第24章　MIAW及其相关内容

Movie in a Window(窗口中的影片)，通常称为MIAW，是Director里的一种不寻常的成份。它使我们能够在舞台之外打开一个含有Director影片的窗口。

MIAW有多种用途。由于它们是相互独立的，并且也与舞台相互独立，它们适合于那些不应当出现在舞台上的功能。用 MIAW可以创建自定义的对话框或信息窗口。我们甚至可以创建像Director或其他许多专业软件那样的使用多个不同窗口的软件。

24.1　使用MIAW

MIAW的Lingo的难易程度各不相同。如果想要打开一个窗口，以显示一些信息，只需要几行代码。而另一方面，要进一步运用 MIAW，需要使用一整套现有的命令、函数、属性和特殊事件处理程序。无论是哪种情况，首先需要的是创建 MIAW。下面这节就讲述这个内容。

24.1.1　创建MIAW

要使用MIAW，首先要有另一个影片文件。如果所需要的是一个较小的窗口，应当相应地设置那部影片的舞台尺寸。把那个文件存储为 miaw.dir，再打开另外一个新影片。我们实际上可以用消息窗口显示这个MIAW。

```
miaw = window("Test MIAW")
miaw.filename = "miaw.dir"
miaw.visible = TRUE
```

一个新窗口出现在舞台上方。它的标题应当是“ Test MIAW ”。所创建的全局变量MIAW包含对那个窗口的引用。可以得到它的值：

```
put miaw
-- (window "Test MIAW")
```

用这个全局变量或用window("Text MIAW")结构都可以引用这个窗口。要关闭或删除这个MIAW，也可以在消息窗口里进行：

```
close(miaw)
forget(miaw)
```

打开MIAW的处理程序与刚才在消息窗口里输入的信息相似。下面就是这个处理程序以及关闭MIAW的处理程序：

```
on startMIAW
  global gTestMIAW
  gTestMIAW = window("Test MIAW")
  gTestMIAW.filename = "miaw.dir"
  gTestMIAW.visible = TRUE
end

on endMIAW
  global gTestMIAW
  close(gTestMIAW)
```

```
  forget(gTestMIAW)
end
```

可以注意到关闭 MIAW需要使用两个命令。第一个命令 close实际上只能使该窗口不可见；第二个命令forget则把它从内存里清除。如果只使用第一个命令， MIAW仍旧存在于内存里，并当我们打算使用其他MIAW时可能会导致错误的发生。

24.1.2 MIAW属性

前面的简单例子里使用了 MIAW的一些基本属性，即 filename和visible属性。下面列出MIAW的全部属性：

■ visible —— 确定显示或隐藏窗口。隐藏的窗口也能执行 Lingo程序。

■ filename —— 与用于MIAW的Director影片相对应的文件名。如果计算机正在上网，也可以使用某个网址。

■ name —— MIAW的名称。这是窗口的缺省标题，并在 window结构里用来指代这个窗口。

■ title —— 用在窗口的标题条里以代替 name属性。

■ titleVisible —— 当MIAW的类型为-1时，该属性确定MIAW是否显示标题条。

■ windowType —— 可以把该属性设置为-1、0、1、2、3、4、5、8、12、16或49。表24-1和表24-2出了这些类型的含义。

■ drawRect —— 这个功能强大的属性可以用来缩放 MIAW，包括其中的位图。

■ rect —— 这个属性使我们能够剪裁或扩大 MIAW，但不发生缩放。

■ sourceRect —— 返回drawRect或rect变化之前的MIAW的原始坐标。

■ modal —— 当某个窗口的这个属性被设置为 TRUE时，这个窗口将拦截所有的输入信息，阻止包括舞台在内的其他窗口接收鼠标或按键信息，直至关闭该窗口或该窗口的 modal被设置为FALSE时为止。

表24-1 Windows的MIAW类型

类型编号	描述	可移动	关闭按钮	最大化按钮	最小化按钮
-1	缺省	是	有	无	无
0	标准	是	有	无	无
1	警告框	否	无	无	否
2	矩形	否	无	无	无
3	矩形	否	无	无	无
4	文件	是	有	无	无
5	文件	是	有	无	无
8	文件	是	有	有	无
12	文件	是	有	有	无
16	文件	是	有	无	无
49	面板(放映机里的除外)	是	有	无	无

尽管表24-1和表24-2给出了每种窗口类型的外观特征，但窗口的具体情况还受 Director版本和操作系统版本的影响。 MIAW在Mac OS 8.1与Mac OS 8.5里的外观互不相同，在Windows 95与Windows 98里的外观也互不相同。

表24-2 Mac的MIAW类型

类型编号	描述	可移动	关闭按钮	拉伸按钮	重新设置按钮
-1	缺省	是	有	有	无
0	标准	是	有	有	无
1	警告框	否	无	无	无
2	矩形	否	无	无	无
3	带阴影的矩形	否	无	无	无
4	文件	是	有	无	无
5	文件	是	无	无	无
8	文件	是	有	有	有
12	文件	是	有	无	有
16	曲线边框	是	有	无	无
49	面板(放映机里的除外)	是	有	无	无

24.1.3 窗口命令

open、close和forget是MIAW所使用的主要命令。不过，当多个 MIAW同时打开时，还需要使用另外两个命令。还有一些命令能使 MIAW与舞台相互对话。重要的窗口命令有：

■ open —— 创建一个新MIAW，并返回引用它的变量。

■ close —— 隐藏一个MIAW，其实它仍旧存在。

■ forget —— 把MIAW从内存里清除。

■ moveToFront —— 把MIAW变成最顶层的窗口。

■ moveToBack —— 把MIAW变成最底层的窗口。

■ tell —— 向MIAW发送一个Lingo命令或处理程序的一次调用。它也可以用来向 MIAW发送一组Lingo命令。

tell是MIAW用来与舞台交换信息的命令。可以像这样用它来发送一个命令：

```
tell window("Test MIAW") to myHandler
```

也可以向MIAW发送一组命令：

```
tell window("Test MIAW")
  myHandler
  myOtherHandler
  go to frame "x"
end tell
```

24.1.4 MIAW的系统属性

除了上述的窗口属性外，还有一些与窗口相关的系统属性。下面列出了全部系统属性。它们可以告诉我们哪些窗口存在、哪些窗口处于激活状态以及哪个窗口位于最顶层。

■ the windowList —— 返回一个列表，其中包含当前全部 MIAW，包括不可见的MIAW。如果当前没有窗口，则返回一个空列表。

■ the activeWindow —— 返回当前处于激活状态的窗口的引用变量。如果舞台是当前处于激活状态的窗口，则返回(the stage)。

■ frontWindow —— 返回处于最顶层的的窗口的引用变量。如果是舞台，则返回 (the stage)。

■ windowPresent() —— 当为该函数赋予某个窗口的名称时，它可以告诉我们这个窗口是否存在。它只能使用窗口名称，不能使用引用窗口的变量。

可以编写一段程序，利用the windowList关闭所有MIAW。它从the windowList里得到窗口的个数，然后用close和forget命令关闭所有窗口：

```
on closeAllMIAWs
  n = count(the windowList)
  repeat with i = 1 to n
    close window(1)
    forget window(1)

  end repeat
end
```

24.1.5 MIAW事件处理程序

MIAW还可以使用很多特殊的事件处理程序，其中包括典型的窗口事件，如打开、关闭和移动窗口。

下面列出的每一个处理程序都可以用在 MIAW的Director影片的影片剧本里。以下列出了全部处理程序：

■ on activateWindow —— 如果窗口当前没有被激活，而用户点击它想要激活它时，就调用该处理程序。

■ on colseWindow —— 当用户点击关闭按钮关闭窗口，或用 close命令关闭窗口时，就调用该处理程序。

■ on deactivateWindow —— 如果该窗口是处于激活状态的，而用户点击了其他窗口，则调用该处理程序把窗口变为非激活状态。

■ on moveWindow —— 用户每次在屏幕上拖动MIAW时，该处理程序被调用。调用的具体时间是在用户释放鼠标按钮的那一刻。

■ on openWindow —— 当MIAW初次被打开时，该处理程序被调用。

■ on resizeWindow —— 每当用户拖动窗口的顶角或边框重新调整窗口的尺寸时，该处理程序被调用。

■ on zoomWindow —— 每当用户点击缩放、最大化或最小化按钮时，该处理程序被调用。

参见第13章“重要的Lingo句法”里的13.4节“使用处理程序”，可以获得更多有关创建处理程序的信息。

24.2 创建对话框

有了MIAW，我们就不必总是使用平淡、普通的 Mac和Windows的标准警告对话框和其他普通对话框了。既然舞台上的画面可以千奇百怪，对话框为什么不可以变个花样呢？我们可以运用窗口属性和事件处理程序构筑自定义的对话框和警告对话框。本节将介绍具体的制作方法。

24.2.1 确认对话框

确认对话框通常提一个问题，让用户回答“是”或“否”。在大多数软件里，“是”或

“否”都表现为OK或Cancel。不过，用MIAW可以把它们设置为任何内容。

图24-1是带有MIAW确认对话框的舞台。与本章前面的一个例子一样，它只是一个普通的MIAW。对它的windowType进行了设置，使得该窗口是一个不可移动的矩形。它的 modal 属性已被设置为 TRUE，这样用户必须回答其中的问题。

图24-1 用MIAW可以很容易地制作风格独特的对话框

下面这段程序以正确的 windowType和modal打开了这个MIAW：

```
on mouseUp
  global gFunkyDialog
  gFunkyDialog = window("Funky Dialog")
  gFunkyDialog.filename = "24funkydialog.dir"
  gFunkyDialog.windowType = 1
  gFunkyDialog.modal = TRUE
  gFunkyDialog.visible = TRUE
end
```

在这个MIAW内，两个按钮可以与一些命令相联系，使得这个对话框关闭，并告诉舞台应当做什么。例如，下面是与 Yes按钮联系的命令：

```
on mouseUp
  close(the activeWindow)
  forget(the activeWindow)
  tell (the Stage) to continueYes
end
```

on continueYes处理程序应当位于主影片里。主影片里还应当有由 No按钮调用的 on continueNo处理程序。

在前面这段程序里， the activeWindow被方便地用来确定要关闭哪个 MIAW。当用户点击它时，它毕竟是要变为激活状态的窗口的。使用这一功能比用全局变量来传递信息要更方便些。

24.2.2 警告对话框

警告对话框也是非常简单的MIAW。不过，我们需要创建能够处理多种警告的 MIAW。要实现这一点，可以使用tell命令向MIAW里放置不同的文本，以便显示不同的信息。图 24-2就是这样一个对话框。

图24-2 可以用MIAW来创建自定义的警告对话框

下面的处理程序可以创建这样一个 MIAW：

```
on mouseUp
  global gAlertBox
  gAlertBox = window("Alert Box")
  gAlertBox.filename = "24alertbox.dir"
  gAlertBox.windowType = 0
  gAlertBox.modal = TRUE
```

```
  tell gAlertBox
    member("Text").text = "Danger Will Robinson! "
  end tell

  gAlertBox.visible = TRUE
end
```

这段程序与那个打开MIAW的处理程序几乎相同，只是使用了tell命令来改变MIAW里的文本。

24.2.3 文本输入对话框

另外一类MIAW对话框请求用户向其中输入更多信息。在这类对话框里可以使用单选按钮、复选框甚至文本输入栏。

下面这个MIAW的例子请求用户输入他的名字。图24-3是这个对话框的外观。中间的文本输入栏是一个域演员。

图24-3 MIAW可以通过文本输入栏或其他界面元素收集信息

在主影片里创建这个MIAW与创建其他MIAW的方法是相同的。不过，当用户点击OK时所执行的程序却有些复杂。它用tell命令把域里的文本传送回舞台：

```
on mouseUp
  text = member("Text Input").text
  close(the activeWindow)
  forget(the activeWindow)
  tell (the Stage) to textInputDone(text)
end
```

Cancel按钮的程序是相似的，不过这个按钮只需要传回一个常数VOID，而不是文本。随后，主影片用类似下面的方法处理这些输入：

```
on textInputDone text
  if text = VOID then
    put "Cancelled. "
  else
    put "Text Entered: "&&text
  end if
end
```

当然，在正式的软件里，on textInputDone很可能会把用户的名字存储在一个变量里，或放在一个域里。它很可能还要使用一个go命令前进到影片的下一部分。

参见第14章“创建行为”里的14.5节“完整的按钮行为”，可以获得更多有关创建行为的信息。
参见第16章“控制文本”里的16.5节“使用键盘输入”。

24.3 MIAW的其他用途

确认对话框、警告对话框和输入对话框只是MIAW的众多用途中的三种。在Director里能够实现的任何功能几乎都能在MIAW里实现。因此我们可以随意发挥。

有些创作者把MIAW用作放映机的主屏幕。他们把舞台做得尽可能小，甚至把舞台放在

显示器的显示范围之外，使它的横坐标或纵坐标变为负值。这样，就得到了一个可以移动的主屏幕。即使对于坚持使用不可移动的矩形舞台窗口的 Mac放映机来说，也可以制作可以移动的主屏幕了。

下面列出MIAW的其他一些用途：

■ Shockwave播放器 —— 由于MIAW既可以使用因特网网址，又可以使用文件名，为什么不可以用这种方法在放映机里打开一些 Shockwave元素呢？

■ 超链接 —— 在教学软件里，MIAW可以被用作字典窗口或附加信息窗口。使用文本里的超链接可以激活这些窗口。

■ 多部影片 —— 如果需要同时播放多部影片，可以用单个放映机打开两个 MIAW。

■ 调试 —— 可以在MIAW里放置一些关于主影片的额外信息，供我们进行调试时使用。其中的内容可以用tell命令更新。当整部作品完成后，删除这些 MIAW或隐藏它们就可以了。

■ 舞台覆盖层 —— 如果制作一个普通的矩形MIAW，可以把它放在舞台上面，而用户甚至不知道它竟是另外一个窗口。可以用这种方法在影片里放置动画或其他外部元素。

24.4 使用链接的影片

实现舞台覆盖层的另一种方法是使用链接的影片，而不是使用 MIAW。链接的影片就是一个输入到主影片里的演员。可以把链接的影片放在舞台上，而用户却不知道它是一个独立的影片。这个链接的影片的剪辑室和演员表仍旧作为独立的文件而存在，但它们只作为演员出现在主影片里。

当有了这样的演员后，就可以把它们放在舞台上，并调整它的位置了。我们甚至可以在一系列帧里逐渐移动它的位置。

注释 链接的影片与影片片断间的区别在于链接的影片里的所有剧本仍旧处于激活状态，并仍在起作用。而在影片片断里只有行为和其他一些剧本在起作用。影片片断只是为了演出一些动画画面，而链接的影片则是更复杂的交互式影片。

参见第11章“高级技巧”里的 11.2节“影片片断和链接的影片”，可以获得有关影片片断的基本信息。

24.5 使用MUI Xtra对话框

Director所附带的MUI Xtra用于创作环境的各个部分，它们可以创建与行为加到角色上时所调出的对话框相似的对话框。 MUI代表Macromedia用户界面，是Macromedia制作其所有软件产品时所遵从的规则。

Macromedia还为这个Xtra提供了现成的Lingo界面，使我们能够创建一些标准的以及自定义的对话框。对话框里的所有元素的外观都与 Macromedia的标准界面元素相同，不过这并不是什么缺点。Macromedia已经创建了一整套与标准对话框相似的跨平台元素。

调用MUI Xtra的方法有五种。前四种都是一些标准的对话框类型：打开文件、保存文件、打开网页和警告对话框。使用 MUI Xtra的第五种方法是从头开始创建自定义的对话框。

24.5.1 创建打开文件对话框

创建打开文件对话框是非常简单的。我们所需要做的就是创建这个 Xtra的一个实例，使

用它的FileOpen处理程序生成一个对话框，然后返回它的结果。图24-4就是一个例子。下面的处理程序就可以完成这个任务：

```
on muiFileOpen
  gMUI = new(xtra "mui")
  filename = FileOpen(gMUI,the pathname)
  gMUI = 0
  return filename
end
```

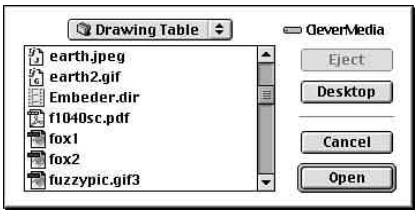

图24-4 用MUI Xtra制作的打开文件对话框

FileOpen的另一个参数是文件的缺省路径名。该函数返回最终得到的文件名。在此之后，我们应当通过把该实例设置为VOID或0而删除它。

24.5.2 创建保存文件对话框

MUI Xtra的FileSave处理程序与上面的例子的工作方式相似。它需要两个参数：文件的缺省名称和将要显示在对话框里的文字。图24-5是一个例子。

```
on muiFileSave
  gMUI = new(xtra "mui")
  filename = FileSave(gMUI, "myfile", "Save Game")
  gMUI = 0
  return filename
end
```

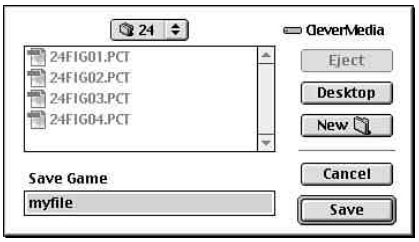

图24-5 用MUI Xtra制作的保存文件对话框

FileSave命令返回将要创建的文件的路径名。

24.5.3 创建打开网页对话框

GetURL处理程序可以打开一个名为 Open URL的对话框。我们可以指定一个缺省的网址，并确定该对话框是否可以移动。图 24-6是一个例子。

```
on muiGetURL
  gMUI = new(xtra "mui")
  default = "http://clevermedia.com"
  moveable = TRUE
  filename = GetURL(gMUI, default, moveable)
  gMUI = 0
  return filename
end
```

图24-6 用MUI Xtra创建的Open URL对话框

24.5.4 创建警告对话框

创建警告对话框比较复杂，因为其中有许多可选项。这个处理程序能创建图 24-7所示的警告对话框。这是一个带有一个“ caution ”图标和一个OK按钮的对话框。

```
on muiAlert
  gMUI = new(xtra "mui")
```

```
  list = [:]
  addProp list, #buttons, #Ok
  addProp list, #default, 1
  addProp list, #title, "Alert! "
  addProp list, #message, "Danger Will Robinson"
  addProp list, #icon, #caution
  addProp list, #moveable, FALSE
  res = Alert(gMUI,list)
  gMUI = 0
  return res
end
```

图24-7 用MUI Xtra创建的一个简单的警告对话框

从这个处理程序里可以看出，有一个列表被送进了 Alert函数。这个列表的内容是决定对话框的外观的各种选项。

第一个属性#buttons可以被设置为#OK、#OKCancel、#AbortRetryIgnore、#YesNoCancel、#YesNo或#RetryCancel。其中每一个选项都决定着对话框里按钮的数量以及每个按钮的名称。

#default属性决定哪一个按钮是缺省按钮。这个按钮的外观与其他按钮不同，并对回车键有所反应。

#title属性允许我们自定义警告窗口内的图标以及标题条上的文字。 #message和#icon属性允许我们确定警告的内容。 #icon的选项有#stop、#note、#caution、#question或#error。

#moveable属性使我们能够确定警告对话框是否可以被移动。

这类对话框返回被按下的按钮的编号。因此如果它的按钮是#OKCancel，它可以返回1或2。

24.5.5 创建自定义的MUI对话框

自定义的 MUI对话框比警告对话框更复杂，因为我们需要指定对话框的各项内容以及其中的界面元素。

学习制作自定义对话框的最好方法是看一个实例。但即使是一个简单的例子也有许多Lingo命令行。图 24-8所示的简单的自定义对话框就是由下面的处理程序创建的。

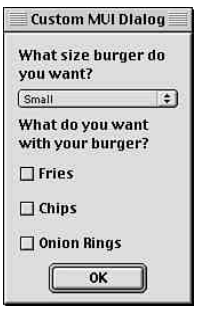

图24-8 用MUI Xtra创建的自定义对话框

```
on muiCustom
  global gMUI
  gMUI = new(xtra "mui")
```

接下去，必须定义该窗口的属性。 MUI Xtra允许我们使用getWindowPropList取得缺省的属性列表的一个拷贝，而不需要我们从头开始编写很长的窗口属性列表。这个缺省的列表里含有定义一个窗口所需要的全部属性，因此我们所要做的事情就是修改有关属性的设置：

```
  windowProps = getWindowPropList(gMUI)
  windowProps.type = #normal
  windowProps.name = "Custom MUI Dialog"
  windowProps.callback = "myCallbackHandler"
  windowProps.width = 160
  windowProps.height = 230
  windowProps.mode = #pixel
```

在这里， name和callback处理程序以及窗口的 width、height和mode被修改了。callback处

理程序是每次用户使用这个对话框后被调用的影片处理程序的名称。mode可以被设置为#data、#dialogUnit或#pixel。其中第一个选项可以帮助我们确定对话框内容的布局，另外两个选项允许我们指定各个项目的位置。

在设置了窗口的属性之后，应当着手开始创建将要添加到对话框里的界面元素。先创建一个列表，从而把这些项目都添加进去。

```
list = []
```

现在就可以创建第一个元素了。同窗口属性一样，元素的属性也十分复杂，因此 MUI Xtra专门设置了一个getItemPropList函数，它可以返回一个缺省的属性列表。对这个列表进行修改，就得到了特定的界面元素。下面是一个标签元素：

```
element = getItemPropList(gMui)
element.type = #label
element.value = "What size burger do you want? "
element.locH = 10
element.locV = 10
element.width = 140
element.height = 40
add list, element
```

接下去，可以添加一个下拉菜单元素。这个元素需要一个特殊的 #attributes属性，它将指定关于这种界面元素的某些特定信息。

```
element = getItemPropList(gMui)
element.type = #popupList
element.locH = 10
element.locV = 45
element.width = 140
element.height = 20
element.attributes = ¬
  [#popupStyle: #tiny, #valueList: ["Small", "Medium", "Large"]]
add list, element
```

在另一个标签之后，是三个复选框。

```
element = getItemPropList(gMui)
element.type = #label
element.value = "What do you want with your burger? "
element.locH = 10
element.locV = 70
element.width = 140
element.height = 40
add list, element

element = getItemPropList(gMui)
element.type = #checkBox
element.title = "Fries"
element.locH = 10
element.locV = 110
element.width = 140
element.height = 20
add list, element

element = getItemPropList(gMui)
element.type = #checkBox
```

```
  element.title = "Chips"
  element.locH = 10
  element.locV = 140
  element.width = 140
  element.height = 20
  add list, element

  element = getItemPropList(gMui)
  element.type = #checkBox
  element.title = "Onion Rings"
  element.locH = 10
  element.locV = 170
  element.width = 140
  element.height = 20
  add list, element
```

所需要的最后一个元素是OK按钮。

```
  element = getItemPropList(gMui)
  element.type = #defaultPushButton
  element.title = "OK"
  element.locH = 40
  element.locV = 200
  element.width = 80
  element.height = 20
  add list, element
```

当所有这些元素都创建好以后，就调用 Initialize来创建对话框。需要把窗口属性和元素属性传送给它。然后，使用 Run创建对话框。

```
  Initialize(gMUI, [#windowPropList: windowProps, #windowItemList: list])
  Run(gMUI)
end
```

与其他 MUI对话框不同的是，自定义对话框不返回一个简单的值，而是用所定义的callback处理程序传送行动信息。它传送任何鼠标操作或任何项目的改动。它把这些信息作为三个参数传送：发生了什么、它影响哪个项目以及那个项目的属性列表。

下面这个处理程序可以接收这些消息。它是一个很简单的处理程序，只向消息窗口发送消息：

```
on myCallbackHandler action, elementNumber, elementList
  global gMUI
  put "Action Reported: "&&action
  if action = #itemClicked and elementList.title = "OK" then
    put action, elementList.title
    put gMUI
    Stop(gMUI,0)
    gMUI = VOID

  else if action = #itemChanged then
    newval = elementList.value
    itemName = elementList.title
    put itemName&&"changed to"&&newval
  end if
end
```

在真实的软件里，应当把用户提供的信息记录在这个处理程序的全局变量或域里。每当

改变一个项目后，都可以把所进行的改动记录下来。另外一种方法是通过全局变量 gMUI引用所有界面元素的值。

自定义MUI对话框里不仅能够安置弹出菜单、标签和复选框，而且可以有单选按钮、滑动条、位图、可编辑的文本域、分界符以及其他类型的按钮。

要想了解全部可以使用的元素，可以在消息窗口内试做如下操作：

```
put Interface(Xtra "Mui")
```

这样得到的将是一个很长的清单，其中列着 MUI对话框里可以使用的全部元素。这个清单的内容也是最新的。Macromedia经常更新MUI Xtra，以适应软件的新需要。Director每次升级后其MUI Xtra都有所不同。

24.6 MIAW及其相关内容的故障排除

■ 每当不再需要某个MIAW时，应当记着使用forget命令。用close命令只能隐藏它。

■ 一定要在MIAW里测试所有代码。如果MIAW运行时发生了Lingo错误，Director不一定能返回关于错误类型及错误的位置的有效信息。

■ 在测试MIAW时，在两种平台及其各种版本上都要进行测试，以确保它没有问题。

■ MUI Dialog Xtra不是供创作者使用的，而是Macromedia准备用于Director和Xtra的进一步升级的。它属于非常高级的技巧，初学者最好不要使用它。它也并不完全被 Macromedia所支持。

■ MUI Xtra的属性列表里的错误会导致Director陷入死循环。在使用MUI Xtra时，要经常注意把影片存盘。

■ 对于不同的屏幕尺寸， MIAW的位置可以是不同的。一定要在不同的显示器设置状态下测试MIAW，并根据具体情况调节rect属性。

24.7 你知道吗

■ 可以用一个MIAW文件创建多个MIAW。只要在显示MIAW前用tell命令让MIAW走到MIAW影片文件的另一帧就可以了。你甚至可以使用舞台所使用的那个影片文件！

■ 可以用drawRect缩放MIAW，其中所有位图以及其他可缩放的元素也都随之改变。例如，可以用这种方法把一个MIAW显示为双像素尺寸。

■ 当自定义的MUI对话框在屏幕上时，我们可以改变其中的项目，使得在 MUI对话框里的位图会随着另一个界面元素的改变而改变。

■ 在MIAW的某段程序里使用(the stage).picture，或使用tell命令，就可以像为舞台拍照那样为MIAW拍照。在拍照时MIAW甚至可以是处于隐藏状态的。