Visual C++

图形用户界面开发指南

李博轩　等 编著

编程类专题

开发指南

清华大学出版社
http://www.tup.tsinghua.edu.cn

Visual C++ 图形用户界面开发指南

李博轩 等 编著

清华大学出版社

(京)新登字 158 号

内 容 简 介

本书通过大量实例深入浅出地介绍了 Visual C++ 图形用户界面开发技术,对 Windows 界面中最重要的组成元素分别进行了介绍。全书共 10 章,主要内容包括:Windows 用户界面制作基础、按钮控件、编辑控件、组合框控件、列表视图控件、树视图控件、菜单、工具栏、状态栏、框架窗口等编程技术。对每种编程技术,都给出了具有代表性的应用实例,使读者能够通过实例的学习,迅速掌握图形用户界面编程技术。

本书内容全面、深入,适合中高级读者、大专院校师生、企业技术开发人员学习参考,也适合各类高级培训班学员学习 Visual C++ 图形用户界面开发技术。

书　名: Visual C++ 图形用户界面开发指南
作　者: 李博轩 等
出版者: 清华大学出版社(北京清华大学校内,邮编 100084)
http://www.tup.tsinghua.edu.cn
印刷者: 北京市清华园胶印厂
发行者: 新华书店总店北京发行所
开　本: 787×1092 1/16　**印张**: 26.25　**字数**: 621 千字
版　次: 2000 年 11 月第 1 版　2000 年 11 月第 1 次印刷
书　号: ISBN 7-900630-68-6
印　数: 0001 ~ 5000
定　价: 46.80 元

前　言

我们设计的程序是否能够吸引用户，是否总能寄希望于程序精彩的内涵呢？“酒香不怕巷子深”并不是在任何情况下都适用。在一个软、硬件技术不断发展的时代，对大多数并非计算机专家的用户，在他们面临多种选择时，怎样才能使自己的产品脱颖而出？这对大多数程序，尤其是商用程序来说是一个相当重要的问题。

显然，别具一格的程序外观足以影响用户的选择。拥有友好而富于个性的用户界面的软件，往往具有更强的功能。这也是经过事实证明了的。

另一方面，随着开发工具的不断发展，程序员所要做的工作越来越少，而这必然导致系统的许多细节被隐藏起来，能供程序员自由发挥的空间越来越小。其最直观的表现就是，通过 Visual C++ 开发的应用程序具有极其类似的界面。如果你希望编写出富有个性的界面，就必须付出更多的精力。

漂亮的界面有利而无弊，可是我们怎么实现它呢？

本书对 Windows 界面中最重要的组成元素分别进行了介绍。对于每个元素，都通过大量经典实例深入浅出地向读者展示：如何使其具有更强的功能和更漂亮的外观。但是本书绝非代码的堆积，而是试图通过对经典实例的分析，使读者理解界面设计的原则和方法，并进而对 Windows 高级编程技术有更深的理解。

本书侧重 Visual C++ 在图形用户界面方面的应用，重点介绍如何利用 Visual C++ 实现应用系统的图形用户界面开发。全书按应用程序界面开发的不同主题来编排内容，分别讲述 Windows 用户界面制作基础、按钮控件、编辑控件、组合框控件、列表视图控件、树视图控件、菜单、工具栏、状态栏、框架窗口等。对于每个主题，都给出了 Visual C++ 在这一方面的要领，以及运用这类技术的实例和技巧，使读者能够通过实例的学习，迅速掌握图形用户界面编程技术。

本书所附的光盘中含有全部实例的源代码。

本书的特点在于讲述如何用 Visual C++ 实现应用程序的图形用户界面，而且将 Visual C++ 的图形用户界面特性以及技术难点融入到具体的实例中。不强调“大而全”，而侧重“专而精”。

参与本书编写的除封面署名作者外，还有吴灵、刘秀蓉、吉尚戎、吉二源、杜丽、何震声、宋淼、陈明、李洪声、刘海涛、李敏、刘志诚、朱志言、刘兵，另外，王宇红、李兵、刘海兰、姚文龙、李晓霞、向文兵、刘斌、张勇、张碧霞、孟文征参与了校对与录排工作，在此对他们的辛勤劳动表示感谢。由于水平所限，不足之处恳请读者批评指正。

作　者

2000 年 7 月

目　录

第 1 章　Windows 用户界面制作基础

除非应用程序只是给自己使用，否则它的价值必须由别人决定。应用程序的界面对用户的影响很大——无论程序代码如何高效，功能如何强大，如果用户发现它太难于使用，那么这个程序就不会得到广泛的应用。

本章主要向读者介绍应用程序界面设计原则、Windows 编程机制以及 Visual C++ 应用程序框架的结构和功能。由于现在 Visual C++ 为用户所做的工作越来越多，造成用户对 Windows 编程的机制的了解反而愈来愈少。只有充分理解 Windows 应用程序的结构和运行机制，才能真正掌握 Windows 编程。本书主要关注应用程序界面的设计，而如果希望设计出独特而有创意的界面，那么就必须对 Windows 内部机制有一个较为深入的理解。

本章要点：

- 界面设计基础；
- Windows 编程机制；
- Windows 应用程序结构体系。

1.1　用户界面设计基础

并不是只有拥有出色的艺术天赋，才能创建出优秀的用户界面——绝大多数界面设计规则与小学图画课上讲的完全一致。在那些课上介绍的组分、颜色的设计规则同样也适用于计算机屏幕，这与使用蜡笔在纸上画图没有什么不同。

对于使用可视工具编程的程序员来说，创建用户界面是一件非常容易的事情，简单到只需将控件拖动到对话框中，或在代码中调用一两个函数将自己绘制的位图设置为工具栏。然而正是这种简单性，使得创建出的应用程序越来越千篇一律，缺乏个性。关于这一点，可能读者也有体会。在用户界面设计上多花一些心思不仅能美观应用程序，提高易用性，而且能够增加程序对用户的吸引力（当然，从根本上说程序的功能才是最重要的因素）。

在设计用户界面之前，最好首先参考一下那些销路很好的软件界面，如果可能再参考一些销路并不很好的软件界面，这样可以得到很多有用的信息。而且自己在使用软件的过程中，对于用户界面也可能有一些想法和期望。例如对于某个软件，你可能对它的一些地方很满意而对另一些则不太满意，那么就可以在自己的设计中加以借鉴和改进。当然，个人的看法可能并不太适合大多数用户，因此也需要用户的确认。

成功的应用程序都会为用户提供多样化的选择。例如，Mircosoft Internet Explorer 允许用户使用菜单命令、键盘命令或直接拖放来进行文件拷贝。提供多样化选择，能够提高应用程序的吸引力。在应用程序中同一功能，应该至少能够通过键盘或鼠标完成。

1.1.1 设计原则

作为程序员,当然应该十分精通计算机,然而大多数用户可能并不太理解(或不关心)应用程序的技术问题。他们只是将应用程序作为一种更简单的解决问题的工具。一个好的应用程序界面应当将底层技术与用户分开,就像 MFC 类一样将底层功能包装在一个个的函数中。

用户界面设计的一个总的原则就是用户至上。例如应用程序的设计是否能够吸引用户;在没有帮助的情况下,用户是否能够比较容易地发现应用程序的不同功能;应该如何为用户提供在线帮助等等。所有的这些都要求程序员考虑问题时,必须更多地从用户的角度出发。

在实际使用中,软件和硬件并不能实现无缝结合,硬件的发展速度和软件的更新并不同步。这也就是说,错误几乎是不可避免的,因此错误处理也是用户界面设计的一部分。当出现错误时,通常的方法是显示一个对话框,并询问用户应该如何处理。而另一种不太常用的方法(但是更值得使用)就是自己后台解决,而不是麻烦用户。总而言之,用户关心的是完成特定的任务,而不是技术细节。因此在设计用户界面时,需要考虑到可能出现的错误,并确定哪些需要与用户交互处理,而哪些可以自己内部解决。

当应用程序出现错误时,如果需要与用户进行交互,一般是通过对话框完成,而在代码中则使用分支语句(例如 if...else 语句)。对话框是应用程序界面的一部分,它的设计对于应用程序的易用性也非常重要。

有一个笑话:在应用程序中设计很多对话框的程序员,在自己的人际交往中往往不能与他人很好地交流。诸如"A sector of fixed disk C: is corrupted or inaccessible. Abort, Retry, Ignore?"(如图 1-1)的消息对于大多数用户来说毫无意义。可以设想,服务员问您"可能饭菜卖完了或者厨房失火了。取消、重试还是忽略?",这时您将如何回答?实际上,这告诉我们应该以用户能够理解的方式来提示选择,例如图中下方的对话框中的消息"There is a problem saving your file on drive C. Save file on drive A, Don't save the file?"就友好得多。

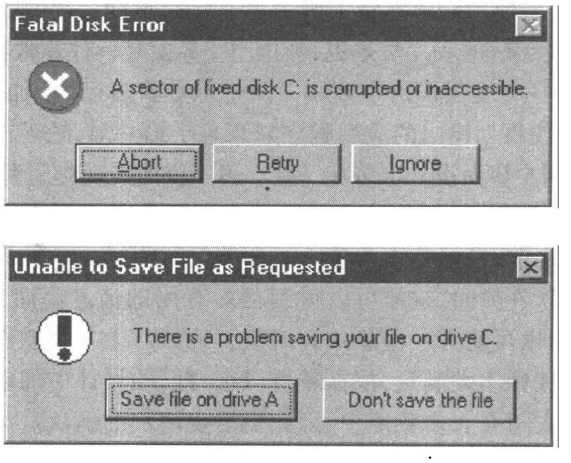

图 1-1 使用更易于理解的语言

在应用程序中设计错误信息对话框时,首先考虑这个信息对于用户是否有用;其次考

虑按钮所给出的选项意思是否清楚、选项和消息是否适合等。此外，如果仅仅显示一个错误信息会给用户造成不好的印象。关于这一点，可能读者也有体会，例如 Windows 著名的蓝屏现象。

并不是所有的错误都需要用户进行处理。对于在代码中就能够处理的错误，就无需通知用户，或者以不中断当前进度的方式进行通知。关于这一点，Word 里面的“自动更正”功能就是一个很好的例子：如果常用词出现错误，则自动修改；而如果不常用的词出现错误，则以红色波浪线表示它，以便用户以后修改。错误处理的技术有好几种，但哪个更适合就需要程序员自己作出决定。下面是一些通用的规则：

- 在“编辑”菜单中添加“撤消”命令，这要比打断用户当前的操作，并显示对话框询问是否要撤消好一些。
- 在状态栏显示提示，或使用闪烁的图标表示错误——这样用户在方便的时候进行处理。
- 当错误很明显时，直接改正它。例如当用户保存文件时，如果当前驱动器已满，则检查其他驱动器的空间，如果有可用空间则保存文件；同时在状态栏上显示信息，告诉用户程序都做了什么。
- 并非所有的错误都需要立即解决，这时可以采用将错误记录在日志文件中的方法，供用户方便时浏览。例如在 Windows 2000 中，如果登录时网络连接出现问题，则将问题记录在系统日志中，用户可以通过系统信息管理器查询。

1.1.2　Windows 界面规则

Windows 操作系统的一个重要优势在于它为所有应用程序提供了相同的界面。这样，一个有经验的用户能够很快掌握原先并没有使用过的 Windows 应用程序。菜单就是一个很好的例子：大多数 Windows 应用程序遵循“文件”菜单在最左边，其后是“编辑”、“工具”菜单，最右边是“帮助”菜单的规则。可能有人认为“文档”要比“文件”更合适，或“帮助”菜单应该排在最左边，并且完全可以按照这些想法设计应用程序的界面。然而，这样就会给用户带来不必要的混乱，而且当用户在切换到另一个应用程序时，还必须调整一下使用方式，这样就降低了程序的可用性。此外菜单命令的位置也很重要。例如，用户希望在“编辑”菜单下找到“拷贝”、“剪切”和“粘贴”命令。总之，除非有充分的理由，否则不应该改变公认的 Windows 界面规则。使用已经存在的界面设计规则，不但能够使应用程序的可用性更好，而且会使自己的设计有个很好的开端（也就是说，用户界面的布局有个大概的框架）。

检测用户界面可用性最好的手段，就是使用户自始至终地参与到设计过程中。可用性检测的一个关键就是用户是否能够较为容易地了解如何使用应用程序的功能。例如在 Windows 3.1 中，很少有用户知道使用 Alt + Tab 键能够在应用程序间进行切换，因为几乎没有什么线索能够帮助用户发现这一点。要测试这一点，可以要求用户完成某个任务而不加任何解释（例如，要求用户从信函模板创建一个新文档）。如果他们不能完成，或中间经历了多次尝试，那么就需要重新考虑设计了。

1.1.3 界面布局原则

应用程序的布局不仅影响其外观,而且对其本身的易用性有着举足轻重的作用。这包括控件的位置、元素之间的协调性、空间的使用以及设计的简单性等。

1. 控件位置

在大多数界面设计中,并非所有的组成元素都具有同等的重要性,因此必须保证常用的重要元素处于最明显的位置。

绝大多数语言都具有从左到右和从上到下的书写顺序。因此当用户观看屏幕时,会从左上角开始——最重要的元素应该放置在那里。例如,如果在屏幕上显示包括消费者信息的表单,则应该首先显示姓名字段;而诸如"确定"或"下一步"之类的按钮,则应该在屏幕右下方显示。这是因为,通常用户只是在完成了整个表单后,才会单击这些按钮。

将元素和控件分组也非常重要。将信息、功能相近或关联的控件分组排列,要比分散排列好。分组排列大多是通过组框等控件完成的。

2. 协调性

用户界面必须保证其协调性。协调的外观和感觉将使应用程序看起来很舒服。相反,缺乏协调性,会使应用程序看起来很混乱,从而使用户低估其功能和稳定性。

为了可视协调性,需要在开发前有一些大概的规划。例如,控件的种类、尺寸以及字体等等。这时先做一个模型将会有助于后续的开发。

由于 Visual C++ 提供了多种类型的控件,这很容易导致你产生将各种类型的控件都加以使用的想法。在开发中必须注意避免出现此类想法,而应该仔细选择最适合应用程序的一些控件。例如列表框、组合框、列表视图控件和树视图控件都能用于显示一系列信息,但是对于开发者来说,应该尽可能地选择单一风格的控件。

此外,还需要注意所用的控件是否合适。例如将编辑框设置为只读用来显示文本不如使用静态文本控件更合适。保证一致性对于界面设计也十分重要,例如如果在某处将允许用户编辑的文本属性设置为黑底白色,除非有充分的理由,不要在其他类似之处使用其他颜色配置。

出于易用性的考虑,在应用程序的不同窗口中也需要保证一定的一致性。如果在应用程序的一个对话框中使用灰色背景,而在另一个对话框中使用白色背景,这会使整个应用程序看起来缺少内部的联系。总的原则就是选择什么风格就坚持什么风格,即使在某些情况下需要重新设计某些功能。

3. 一致性

抽象地说一致性就是某个对象的可视化线索,实际的例子到处都是。例如,自行车上的把手就是手扶的地方。又比如,在 Windows 应用程序中,"打开"工具栏按钮图标就是用于打开文件夹的。通俗地讲,一致性就是看到其外观就能猜到其功能。

用户界面也需要使用一致性。例如,应用程序中的具有三维效果的按钮表示它们可以被按下;而同时又存在具有平坦风格的按钮,那么就失去了一致性。用户无法确定一个按钮到底是否是命令按钮。有些情况下平坦风格的按钮可能会更合适,例如游戏或多媒体应用程序。但是,在应用程序中保持一致性是压倒一切的。

编辑框控件也提供某种方式的一致性。一般来说,允许用户进行编辑的控件具有边界和白色背景。当然你完全可以将控件的风格设置成为无边界(这就有些像静态控件),这时用户就不会很快意识到它是可以被编辑的。

4. 空间的使用

应用程序中空间的使用也很重要,它有助于改善程序的外观和对某些元素的强调。当然空间并非一定要使用。然而过多的控件拥挤在一起,会增加寻找的时间降低效率。在设计时,需要综合考虑以确定最佳的分布。

使控件间距一致,以及控件垂直和水平对齐也会提高应用程序的易用性。这与报纸中文字的有序排列会方便阅读的道理一样。在 Visual C++ 中提供了一些工具用于调整布局,当读者使用资源管理器时会经常用到。

5. 简单性

界面设计中最重要的准则可能就是简单性了。简单即美,这也是艺术中的一个准则。如果应用程序的界面过于复杂,也会使用户望而却步。

用户界面设计中经常出现的一个误区是试图以真实世界中的对象作为设计的模型。例如要设计一个用于完成保险表单的应用程序,很自然的反应就是创建一个实际表单的翻版作为界面。而这将会导致一些问题:纸的尺寸和形状与屏幕上的不同,完全照搬就会使界面元素限于文本框和复选框,而这并没有为用户带来任何方便。

最好的方法是设计自己的界面,例如设计打印功能界面。通过逻辑关系创建不同的标签或链接表单,从而使用户无需滚动翻页即可得到全部信息。此外还可以使用列表控件为用户提供选择,这样能够减少用户的输入量。只要多从用户的角度考虑,就会设计出相对简单易用的界面。

简化界面的另外一个有效方法,就是根据某个功能的使用频率决定其显示方式。例如 Word 2000 中会智能地根据某个按钮(控件)的使用频率决定其显示与否(位置)。有时提供默认值也会简化界面。例如,如果大多数用户都喜欢使用加粗的文本,那么将加粗风格设置为默认值,会比用户自己主动选择要方便一些。

使用向导也能简化复杂的操作,这一点使用 Windows 软件较多的读者都深有体会。

测试应用程序简单性的最好方法就是实际使用该程序。如果在没有帮助的情况下,普通用户不能很快完成某个操作,那么就需要考虑重新设计了。

6. 颜色

多彩的界面会增加应用程序的吸引力。现在使用的绝大多数显示器能够显示数百万种颜色,这往往会导致用户对颜色的过度使用。颜色的效果是千差万别的,用户的喜好也

会因人而异。因此在设计应用程序时,需要注意不同文化中颜色的差异。一般来说,在颜色的使用上应该持保守的态度,尽量选择软色调和中性颜色。

当然,对颜色的选择也需要考虑用户的情况。例如,在设计儿童软件时,亮红色、绿色和黄色是很好的选择;而对于金融或银行应用程序,这些颜色会产生反面影响。使用少量的亮色能够有效地突出重要区域。作为原则,在应用程序中应该限制颜色的数量,而且应该保持前后的一致性;应该尽可能使用标准 16 色调色板,这样会扩大应用程序的兼容性;此外,色盲用户也是需要考虑进内的。

7. 图片和图标

在应用程序中使用图片和图标,也能提高应用程序的吸引力。图片所传达的信息有时是文字所远远不及的,同样,其致命的弱点在于不同人的理解可能也会不同。

工具栏按钮上的不同图标,形象地表达了不同的功能。但是当用户不能确定某个图标的功能时,就会产生反作用。在设计工具栏图标时,应该参考其他应用程序以得到设计的标准。例如绝大多数 Windows 应用程序,会将打开文件夹的图像作为"打开"命令工具栏按钮图标。如果你的应用程序违反了标准,就会给用户带来混乱。

此外还需要考虑文化的差异,例如美国应用程序经常将 作为"邮件"按钮的图标。而在其他国家中用户却并不将该图标看作邮筒,从而对按钮的功能理解出现偏差。

在设计应用程序时,应尽量使用 16 色调色板图像,以提高应用程序的兼容性。

8. 字体

字体是用户界面的重要组成部分,因为它们通常用于与用户进行重要信息的交互。在选择字体时,必须保证它能在各种不同分辨率的显示器上都有良好的可读性。

除非计划将应用程序与字体一同发行,否则最好使用标准 Windows 字体,例如 Arial、New Times Roman 或 System 字体。如果用户的系统不包括指定的字体,则系统将使用其他字体,这就有可能造成实际的显示与预期不一致。例如,CorlDraw 9.0 的英文版在 Windows 9x 下字体就无法完全显示,造成的后果就是文字不全。如果应用程序将在世界范围内发布,则需要为每种语言选择合适的字体。此外还需要考虑字体所占的空间,例如中文比英文所占的空间大 50% 。这里还要旧话重提,一定要保持一致性,类似功能的文本要使用相同的字体。

1.1.4 用户辅助模型

无论使用多么优秀的用户界面,用户总会需要帮助。应用程序的用户辅助模型包括在线帮助、帮助手册、工具提示、状态栏、"这是什么?"以及向导等。

与应用程序其他部分的设计一样,用户辅助模型的设计也应该在开发前进行。模型的内容则与应用程序的复杂程度以及预期的用户有关。

在线帮助是应用程序的重要组成部分——它通常是用户寻找问题答案的首选。在设计帮助系统时,要时刻牢记它的基本用途是回答问题。当创建主题和索引时,一定要从用

户的角度出发。例如,“应该如何格式化页面?”要比“编辑、页面格式菜单”更容易定位问题。此外,一定要保证上下文敏感性,也就是说当用户选择了“格式”然后按下 F1 键,那么应该出现的是针对这个词的帮助,而不是出现帮助主题。印刷或电子的帮助手册是另一种非常有用的工具,它们能提供在简洁帮助主题中难于传递的信息。此外,工具提示、状态栏等也能够为用户提供很大的方便(使用 Visual C++ 能够很轻易地为应用程序添加此类支持)。

1.2　Windows 编程机制

要更好地使用 VC++ 进行 Windows 编程就需要了解其内部运行机制,这主要包括消息驱动和 MFC 类库。

1.2.1　消息驱动

进行 Windows 编程之前,必须了解一些 Windows 的运行机制。在 Windows 操作系统中,应用程序主要是以窗口的形式存在。消息传递是 Windows 操作系统和应用程序间、不同的应用程序间互相交流的主要形式。每一个应用程序都对相应的消息作出执行动作,这种执行动作称为消息响应。

消息的产生和发送以队列的形式进行。消息响应遵循一定的顺序。MFC 类库为这种消息响应机制提供了比较完整的处理功能。构成 MFC 类库的很多基类都具有处理相应消息的函数。在 DOS 等控制台模式程序中,对诸如鼠标、键盘等的控制是通过轮询(分别定时查询这些设备的输入请求)完成的。而在 Windows 环境中,这些控制是通过消息完成的,因此 Windows 也被称为“基于事件的,消息驱动的”操作系统。在编写 Windows 程序时,必须保证与 Windows 平台下同时运行的程序协同合作。

在 Windows 应用程序中,窗口既是一个可视界面,也是应用程序的控制消息发送接收端。窗口不仅提供了可视化的应用程序的命令,也是 Windows 消息的产生和响应的地方。图 1-2 所示为一个标准的 Windows 窗口,其中包括客户区和非客户区。非客户区包括窗口的边框、菜单和标题区。客户区是窗口中除了非客户区剩下的部分。通常情况下,窗口的非客户区由 Windows 维护,客户区由应用程序进行管理和操作。Windows 下的窗口有许多类型,如对话框、属性页、控件台模式的 DOS 命令框、工具栏和菜单栏等。甚至按钮、列表框等控件也可以被看作一个窗口,实际上在 MFC 类库中,这些控件的管理类几乎都是由 CWnd 类(窗口管理类)派生出来的。每一种类型的窗口都有一套适于它们自己的公用操作,如移动、改变大小、隐藏、显示、允许使用和禁止使用等。

在 Windows 中,每个应用程序都是基于事件和消息的,而且包含一个主事件循环。该循环持续反复检测是否有用户事件发生。每次检测到一个用户事件,程序就对之作出响应。这些事件包括移动鼠标指针、按键、单击或双击按钮等。当 Windows 接收到这些事件后,会产生一些相应的消息。应用程序接到这些消息后,按每一个消息的专门用途,产生