（Visual Object System，VOS）基础上的，其对象模型是.NET 基础架构的一部分。在下面将会谈到这样做的另一个好处就是兼容性。

借助于从 Visual Basic 中得来的丰富的 RAD 经验，C#具备了良好的开发环境。结合自身强大的面向对象功能，C#使得开发人员的生产效率得到极大的提高。对于公司而言，软件开发周期的缩短将能使他们更好地应付网络经济的竞争。在功能与效率的杠杆上，人们终于找到了支点。

1.2.3 与 Web 紧密结合

.NET 中，新的应用程序开发模型意味着越来越多的解决方案需要与 Web 标准相统一。例如，超文本标记语言（Hypertext Markup Language，HTML）和 XML。由于历史的原因，现存的一些开发工具不能与 Web 紧密地结合。SOAP 使得 C#克服了这一缺陷，大规模、深层次的分布式开发从此成为可能。

由于有了 Web 服务框架的帮助，对程序员来说，网络服务看起来就像是 C#的本地对象。程序员们能够利用已有的面向对象知识与技巧开发 Web 服务。仅需要使用简单的 C#

语言结构，C#组件就能够方便地为 Web 服务，并允许它们通过 Internet 被运行在任何操作系统上的任何语言所调用。举个例子，XML 已经成为网络中数据结构传送的标准，为了提高效率，C#允许直接将 XML 数据映射为结构，这样就可以有效地处理各种数据了。

1.2.4　完全的安全性与错误处理

语言的安全性与错误处理能力是衡量一种语言是否优秀的重要依据。任何人都会犯错误，即使是最熟练的程序员也不例外。忘记变量的初始化、对不属于自己管理范围的内存空间进行修改，这些错误常常产生难以预见的后果，一旦这样的软件被投入使用，寻找与改正这些简单错误的代价将会让人无法承受。

C#的先进设计思想可以消除软件开发中的许多常见错误，并提供了包括类型安全在内的完整安全性能。为了减少开发中的错误，C#会帮助开发者通过更少的代码完成相同的功能，这不但减轻了编程人员的工作量，同时也更有效地避免了错误的发生。

.NET 运行库提供了代码访问安全特性，它允许管理员和用户根据代码 ID 来配置安全等级。在默认情况下从 Internet 和 Intranet 下载的代码都不允许访问任何本地文件和资源。比方说，从网络共享目录中运行的程序，如果它要访问本地的一些资源，那么异常将被触发。若拷贝到本地硬盘上运行，则一切正常。内存管理中的垃圾回收机制减轻了开发人员的内存管理负担，.NET 平台提供的垃圾收集器（Garbage Collection，GC）将负责资源的释放与对象撤消时的内存清理工作。

变量是类型安全的，C#中不能使用未初始化的变量。对象的成员变量由编译器负责将其置为零，当局部变量未经初始化而被使用时，编译器将会提醒。C#不支持不安全的指向，不能将整数指向引用类型，例如当对象进行下行指向时，C#将自动验证指向的有效性。C#提供了边界检查与溢出检查功能。

1.2.5　版本控制

C#提供内置的版本支持来减少开发费用。使用 C#将会使开发人员更加轻松地开发和维护各种商业应用。

升级软件系统中的组件模块是一件容易产生错误的工作。在代码修改过程中，可能对现存的软件产生影响，很有可能导致程序崩溃。为了帮助开发人员处理这些问题，C#语言内置了版本控制功能。例如，函数重载必须被显式地声明，而不会像 C++或 Java 那样随意声明和使用，这可以防止代码级错误和保留版本化的特性。另一个相关的特性是，接口和接口继承的支持，这些特性可以保证很容易开发和升级复杂的软件。

1.2.6　灵活性和兼容性

在简化语法的同时，C#并没有失去灵活性。尽管它不是一种无限制的语言，比如它不能用来开发硬件驱动程序，在默认的状态下不能使用指针等。但是在学习过程中，你将发现它仍然是那样的灵巧。

如果需要，C#允许将某些类或类的某些方法声明为非安全的，这样一来，将能够使用

指针结构和静态数组，并且调用这些非安全的代码，不会带来任何其他的问题。

正是由于其灵活性，C#允许与 C 风格的需要传递指针型参数的 API 进行交互操作。DLL 的任何入口点都可以在程序中进行访问。C#遵守.NET 公用语言规范（Common Language Specification，CLS），从而保证了 C#组件与其他语言组件间的互操作性。

本节主要介绍一些与 C#语言相关的特点，在接下来的内容中，将对 C#语言的基本使用方法进行详细介绍。

先从最常见的数据类型开始讲解。C#数据类型可以分为两大部分：值类型和引用类型。

1.3 值类型

应用程序总是需要处理数据，而现实世界中的数据类型多种多样。我们必须让计算机了解需要处理什么样的数据以及采用哪种方式进行处理、按什么格式保存数据等。比如在科学运算中，不同情况下需要不同精度的小数，这些都是不同的数据类型。

其实任何一个完整的程序都可以看成是一些数据和作用于这些数据上的操作的总和，在本节中，将系统地学习 C#语言提供的数据类型以及使用这些数据类型时的要点。

C#的值类型可以分为以下几种：

- 简单类型
- 结构类型
- 枚举类型

简单类型有时也称为纯量类型，是直接由一系列元素构成的数据类型。C#语言为我们提供了一组已经定义的简单类型。从计算机的表示角度来看，这些简单类型可以分为整数类型、布尔类型、字符类型和实数类型。

1.3.1 整数类型

顾名思义，整数类型的变量值为整数。数学上的整数可以从负无穷大到正无穷大，但是由于计算机的存储单元是有限的，所以计算机语言提供的整数类型的值总是在一定的范围之内。

C#中有 9 种整数类型：短字节型（sbyte）、字节型（byte）、短整型（short）、无符号短整型（ushort）、整型（int）、无符号整型（uint）、长整型（long）、无符号长整型（ulong）、字符整型（char）。C#中的 9 种整数类型及含义如表 1-1 所列。

表 1-1 整数类型

数据类型	数据类型描述
sbyte	有符号 8 位整数
byte	无符号 8 位整数
short	有符号 16 位整数
ushort	无符号 16 位整数