如何开发Windows环境下的图形用户界面应用程序

崔洪斌　王　智　编著
李文博　审校

- 界面设计原理
- 界面设计三要素
- 事件及事件驱动编程
- 菜单、按钮、对话框
- 单选钮、复选钮、列表框和组合

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
URL:http://www.phei.co.cn

如何开发 Windows 环境下的图形用户界面应用程序

崔洪斌　王　智　编著
李文博　　审校

電子工業出版社

内容提要

Windows 环境下的图形用户界面(GUI)以其一致性和友好实用的特点广泛应用于各种领域。本书重点讲解了什么是良好的用户界面，如何生成方便用户使用的一致性的界面，从而使开发软件具有连贯性，并能引导用户正确使用软件。

本书首先阐述用户界面的优劣，并以实例说明其在程序设计中的重要作用。然后分别讲述 GUI 开发的各种要素，包括事件驱动编程、菜单、按钮、对话框以及列表框等要素的编程细节。

本书适合在 Windows 环境下从事科研、开发工作的技术人员阅读，尤其适合 Windows 环境下应用程序的开发人员参考。

如何开发 Windows 环境下的图形用户界面应用程序

崔洪斌　王　智　编著
李文博　　审校

责任编辑：夏春和(特邀)　王世忠

*

电子工业出版社出版
北京市海淀区万寿路 173 信箱(100036)
电子工业出版社发行　各地新华书店经销
北京志达排版公司排版
北京科技印刷厂印刷

*

开本：787×1092 毫米 1/16　印张：16　字数：389.38 千字
1997 年 2 月第 1 版　1997 年 2 月第 1 次印刷
印数：1-4000 册　定价：28.00 元
ISBN 7-5053-3895-1/TP・1676

引　言

顾名思义，本书是讨论用户界面设计的，笔者写这本书是因为对用户界面设计的现况感到不满。在当前 Windows 市场潮流下，这一论断似乎令人感到奇怪。然而，我们的确发现越来越多的应用软件的界面让用户无所适从。这并不是说以前就没有不好的用户界面——当然有。只是不好的用户界面在当今受到更多的抨击（由于市场意识的增强，界面的重要性为更多人所理解）。这要求我们程序员应该为用户设计更好的界面。这也是本书阐述的内容。

用户界面问题实际上分为两方面——一是对它的理解，二是如何去建立。这两方面本书都要讨论。如果读者从没学习过软件设计的原理，也不用着急。本书将在书中提供很多用户界面设计理论中的现成实例，只要读者喜欢，就可以尽可能多地将它们用在程序中，而且不必担心会用错。本书将尽可能讲得浅显一些。

本书主要讲述 Windows，同时指出怎样在 Windows 下建立真正强大的用户界面——不只从理论，而且从实际上给予详尽的论述。怎样在对话框上下文中处理一个列表框？当用户双击一个工具按钮时怎样处理？我们将在书中探索这些问题的答案。本书将以许多代码段说明这些问题，在有些时候，还给出完整的程序清单。

本书旨在为程序设计人员提供设计良好用户界面的一个思路。因而，书中讨论的各种内容有些看起来与界面设计无关，但实际上是有用的。它们包括大量的实例，少部分反例，一些难题和一些其他协调问题，所有这些都是有目的的，也就是为了开拓思路，用不同的方法考虑问题，其中一些最好的界面设计方案来自横向思考——采用各种新思路来考虑问题，甚至那些以前从未想到过的。本书不把现成的结果直接写出来，那样将妨碍读者自己的思考。

编写本书是一项艰巨的劳动，是集体智慧的结晶，除了封面的编者以外，蔡汇锦、屠义明、周锐、田彦茹、赵文、于朝晖、李明、宋伟等也参予了编写和整理工作，从事多年计算机编程的李文博老师审校了全书的内容，在此深表感谢。

目　录

第一章　理解用户界面

欢迎加入我们的队伍，向用户界面设计这一困境挑战。之所以这样说有两个原因。一是我们发现用户界面设计已消耗了许多本来极富有工作效率的人员和公司的时间及精力，他们看上去已陷入困境；二是我们发现（相信用户也已发现）许多界面设计得一团糟，好像设计时并没有考虑到产品应该是什么样，谁应当去做以及怎样去做。

这是不好的。

幸运的是，这种不好的现状可以改变。用户界面设计既不是变魔术，也不是炼丹法。当然，必须承认，有时候是有些类似。从根本上说，用户界面设计就是构造一个用户可用的界面。

基于此，就有一个任何人都不会忽视的论断，也确实是一个事实，那就是不论你的界面多么漂亮，图标画得多么精致，如果软件产品对使用它的人来说不可用，那这个界面就没有任何意义了，结论就是这样简单。

一个有助于了解用户界面设计本质的好的练习就是多想想各种常见产品的用户界面（不仅是一般软件产品）。

例如，一扇门的用户界面是什么？

这通常看起来像个难题，究竟是什么意思呢？门有什么用户界面，手放在把手上，拧或推，门就开了，里面并没什么。

但实际上这就是我们所说的，门把手的位置及其形状可以使门的使用者了解门的使用方法，即门转动的方向并知道如何打开它。有一本书中描述了一种不符合上述条件的门，其结果是这些门几乎不能被打开。并不是这些门不能使用，如果指明了转动的方向，就可以知道门肯定能打开，但现在的情形是没有任何信息能够帮助我们判断何种方向可以打开它。

门把手是长的、扁的金属杆，直接放在玻璃门的中部，这一点很明显，没有其他的可视性暗示，也没有下面的线索：

- 应当向哪一边施加压力（左或右）；
- 门向哪边旋转（里或外）。

人们不能使用门，这是个极端的例子，但这里还有个稍好的——有人曾做了个试验，他们将门把手装在有合页的那边，如果此时推没有把手的哪一边，门很容易打开。这时候人们也打不开此门，他们站在那里，与门把手较劲，又拉又推，但无济于事。

这一试验的原因是此门的开启方法与人们概念中的方法不一样，这带给我们有关用户界面的第一个要素：

- 使用你的界面的人对如何使用它已先有了一个假设。

这一条在任何地方都适用，不只是在使用软件上。人们的头脑中对这个世界如何运转都有一种模式。如果没有的话，毫无疑问，生活将一片混乱。人们知道如何使用电梯，如何打电话，如何使用一扇门，这些都有一个模式。人们对所做的任何事几乎都有一个模式，如果我们头脑中的模式与现实的实际情形不符，就会遇到麻烦。

例如，我们都有驾驶汽车的经验，但如果其他人的经验与我们不同，那时会怎样呢？不要笑——这是可能发生的。如果我们学会了在中国驾驶，那么在英国会怎样呢？他们都行驶在错的一边（至少我们这样看），而对他们来说是在正确的一侧行驶，而我们却成了白痴。

当然，要点并不是究竟是靠右行驶还是靠左，两者都适合于不同的已经被广泛接受了的各自社会习俗。重要的是，头脑中存在的模式使我们在做某些事时也许有些不习惯，这些模式给予我们一种思维框架，运用此框架去理解某种事情以及其表现方式。

这种情形在编写软件时又是什么样的呢？很简单，人们对所用的软件如何工作都已有假设，这些假设也许正确或不正确，但不管这些假设是否正确，人们都会依其行事。作为界面设计人员的工作，就是使用户头脑中的假设与软件的实际工作情形相一致。

要做到这一点，应注意下面几点：

- 确保软件前后一致；
- 给出如何使用软件的规则；
- 不要让用户不知所措；
- 给出好的反馈信息；
- 使用户有一定的权力以灵活使用软件。

上述每一个要素还需要一些解释，并且每一要素在第二章中都自成一节，可称为用户界面的五要素。每一个要素的内容都不会很多，也就是二、三页左右的讨论，但又全都是重点。因为没有这几个要素，就无法设计出能真正使用的软件。

第一个要素称为一致性。

第二章　界面设计原理

2.1　使界面一致：不要让用户困惑

一致性是好的用户界面设计的基础，它可使用户基于以往使用软件的经验，预测出在某些特定情况下该软件是如何运行的。

界面的一致性是指采用在其他产品中已给定的某些东西，而在设计我们的软件时就很少考虑到为什么要这样。例如，如果每一次上了汽车都要想想如何转动方向盘，那情景是多么糟糕。若有时方向盘向左转动因而使车也向左转，而有时向右转动而使车向左转，如果我们真有这样一辆车，则要立刻去修理它。

汽车的这种界面显然很令人讨厌，但有些软件的错误与之相比毫不逊色。某些软件包在扩展地使用键盘功能键时并没有提供一致的映射，有时 F9 表示“是”，有时又意味着“删除”（这种情形我们亲眼见过）。显然，同样的键既表示“是”，又表示“删除”，这将产生混乱以至危险。而这对编程人员来说，是根本不应发生的。

Microsoft 在编写 Windows 3.1 时遇到的最大问题之一就是在许多地方缺乏一致性界面，如打开及保存文件、选择颜色及其他通用操作。在此之前，关于打开和保存文件还没有一致的办法。

如果用户想在程序中使用 File Open 对话框（几乎每人都需要），就必须自己编写。因为每人都编写 File Open（或 Save）对话框，因而就产生了很多不同的完成这件操作的办法。

在 Windows 3.1 中，微软公司提供了一种通用对话框，作为编程人员，就有办法在不同的应用程序中使用相同的 File Open 对话框，这样就在不同应用程序中为打开或保存文件创建了一致的界面。

我们经常听到很多人坚持说：“我能写出比它更好的文件打开对话框”。这话没错，但从用户角度来看，这并没多大作用。用户不想在一个程序中使用更好的方法来打开文件，而是想在所有程序中用一致的方法打开文件。一旦用户已掌握了一种打开文件的方法，他们就不想再学习新方法了。

应用程序中的一致性非常重要，研究表明，用户倾向于在 Windows 之类的图形环境中更多地使用应用程序，而不是在 DOS 之类的命令行环境下使用。用户使用应用程序越多，就越希望应用程序拥有相同的操作方法，而厌恶那些操作方法不同的程序，即使这些应用程序能拥有更强大的功能。为什么呢？因为它们没有采用与用户已经习惯了的那些操作方法一致的方法。

如果对完成某个任务（如常见的打开文件对话框）已有了标准办法，毫无疑问应当采用它。如果还没有用户所需要的这样的标准方法，在采用新方法代替现行方法时就要仔细考虑。

一致性对于设计用户所使用的工具也是很重要的。现在，许多应用程序提供工具条以让用户选择所用工具。然而，有时这些工具的工作方法是不一致的，如果该工具一会儿执

行甲功能，另一时刻又执行乙功能，用户就会感到此工具难以使用。

这里也有其另一面，如果相似的工具的操作方法也相似，应确信用户能看出其一致性。一个工具操作方法的轻微变化就不必用两个工具代替它，应当使这一工具能以这一种相似的方法运作。

例如，在一个绘画程序中有一个 Pencil 工具，它可用来使用所选中的颜色画单象素线。一个用户建议这个 Pencil 工具应当加一个内嵌橡皮擦，用户对该功能的要求是：

- 如果绘画时铅笔使用的颜色与所选择要画的颜色不同，就画上所要画的颜色；
- 如果绘画时铅笔使用的颜色与所选择要画的颜色相同，就擦去所画颜色，画为白色。

这个要添加的功能很简单，并且保持了一致和简单的模式，使用户在使用时很容易掌握。

然而，有些用户不喜欢此功能。虽然他们知道该功能的作用，但就是不想照这样来用，所以对该 Pencil 工具加了个选项，使它可关掉此功能——现在，Pencil 工具就总是画出它选中的颜色，而不管绘图开始时的颜色是什么，见图 2-1。

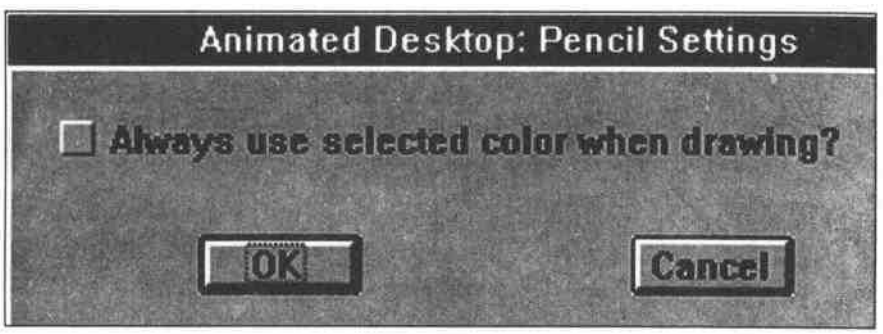

图 2-1 使用户设置 Pencil 工具选项的对话的屏幕显示，让用户选择想要的方式

这种 Pencil 功能进行选择的能力使用户可以建立与他们所需的工具使用方式最为一致的使工具如何使用的模式，因而，它是一种带有灵活性的一致性。

> **一致性**：使界面一致，否则用户要责怪你。

当然，有时保持一致性是不可能的。在有些场合，我们不能让用户根据他们的需求来工作（这种情况甚至很多）。但我们此时仍要尽可能地保持产品工作方式的一致性，这一点仍然很重要。记住，如果不这么做，用户就不会去使用该产品。

2.2 使界面合理化：引导用户使用正确方法

用户界面的第二要素，与上面谈到的一致性显然关系密切（给这些要素一个适当的名字是很困难的），是帮助信息。前面的一致性要求对用户的操作提供相同的回应，而帮助信

息要求对用户完成的某些操作提供清晰、明显的路径。

这里的含义是什么呢?很简单,好的用户界面在特定情况下应能给出完成下步操作的线索。为什么要这样做?在试图完成想要做的工作时,用户不会迷惑混乱(用户这时想做的,即使正是我们尽力阻止他去做的事情,他也会去做,因为我们也是用户,所以我们知道这点)。

当然,在日常生活中,也有很多类似的线索告诉我们下一步怎样做。例如驾车时,从路面情况就能知道该到哪儿了,比如路形、路标、路上的条纹,以及路上的突起不平部分等等。

相反,如果在一块盐碱地中部开始驾车就完全不同了,这里没有路,没有路标,无法知道向哪里去,实际上向哪里开都可以,这是令人非常害怕的事。由于可以自由地向任何方向走,因而也无法知道走哪一条路更好,或者即使有最好的路也不能知道。

在驾驶时如果没有任何路标,就有可能完全迷路,在二次大战的非洲战役中就是这样。人们在沙漠中迷路以至死亡,而此时仅离城市或宿营地数百码之远。沙漠中没有任何标记,并且外形又如此相似,使人们无法辨别方向。

这种事读者也可能遇到过,他们不会死,但可能不知道如何去办,有时完全不知道。看一个例子,比如C:>就没有任何帮助,它没有给出任何提示,使得设计者最终去寻求一个直观的界面,例如 Macintosh 或 Windows。

通过让人们处于一个可视的环境中,窗口化界面提供了关于用户下一步如何动作的反馈信息。当然,它还需要改进,并非完全直观的,但它给用户提供了组织及显示有关信息的方法的指南,因而使得信息可以逻辑有序地浏览。

列表框总是给用户提供一套选择,这点是一致的,但列表框所提供的这组选项的内容都是相关的,例如文件名,这对指导用户正确操作是有帮助的。如果一个列表框在不同时间使用时提供了根本不同的选项,那并没有违反一致性原则,因为总是在使用这个列表框。但这时一定违反了有助性原则,因为它无法告诉用户如何分辨哪一项目前很重要,哪一项不重要。

指南可被看作一种使用户缩小有效的选择范围的方法,它告诉用户在当前点的一些必要信息,诸如是否能做,怎样更容易,以及当前做什么合乎逻辑等。

这一点比想象的还要重要,因为作为产品设计人员,已经知道产品怎样工作,哪些动作之间可能不兼容,哪些动作在哪一上下文中关联。然而,用户对此一无所知。所以,设计人员有责任将程序内部重要的各种动作间的相关性告诉用户,这并不是把所有的内部相关性都告诉用户,而只是那些对用户来说应当知道的重要相关性。

例如,图 2-2 给出了一个绘画程序。它包括一套绘画工具,例如 Pencil,Paint Bucket,Spray Can 以及 Paint Brush 等工具,这些工具根据逻辑上的相似功能分成不同组。用户如果选择了一种绘图方法,就不能选择另一种方法,从工具的分组情况来看就更加清楚了。

如果使用 Pencil 工具,就不能使用 Paint Brush 工具;如果使用 Paint Brush 工具,就不能使用 Paint Bucket(填充)工具;如果使用 Paint Bucket(填充)工具,就不能使用 Spray Can 工具。以此类推,工具设计使用户一次只能用一种工具,选择了新工具就自动放弃了旧工具。

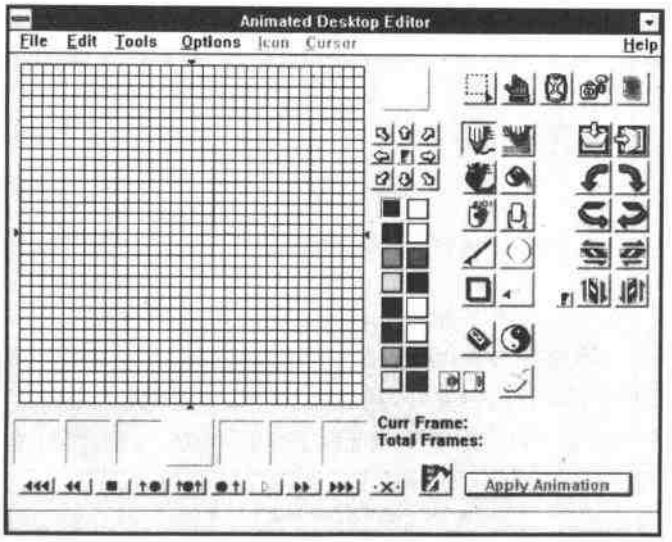

图 2-2 图标/位图编辑器。注意，工具是以其所实现的逻辑上的相关功能来划分的（如绘图），这可使用户一眼看出哪些工具是逻辑上相关的，而哪些在功能上不同

这一点很重要，因为它向用户提供了产品如何工作的指南。当使用一种绘图工具时，用户会发现不能使用其他工具。因而，如果让用户面对一套互相排斥的工具时，应确信他们能发现这一点，这一点十分重要，所以要让用户了解这一点。

第二套工具（绘图工具上方的箭头键）是各自独立的，因为他们执行不同的功能（允许用户四处移动图象）。然而应当注意，图象移动功能与绘图工具是独立无关的，换句话说，可以在任何时候移动图象，而不必考虑当时所使用的绘图工具。

从用户观点来看，这一特点是有帮助且易于学习的。用户会想："我能在任何时候移动图象"，他们这样认为并且确实可以做到。但是，假如把箭头与绘图工具分到一组，就让用户觉得它们之间存在着相关性，而实际上却没有，由于视觉的暗示会使用户认为："这些是绘图工具的一部分"。

用户经过一段时间学习后，会发现箭头实际上不是绘图工具的一部分，这说明界面设计得不够好，当然不是不能用。但假如在箭头键与绘图工具之间确实创建了相关性，比如用户不能移动图形，除非他们首先选择了 Pencil 工具。这样做会使界面变得很糟，因为：

- 并没有帮助性信息指出使用箭头键之前必须先选择 Pencil 工具；
- 对于移动图象时必须先选择 Pencil 工具这种做法，并没有明显的原因。

可以看到，如果界面真的做成这样，对用户来说是很容易弄糊涂的。虽然用户最终会学会如何使用，但不会有人喜欢它。当然，程序的功能根本就没有修改，仍然执行绘图和移动的相同功能。

帮助信息：设计软件时要引导用户正确地使用该软件。

在这个例子中，只是界面的修改才使得软件有了不同。这说明界面可使得一软件具有帮助性并能引导用户使用它，或是根本没法使用。

2.3 使界面健全易用：不要让用户不知所措

用户界面的第三个要素是不要让用户困惑，因为用户经常会说："哦，我如何再进到这里呢？那个菜单选项哪儿去了？我应按哪个按钮？"以及其他相似的问题。这种情况会在很多初学用户那里看到，对一些较有经验的用户，也可能如此。

迷惑与有助性是相反的，有助性引导用户采用正确方式，而迷惑使用户在界面中不知所措。我们所谈的是在软件中向用户提供完成某项特定任务的方法引导的重要性，但同样重要的是，不应当人为地限制用户采取某些有益的尝试，这正是产生迷惑的原因。界面应当含有足够的引导信息以使新用户（此处的新用户并没有贬意，仅指那些初次使用产品的用户）能在界面中完成有意义的任务，并尽可能少地产生疑问。然而另一方面，界面也应允许有能力的用户完成较复杂的任务，而不应干预太多。

当与人们讨论用户界面时曾有一个趣闻，我们问他们"你们是需要带有详细帮助信息的长卷毛狗式的用户界面，还是赛车式的界面？"。所谓长卷毛狗就是不停地在主人脚下跑、跳、舔并轻声叫喊的狗，让你整天被它纠缠。我们确信大家都一定见到过此种界面，"你要继续进行吗？你确认了重新格式化硬盘吗？"之类的无休止的问题。从琐碎的到危险性的问题它都要用户做出回应，一直纠缠着用户。

这种界面对于新用户也许较好，但太过份了点。软件使用一段时间以后，用户就开始厌烦这些问题了，以致于不管琐碎的还是重要的问题都不给回应。

另一方面，赛车式界面很强有力，但很难驾驭。此类界面几乎能让用户做任何操作，但至于用户的操作是否好的、坏的还是无破坏性的都一律不给予反馈信息。长卷毛狗式的界面对于一切操作都寻问用户，而赛车式的界面则一律不问任何问题，它假设用户是万能的，无论用户如何瞎操作都行。

这是用户界面设计中两种极端的情况，好的界面应在两者之间。这是真正的技术，使设计的界面既要能足以把工作做好，但又不能让它走向上述极端，即功能强大到让用户不知所措。迷惑正是用户利用了界面提供的过于强大的功能而得不到足够引导时产生的，而引导帮助过多，但功能不够强大的界面则过于死板。

具有迷惑性的界面当然也迷惑菜单与对话框之类的组件，尽管设计者认为一切都已做得很好，迷惑仍然可能乘虚而入。有些时候，"视窗"选项不是被放在"编辑"菜单中了吗？除非真是布局合理，否则迷惑难以避免。

迷惑：使界面功能强大以使人们能完成工作，但又不可过于强大以使人们不知所措。

正如上面所提到的，在产品中采用逻辑分类的方法是很重要的。人们对发掘一个产品

如何工作是有兴趣的，但我们必须引导他们如何去做。如果界面中的相关功能被安放得十分散乱，人们就会讨厌该产品。

2.4 使界面友好：给出好的反馈

设计好的用户界面的一个重要部分是给用户提供恰当级别及恰当种类的反馈信息。这就是界面问题中的下一个要素，渴求信息，即急于得到好的用户界面信息。

要想设计好用户界面，反馈信息是重要的可视部分。简单地说，反馈信息就是用户在使用产品时收到的有关他们已做了什么、怎样做的以及结果是什么的信息。这三部分缺少任何一个，反馈信息都是不充足的。让我们对每一部分加以详细讨论。

2.5 用户已做了什么

当用户执行一操作时，他们想知道他们已经做过什么，是否选中一个对象？是否向打印机发送了文件？拾取了颜料桶工具吗？等等。

这些信息很多是以被动方式出现的，也就是说，用户并未得到关于这些操作的直接提示，而是作为附件提示。例如，当用户从"File"菜单中选取"Print"时，一个带有多种用户选项信息的打印对话框出现了，由于此对话框的出现，用户就已知道他选择了"Print"项，他们可能只是下意识地注意到了这点，因为通过对话框的出现，他们对所做的事已收到了反馈信息。

然而，假设用户选择了"Print"菜单项后对话框没有出现，文件被直接送往打印机却没有任何反应，这时用户可能会感到困惑，怎么什么也没发生。当然，如果从打印机中打出了一页纸，用户会认识到打印文档的工作实际上已经完成。但如果打印机是连接在另一个房间里（这在许多办公室中很常见），用户由于没见到任何反应而会不停地选择"Print"，结果文档会被打印很多份而造成浪费。

缺乏反馈信息可能是个大问题。当任何反应都没有时，用户可能会徒劳地将一系列动作重复多次而试图看到某种反应。当然，实际上已有动作发生，只是用户没有看见而已。

这就是为什么当用户单击了工具条上的一个按钮后，工具条上其图象就会嵌入的原因。这就是视觉反馈，告诉用户此按钮已被选中。如果无此信息，用户就会坐在那里不停地按下按钮而看不到任何反应。负责绘制按钮的各种形态的代码就是为了给用户一个反馈信息，我们可以很容易地写出程序，它使按钮在按下时没有任何变化而动作已经执行。不要跟自己开玩笑，尽管这个代码只是提供视觉反馈，但它在界面中是非常重要的部分。为什么呢？因为它是以不知不觉的方式工作，而用户并不注意界面本身，只注意产品是否按他们所想的方式工作。

用户很快就会熟悉这种"辅助的"反馈层，也就是说，单击一项后就能看出来该项被单击，选中某个菜单项（如 Print）后就能看到询问用户更多信息的对话框。正如上面提到的，这种辅助层很重要，因为它与用户的期望相符。这就是说，当用户在屏幕上做任何事时，就希望屏幕上会有反应，而屏幕上的控件外形有了变化，就说明用户已执行了某种操作。

2.6 如何完成某工作

对用户来说，第二个重要信息是他们如何完成某工作的。在上面的打印例子中，他们选择了“Print”菜单项。但如果在某些界面设计组中有人认为使用“Print”这个词不够精确，而是使用了“Output”这一词，情况会怎样呢？

不要笑——我们常常遇到这种情况。我们不得不委婉地说服这个用户界面的设计者，让他认识到“Output”这个词会妨碍用户使用软件，尽管用户可能会很不顺利地将文件打印出来。但在一段时间内，他们仍不能顺利地重复这些动作。

为什么呢？因为运行在 Windows 下的其他程序都使用“Print”一词。也许这不是最好的词，但它是用户已经知道（记得“一致性”这第一界面要素吗？）并且所期望出现的词，把它换成别的词后，此软件产品就难以学习了（我们最终会把它改回去，但不是在用户使劲寻找“Print”这一选项之前）。

这里的道理很简单，就是为了使用户易于学习。换句话说，一套连续一致的动作总会生成同样结果，而每一次都有同样的反馈信息。他们单击了打印机图标吗？他们选中了“Print”菜单项吗？如果已经做了，确认完成该任务的路径是清晰的。

2.7 已发生了什么

第三个同等重要的问题是用户的操作会有什么结果。在与“打印”类似的操作中，结果相当明显。首先，用户选择“Print”，然后填好对话框（即使没填什么，只是单击 OK 按钮），最后他们的文档就被打印出来。可是，当过程更短暂时（如他们尚未完成打印），这时，使操作结果在屏幕上提供正确的反馈信息是必要的。

在绘画程序 ICE/Works 中，当用户选中了新的绘画工具时，应当改变光标的形状。比如选择了铅笔工具，光标就变成铅笔；选择了颜料桶工具，光标就变成颜料桶工具，并流出颜料；如果选择了喷涂工具，光标就变成了喷涂罐，颜料从喷嘴中流出。一旦用户选择了某工具，光标立即改变形状。

为了弄清楚这些工作方式，让我们总结一下这三种用户界面组件，并看一看它们在 ICE/Works 中与改变工具相关的实际情形。我们以从颜料桶工具到铅笔工具的转换为例（参见图 2-3 至图 2-5）。

1. 用户已做了什么（用户采取的动作）。用户在光标位于铅笔工具按钮上时按下了鼠标左按钮，此时铅笔工具按钮嵌入，当用户在铅笔工具上松开鼠标按钮之前，并没有选中任何工具。
2. 用户是怎样做的（用户完成此动作时的步骤）？用户不得不在铅笔工具上单击，然后松开鼠标按钮以选中该工具。如果用户只是在铅笔工具上按下了鼠标，而将鼠标移到别的地方却没有松开按钮，那么铅笔工具就没被选中（这一特征在设计此类的按钮工具时易被忽视）。用户可以很快学会这些激活工具的必要步骤——在工具上按下鼠标按钮并松开，用户也可在按下鼠标按钮后将鼠标移开以放弃所做的选择。

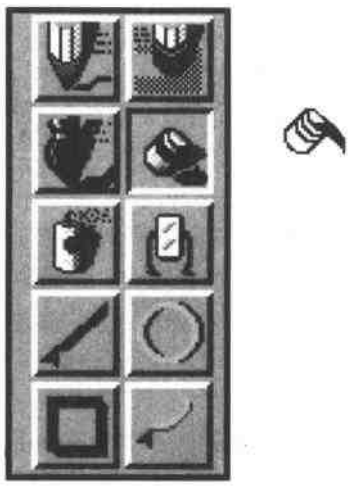

图 2-3 用户从颜料桶工具转向铅笔工具，颜料桶工具开始嵌入，这表明了现在已被选中，光标也成为颜料桶形状

图 2-4 用户正在选择铅笔工具时，该工具按钮嵌入，虽然这个工具按钮也嵌入，但只有用户松开按钮时，该项才真正被选中。而在这之前虽然两按钮都被嵌入，但颜料桶工具仍是活动的工具

3. 用户动作的结果是什么(发生了什么)？当用户已在铅笔工具上按下并放开鼠标按钮时，该工具按钮就被嵌入以指明它被选中。颜料桶工具按钮就升起以表示它已不被选中，光标也变为铅笔形状。

为了增加软件产品的响应能力，所有的反馈信息都是重要的，然而这只是一种反馈。另一种反馈以一种可见的提示去告诉用户在程序中当前所处的位置，这种反馈与用户界

面的第二个要素　　帮助性有关，它引导用户沿正确的路径工作。

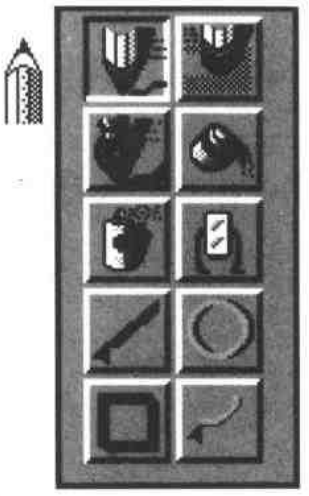

图 2-5　用户在铅笔工具上松开了鼠标按钮——此时该工具嵌入表示被选中，而颜料桶工具复原，表明它已经关闭，同时光标变成铅笔形状，进一步表明了现在用的是何种绘画方式

这种反馈不是暂时的，它本质上更具有组织性，我们的意思是第一种反馈发生在给定时间，当用户采取某个特定动作时出现；第二种反馈给出了用户正在进行着的操作的信息。

例如，在 ICE/Works 中，每个绘图工具都有它自己的光标来显示用户当前所在的方式。正如前面提到的，铅笔工具有它自己的光标——一个铅笔形状的光标指出用户是处在铅笔方式。只要用户不改变所用工具，光标就保持铅笔形状。这种可视的反馈机理的重要性就在于正好出现在用户注意力集中的那点——因为用户总是紧跟光标（因为光标是所要绘图的地方），所以用户可时刻知道他们采用的是何种工具。

另一个可视反馈的例子是打开“镜象方式”。镜象是一种绘画工具，它允许用户观看所画图象对于一个轴的映象。因为镜象可用于所有绘图操作中，所以能够在绘图时区分镜象方式还是非镜象（正常）方式就非常重要，我们是通过在当前绘图操作的光标旁加一个小镜子来实现的。因而，如果用户使用铅笔工具并且镜象方式为打开状态，则光标中铅笔旁边就出现一个小镜子。如果镜象方式为关闭状态，光标旁就不会出现镜子，这一特性体现在所有绘图工具中（见图 2-6）。

这一可视性提示使用户一望而知下一步的绘画中是否打开了镜象工具。他们不需要在屏幕上别的地方去寻找这一信息（以至于分散注意力），就会立刻知道将怎样做。

渴求信息：用户渴求得到有关程序运行中的反馈信息，请大胆地提供给他们。

使用第二种反馈信息的方式显然不同于第一种，它们各自用于不同的对象。然而，这