第一部分

Visual C++环境

第一章　Visual C++简介

尽管可以发现创建其他种类的应用程序的优秀工具，但 Visual C++确实为 Windows 应用程序的编写提供了一个极好的新环境。本章就一些广泛关注的问题介绍一下 Visual C++。对 Visual C++有一定了解的读者，可以略过这一章。

1.1　Visual C++是什么

Visual C++是开发应用程序——尤其是运行在 Microsoft Windows 下的应用程序的完整开发系统。Windows 的优点就是使终端用户的工作显得直观而有趣。但在 Windws 下带有消息驱动体系和精心设计的 API 的环境图形性质，长久以来使得最终用户使用 Windows 很容易而对编程者来说却又很麻烦。

Microsoft 简化 Windows 程序设计的第一个有创新的工具是 Visual Basic，它是一个类似于解释器的编译程序，它在后台处理 Windows 的调用。尽管 Visual Basic 对许多程序设计人员来说是一个理想的结果，但它需要程序设计人员有高水平的机型，这就增加了一些限制（更不必说系统开销），想充分利用 Windows 功能的程序员仍然要在 C 或 C++上进行程序设计。

Visual C++给由 Visual Basic 转入 C++的程序设计人员带来了一些方便，因为它充分利用了 Windows 的所有功能。Visual C++在 Microsoft Foundation Class Library 2.0 版本下是一个应用程序框架，它被设计成 Visual Basic 那样。在程序员写程序前，它就创建了它所能创建的应用程序代码。然后，Visual C++指引你到你可能要增加响应消息的代码的正确位置，当然，程序可以同编程人员所需的那样复杂，许多很普遍的用户界面特征在应用程序中的设计超出要求，你可以通过增加代码来定制它们。

综上所述，Visual C++是一个直观的编程工具。当需要创建特征图形时（如设计一个位图或菜单），这一特点将变得很有意义。程序设计人员可快速获得与之紧密结合的图形编辑软件，然后可轻而易举地把那些图形资源加入程序中。

1.2　为什么 C++与 Windows 联系紧密

用 C++程序设计语言编写 Windows 应用程序是一个很自然的选择，当用 C 来进行程序设计时，即使是最简单的 Windows 应用程序也需要设置大量的参数。而在 Visual C++中，有一个类库提供了适当的缺省值和消息处理程序，程序员可从该类库中继承所有缺省属性，然后对所需定制的属性作更进一步处理。

用 C 进行程序设计的开发人员，在第一次接触 C++时，可能会有这样的疑问：是不是必须从头学一门新语言，但事实上，只需要学习不多的 C++语法就可以开始用应用程序框架来编程。相对于 C 语言，C++语言扩充了论文强有力的功能，但在空闲的时间里，你就可

以学会其中的绝大多数。

除了支持 Windows 编程外，Microsoft Foundation Class Library 还提供了许多类，这些类对于在另一环境下编程非常有用，对 Windows 编程也同样有用。这些类包括字符串、文件和集合类。

1.3 怎样开始学习

同 Windows 编程一样，它的困难有时是过分强调了。学习 Visual C++编程的最佳办法是从头开始创建简单的 Windows 程序。本书的第一部分提供了这一方法的大致过程，并显示出怎样创建框架应用程序架。第二部分教你进一步深入并开始编写简单的应用程序，以说明一般编程技巧。第三部分描述 Visual C++1.5 的新特征。

第二章　Visual C++运行环境概况

Visual C++运行环境有一些需了解的关键部分，以便我们开展工作。本章提供了Visual C++运行环境的高度概括，展示了它的每个部分是怎样结合在一起建立一个应用程序的。

2.1　运行环境的组成

在最高程度上，Visual C++运行环境由四部分组成：

(1) Visual Workbench(主要工作站)

(2) App Studio(资源编辑)

(3) Microsoft CodeView 和实用程序

(4) On-Line Help(联机帮助)

尽管Visual C++有许多特殊的特征，但绝大多数时间是使用运行环境中的两个主要部分与Visual C++发生作用，这两个主要部分是Visual Workbench 和 App Studio。事实上，通过界面的这两个成分便可做所有的工作了。

2.1.1　Visual Workbench

VisualWorkbench 最基本的功能是编辑代码，但它的功能远远超过了文本编辑程序，Visual Workbench 控制建立应用程序的进程，并允许运行、调试和分析代码的结构。除了编辑资源外，程序员可能要在 Visual Workbench 上花费绝大多数甚至所有的时间。

Visual Workbench 用滚动条重叠窗口，并充分利用 Windows 界面，以便于指定工程文件，建立和调试选择项(见图 2.1)。使用语法配色，可以定制、显示关键字、注释和定义不同颜色。

除标准编辑、建立和错误报告外，Visual Workbench 还包括以下主要功能：

(1) AppWizard，创建框架应用程序的工具。

(2) ClassWizard，为任何 Windows 类浏览 Windows 信息的工具，并添加和编辑消息处理程序。

(3) Browser，用来检测函数、类和符号间的相关性。

(4) 集成调试器。

AppWizard 消除了建立标准应用程序骨架的冗长工作，即那些需要在应用程序中描述出来以正确创建窗口的语句。同时，AppWizard 还提供了相当的灵活性，在程序员写代码之前，就可指定应用程序的种类。

ClassWizard 是一个浏览和编辑 Windows 应用程序消息处理例程的理想工具，当应用程序越变越长时，ClassWizard 的作用更加明显。用户选择一个对象或消息，ClassWizard 直接转

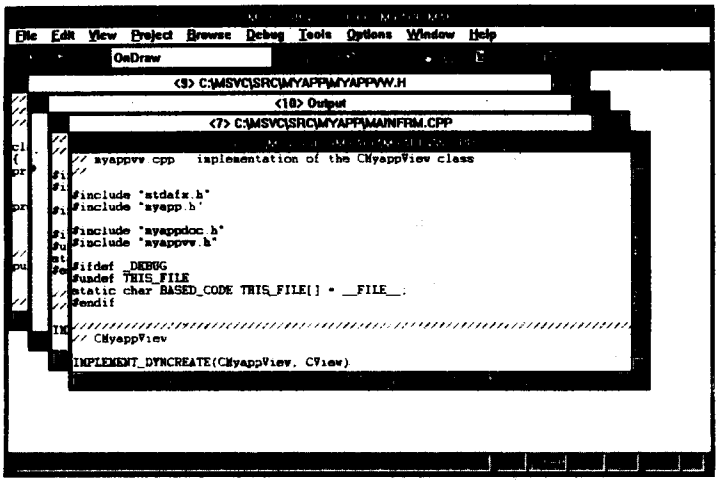


图 2.1 Visual Workbench

到代码的恰当位置，并在需要时创建一个新的消息处理程序。

注意 AppWizard 和 ClassWizard 都假定使用了 Visual C++中的 C++和 Microsoft Foundation Class Library 2.0 版。

第五章描述了怎样使用 AppWizard。第二部分举例描述了 ClassWizard 的常见使用方法。

2.1.2 App Studio

App Studio 使程序员能为 Windows 程序创建和编辑资源，例如：快捷键，位图，对话框，图标，菜单和字符串表，见图 2.2。

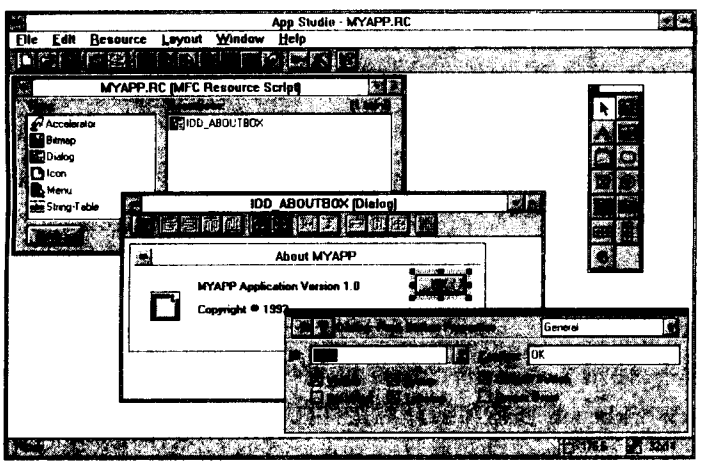


图 2.2 AppStudio

程序员可以使用 App Studio 中的图形工具创建所需的所有资源，AppWizard 创建一个资源并把它加入工程文件，当程序员在 Workbench 中操作 App Studio 时，这个工程文件资源一开始便在缺省情况下被打开。

App Studio 与 Visual Workbench 在其他方面也同样有巧妙的结合，例如，使用 App Studio 内部的 ClassWizard，可以创建支持对话框所需的类。ClassWizard 引导程序员到 Visual Workbench 去编辑代码。

App Studio 能打开 EXE 和 DLL 文本文件，也能打开资源文件，使用 App Studio 的这一功能可使程序员获得一个 Windows 可执行文件的所有资源。程序员可以编辑这些原代码或拷贝它以用作新工程文件的代码(当然，App Studio 并不自动地给这些拷贝文件授与权限——仅在技术功能上可以达到这个要求!)。

2.1.3 CodeView 和实用程序

在断点，观察窗口，调用栈，甚至一个寄存器窗口和混合的资源/汇编方式这类可能需要调试的时候，Visual Workbench 均提供了调试功能。但对于特殊的调试情况，Visual C++的 Professional Edition 还包括 Microsoft CodeView 调试程序，它有一套更扩充的命令(尽管它是面向字符的)。

Visual C++包括一些来自 Windows Software Development Kit(SDK)的实用程序，以检查 Windows 信息和程序存储。另外，这一链接程序的 32 位版本能与打印任选项一起运行，并能以可读形式打印出下一个 EXE 和 OBJ 文件的整个内容。Visual Workbench 在建立应用程序时调用运行在普通方式下的链接程序。但程序员亦可在需要时以命令行方式使用它，Visual Workbench 还调用 NMAKE 来创建工程文件，程序员可在命令行调用 NMAKE 来创建非标准的 MAK 文件。

2.1.4 On-Line HELP

Visual Workbench 和 App Studio 通过菜单命令和 F1 键与 Help 结合。所以可以把 Help 看作图形环境的一部分，还可独立地运行 Help。Help 包含以下每一部分的完整参考信息。C 和 C++语言，运行期的库和类库。浏览一下 Help 是学习有关 Visual C++更多知识的好方法。

2.2 Visual C++是如何处理文件的

这一部分假设我们已获得了创建 Windows 程序时 Visual C++必须提供的所有支持(包括类库和自动创建功能)。

在最简单的情况下，使用 Visual C++编程包含三个步骤：

(1) 使用 Visual Workbench 创建和编译源文件(CPP，H)。

(2) 使用 App Studio 创建和编译资源(RC 和图形文件)。

(3) 在 Visual Workbench 内部建立程序。

实际上，步骤 1 和 2 并不一定按次序去做，在程序完成前源文件和资源间前后移动是很普遍的，然而，这三个步骤给出了 Visual C++应用程序各部分结合在一起的一张大图像。

图 2.3 显示了源代码编译、资源编译和所建应用程序各部分的连接。当然，程序员一次可只选择应用程序的一部分进行编译。图 2.3 的另一个简化是只显示了使用 Visual C++时程序员直接可见的文件。接下来的两部分既描述可见的文件(通常是源文件)，又描述程序员不需考虑的支持文件。

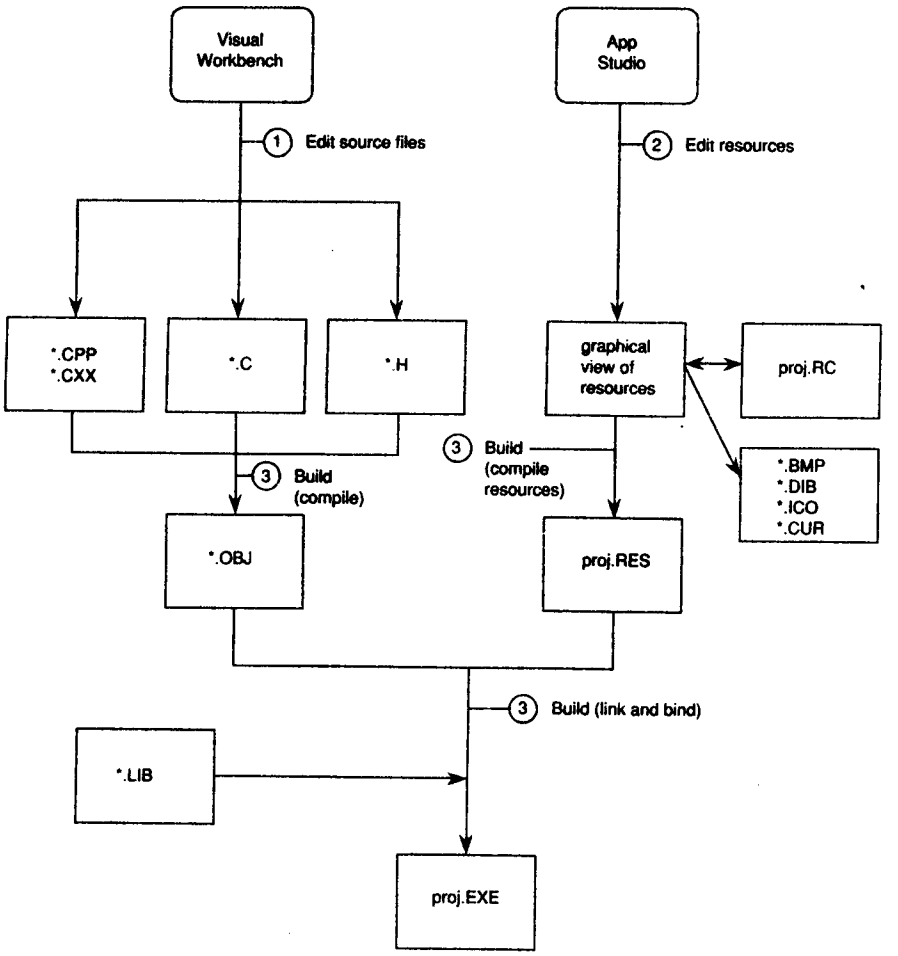


图 2.3 Visual C++如何处理文件

2.2.1 Visual Workbench 源文件

程序员可以使用 Visual Workbench 为应用程序编辑所有的 C 和 C++源代码，包括以下文件：

(1) CPP 文件(也可称为 CXX)。这些文件包含 C++源代码。Microsoft 的 C/C++编译器从文件扩展名知道编译哪一种语言，App Wizard 也创建一些 CPP 文件。

(2) C 文件。这些文件包含 C 源代码，尽管 App Wizard 不创建 C 文件，也可以把 C 文件加入工程文件中。当然，程序员必须注意 C/C++的相容项。

(3) H 文件。这些文件包含类说明,类型定义语句和#define 语句,通过使用#include 命令,H 文件作为主文件包含在应用程序中,App Wizard 也创建部分 H 文件。

(4) 当编辑工程文件时,H 文件并不在工程文件列表框的文件中列出。这是因为 Visual Workbench 通过直接检查源文件的#include 语句得出了与 H 文件的相关性。

在成功编译之后,C/C++编译器为工程文件中的每一个 CPP 和 C 文件创建一个 OBJ 文件。

2.2.2 App Studio 源文件

App Studio 创建的资源不同于编辑源代码,只要正确的 RC 文件是打开的,程序员可以编辑资源而完全不必考虑源文件是否包含于其中。

一个工程文件的所有资源通常与一个 RC 文件相联系(尽管程序员可以采用更多的 RC 文件),App Studio 把为 RC 文件编辑的资源看作一个整体。程序员可以专注于资源的图形范围,而让 Visual C++自己跟踪涉及到的文件。

资源文件包含下列类型:

(1) 工程 RC 文件。这是资源脚本,它包含所有描述菜单和对话框结构的信息,同时也包括需要编译的图形文件。

(2) BMP 文件。这些文件包含位图,参见跟在图形文件后面的注意事项。

(3) DIB 文件。这些文件包含设置独立位图。

(4) ICO 文件。这些文件包含图标。

(5) CUR 文件。这些文件包含光标。

注意 缺省状态下,App Studio 把所有的图形文件(位图、图标、光标)放入一个 RES 子目录中,不同于资源脚本(RC)文件的是,图形文件是二进制的,而不是 ASCII 文本。在编辑工程文件时,图形文件不在工程文件列表框文件中出现,因为 Visual Workbench 通过检查 RC 文件而得出相关性。

还可以使用分别包含 Windows 和 Visual Basic 控制的 DLL 和 VBX 文件。程序员可以把这些控制加入对话框。然而,DLL 文件和 VBX 文件不是工程文件。RC 文件在对话框脚本中指明它们,但 DLL 和 VBX 文件自身并不被编译入应用程序。相反,它们在应用程序运行时必须作为动态的链接库被列出。

RC 文件和图形文件输入资源编译器,创建一个 RES(资源)文件。在工程文件链接后,资源编译器再次运行,并把 RES 文件连接到可执行文件上。但这一步通常对 Visual C++程序员来说是显而易见的。

2.2.3 工程控制文件

另有一些文件在绝大多数时间对 Visual C++程序员来说几乎是不可见的。但对于工程文件却非常重要,下面简洁地描述了每个文件的目的。

(1) PROJECT.BSC。浏览数据库文件。它使 Visual Workbench 能够浏览符号、类和函数相关性,参见 Module.SBR。

(2) PROJECT.DEF。Windows 模块定义文件。在编辑工程文件时,它是这儿所列文件中唯一需要在工程文件中列出的文件。

(3) PROJECT.MAK。这一文件使用NMAKE公用程序接受的语法描述了所有的建立选择项和建立步骤。无论何时编辑工程文件或改变建立选择项,Visual Workbench都可通过修改这个文件来反映这一变化。

(4) STDAFX.PCH。为Foundation Class Library预编译首部。通过在第一次建立工程文件时创建这一文件,Visual C++存储了在后面的建立过程中的有效次数。

(5) PROJECT.PDB。调试信息文件。

(6) MODULE.SBR。包含原模块浏览信息的文件。BSCMAKE从工程文件所有的SBR文件建立一个数据库文件(BSC)。除非浏览支持在工程文件选择项中被关闭,否则Visual Workbench在建立工程文件期间自动地引用BSCMAKE。

(7) PROJECT.WSP。工作区文件,包含Visual Workbench环境设置,这些设置包含窗口安排和语句输入时的色彩。

(8) PROJECT.VCW。这一文件存储工程文件的当前状态,例如Visual C++版本和断点数。

DEF,MAK和VCW文件是ASCII格式,可以发现这对于检测是很有用的,但在对它们作彻底修改时要小心一些。

注意 除PROJECT.MAK文件之外,如果在这部分所列的任何一种文件被删除或破坏,它们都可以通过强行做一个完整的建立(工程文件菜单中的Rebuild All)来重新创建。SBR、BSC和PDB的创建可以通过关闭工程文件可选项中的浏览和调试支持隐含起来。

最后,在正确的LIB和DLL文件的支持下,一个Visual C++应用程序便建成了。这些LIB,DLL文件通常不与工程文件在同一目录下,因而不被认为是工程文件的一部分。

第三章　工具条和控制调色板要点

作为一个图形环境，Visual C++在任何可能的地方都使用工具条。App Studio 也包括一个为编辑对话框和图形色彩的控制调色板。本章摘录了这些特征，以便于读者能尽快理解它们。

3.1　Visual Workbench 工具条

在 Visual Workbench 中工作，Visual Workbench 工具条始终可见。除非 View 菜单上的工具条可选项被关闭(见图 3.1)。选择这一可选项可设定工具条状态是可见的还是不可见的。表 3.1 列出了工具条中的按钮。

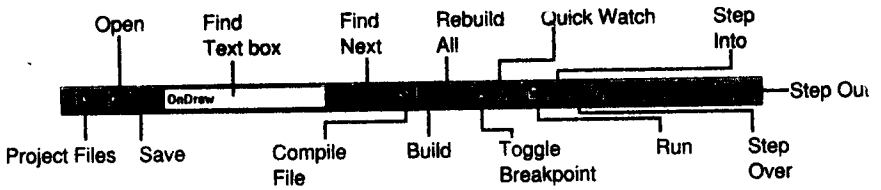


图 3.1　Visual Workbench 工具条

表 3.1　Visual Workbench 工具条按钮

	Project Files(工程文件文件)	列出当前工程文件的所有文件
	Open(打开)	打开一个源文件
	Save(存储)	存储当前活动窗口的源文件
OnDraw	Find TextBox(查找文本框)	执行 Find 命令：敲入需找的字符，按 Enter 键
	Find Next(查找下一个)	执行 Find 命令，查找上次指定的查找字符
	Compile File(编译文件)	编译活动窗口的源文件
	Build(建立)	只建立过期文件

（续表 3.1）

	Rebuild All(重新建立)	重建所有的目标文件
	Toggle BreakPoint(断点触发器)	在当前行设置断点或在断点处消除断点
	Quick Watch(快速查看)	显示对话框以更改 Quick Watch 窗口
	Run(运行)	在装入调试信息后运行应用程序(直接运行应用程序按 Ctrl+F5)
	Step Into	执行一行代码，如果当前指令是一个函数调用则进入该函数
	Step Over	执行当前函数的一行
	Step Out	执行至当前函数的末尾

3.2　App Studio 主工具条

当在 App Studio 中工作时，App Studio 主工具条总是可见的，活动资源例外(见图3.2)。工具条的可见性可通过 Windows 菜单进行开关切换。表 3.2 显示了工具条按钮。

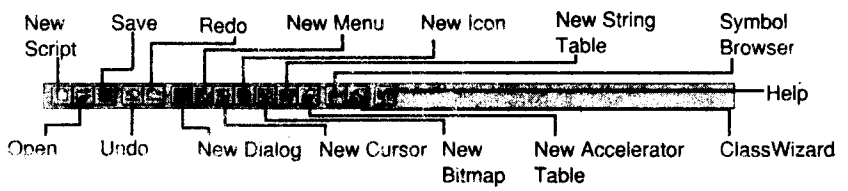


图 3.2　App Studio 主工具条按钮

表 3.2　App Studio 主工具条按钮

	New Script	用一个空的 RC 文件开始编辑
	Open	打开一个已存在的资源脚本(RC)或其他资源文件
	Save	存储 RC 文件或活动资源
	Undo	取消上次编辑，缺省情况下，最多可取消 10 个操作
	Redo	重新记录上次被取消的操作

（续表 3.2）

	New Dialog	创建新的对话框
	New Menu	创建新的菜单条
	New Cursor	创建新的光标
	New Icon	创建新的图标
	New Bitmap	创建新的位图
	New String Table	创建新的字符串表
	New Accelerator Table	创建新的加速键表
	Symbol Browser	显示 IDs 和它们的相关资源
	ClassWizard	打开 ClassWizard 对话框
	Help	打开联机 Help 窗口

3.3 App Studio 的 Dialog Editor 工具条

Dialog Editor 工具条在编辑对话框时出现(如图 3.3),表 3.3 列出了工具条按钮。

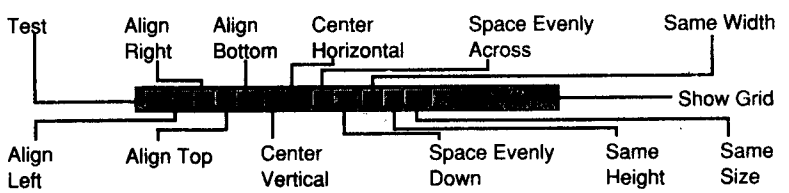


图 3.3 App Studio Dialog Editor 工具条

表 3.3 App Studio Dialog Editor 工具条的按钮

	Test	在测试方式中运行对话框
	Align Left	调整一组中的控制左边缘

（续表 3.3）

	Align Right	调整一组中的控制右边缘
	Align Top	调整一组中的控制顶端
	Align Bottom	调整一组中的控制的末端
	Center Vertical	移动控制到对话框的横向中心
	Center Horizontal	移动控制到对话框的纵向中心
	Space Evenly Across	一组中有三个或更多的控件，则安排这些控制使它们左右空格均匀
	Space Evenly Down	若一组中有三个或更多的控制，则安排这些控件，使它们上下之间空格均匀
	Same Width	使一组中的所有控制具有相同的宽度
	Same Height	使一组中的所有控制具有相同的高度
	Same Size	使一组中的所有控制具有相同的尺度
	Show Grid	触发网格的开或关

3.4　控制调色板(App Studio 对话框编辑器)

缺省情况下，控制调色板在编辑对话框时出现。这一调色板用来在对话框中画新的控制（见图 3.4）。如果它没有出现，可通过按 F2 来显示它。表 3.4 列出了调色板的按钮。

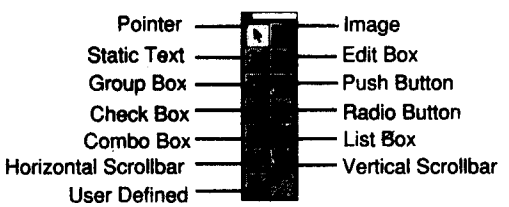


图 3.4　App Studio 控制调色板

表 3.4 App Studio 控制调色板按钮

	Pointer	选择一个控制或标出一组控制
	Image	包含一个图像
	Static Text	包含用户不能编辑的文本
	Edit Box	包含用户能编辑的文本，支持一些内部编辑特征
	Group Box	用来作为环绕一组控制(如单选按钮)的页框
	Push Button	通过单击鼠标激活的按钮，用户单击它们选择一个命令(如：OK 和 Cancel)
	Check Box	在开或关状态间触发一个条件
	Radio Button	提供选择，用户在运行时只能设置一个开关按钮
	Cornbo Box	作为一个简单的组合框，下接组合框，或下拉列表框，这依赖于怎样设置结构特征
	List Box	显示一个简单的(非下拉式)列表
	Horizontal Scrollbar	显示一个用户可以滚动的纵向条
	Vertical Scrollbar	显示一个用户可以滚动的横向条
	User Defined	选择地包含一个安装类似于 DLL 或目标文件的控制

3.5 App Studio 的 Properties 窗口工具条

Properties 工具条在编辑 App Studio 中任何一种资源时都是可见的(见图 3.5)。按 Shift+F2 可使其可见(还可双击一个目标以使 Properties 窗口可见，除非双击被保留用作其他动作)。表 3.5 列出了该工具条的按钮。

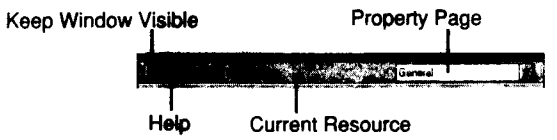
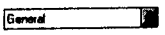


图 3.5　App Studio 的 Properties 工具条

表 3.5　App Studio 的 Properties 工具条按钮

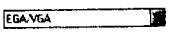
	Keep Window Visible	使 Properties 窗口始终可见(在可能的情况下)
	Help	显示当前特征的分组(页)的帮助
	Current Resource	显示当前资源
	Poperties Page	选择显示哪一个特征分组(页)

3.6　App Studio 的 Icon Editor 工具条

Icon Editor 工具条在编辑图标时出现(见图 3.6)。表 3.6 列出了这些按钮。

图 3.6　Icon Editor 工具条

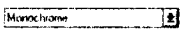
表 3.6　Icon Editor 工具条按钮

	Select Device	选择一个显示图标的装置(VGA,CGA 等等),必须为每个装置编辑一个不同的图像
	New Device	为还没有为图标编辑图像的装置创建图像

3.7　App Studio 的 Cursor Editor 工具条

Cursor Editor 工具条在编辑光标时出现(见图 3.7),表 3.7 列出了这些按钮。

表 3.7　Cursor Editor 工具条按钮

	Select Device	选择一个显示图像的装置(VGA,CGA 等等),必须为每个装置编辑一个不同的图像
	New Device	为还没有为光标编辑图像的装置创建图像

（续表 3.7）

Hotspot: 0,0	Current Hotspot	显示 Windows 用来跟踪位置的点的内部坐标
	Set Hotspot	设置用来跟踪位置的点

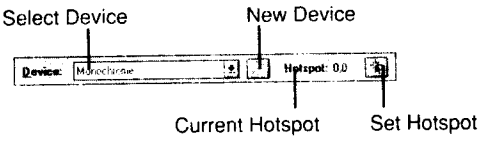


图 3.7 Cursor Editor 工具条

3.8 App Studio 的图形调色板

缺省情况下，这一控制调色板在编辑任何一种图像（光标、位图或图标）时都出现，如果看不见它，可以通过按 F2 来显示它。图 3.8 显示了这一调色板，同时表 3.8 列出了在调色板上的按钮。

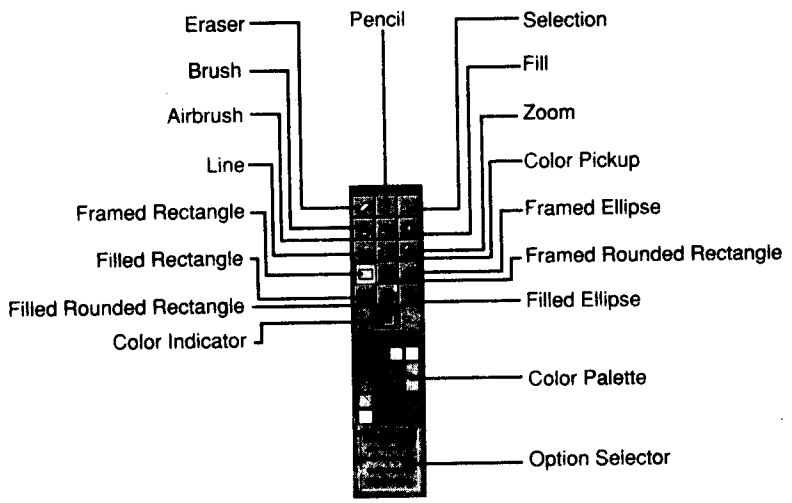


图 3.8 App Studio 的图形调色板

表 3.8 App Studio 图形调色板按钮

	Eraser	用背景色刷新图像
	Pencil	用一个像素的宽度徒手画线

（续表 3.8）

	Selection	选择一个长方形的范围
	Brush	按照通过 Option Selector 设置的形状和尺寸进行刷新
	Airbrush	在以刷子为中心的周围任意地抹颜色
	Fill	用前景或背景颜色填充被环绕的区域，它依赖于鼠标按钮的使用
	Line	用 Option Selector 设定的宽度画直线
	Color Pickup	给选中的像素设置前景颜色（左按钮）或背景颜色（右按钮）
	Zoom	显示被选区域的扩大视图。用两次使效果相反
	Framed Rectangle	创建空的长方形
	Framed Rounded Rectangle	创建空的圆角长方形
	Framed Ellipse	创建空的椭圆形
	Filled Rectangle	创建填满的长方形
	Filled Rounded Rectangle	创建填满的圆角长方形
	Filled Ellipse	创建填满的椭圆形
	Color Indicator	显示当前的前景或背景颜色
	Color Palette	依靠鼠标按钮单击某一块色彩，从而选择前景或背景颜色
	Option Selector	为被选中的工具选择宽度和格式