第七部分　使用高级的Lingo

第21章　Director环境的控制

Lingo 提供了许多种对播放环境进行控制和反馈的途径。我们可以创建自定义的菜单并且改变光标。我们也可以检测用户有多长时间没有进行任何操作，并做一些事情；我们甚至可以确定用户的计算机的类型以及现在的时间，并让影片做出某种反应。

21.1　菜单的使用

当我们创建一个放映机后，它是在普通的窗口中运行的。在苹果机上，它只是一个简单的矩形框；在Windows中，它看起来像一个典型的窗口，但没有菜单条。为了使得放映机看起来更像一个普通的程序，我们要自己加上一个菜单条。

21.1.1　菜单的创建

我们可以用installMenu命令在任何放映器中加一个菜单条。这个命令可以将标准的苹果菜单条放置在苹果机的屏幕顶端，但其中只有我们所指定的项目。在 Windows中，它在放映机窗口的顶端放置一个标准的菜单条。

在我们创作时，当 installMenu命令发出后，它将取代 Director的菜单条，直至影片结束。这使得它很容易进行测试。在 Shockwave中，installMenu根本就不起作用。

要使用installMenu，首先必须创建一个包括菜单描述的域演员。下面列出了这样的一个典型的域：

```
menu: @
menu: File
Open/O | myOpenHandler
(-
Quit/Q | halt
menu: Edit
Cut(
Copy(
Paste(
Clear(
menu: Navigation
Main Menu/M | go to frame "main"
Chapter 1/1 | go to frame "one"
Chapter 2/2 | go to frame "two"
Chapter 3/3 | go to frame "three"
```

这段文本的第一行“ menu: @ ”告诉Director在苹果的菜单条中放置一个含有其全部内容菜单条。这只对Mac有效。在Windows中，这个命令会产生一个带有方块字符的小菜单。

第二行创建了一个名为“ File ”的菜单。接下去的三行在菜单中设置了三个项目。第一项

是“ open ”。斜杠后面是该命令的快捷键。因此， Command+O (Mac)或Ctrl+O (Windows)就是该菜单项的快捷键。

在竖线的后面是将要执行的 Lingo命令。在这个例里，它是一个自定义的影片处理程序myOpenHandler。再下面一行简单的“ (- ”用来在菜单中添加一条分界线。该菜单中的第三项是“ Quit ”，在这里Q是快捷键，并执行Lingo的halt命令。

接下去的菜单是“ Edit ”。由于每行的结尾都有“ (”，因此该菜单中的所有项目都是淡化的处于非激活状态的。

注释 最好每次制作自定义的菜单时都要包括“ File ”和“ Edit ”菜单。几乎所有的程序都使用它们，即使其中的项目都是处于淡化的非激活状态也没有关系。

最后一个菜单称为“ Navigation ”，其中包含四行，可以把用户带到四个不同的帧。结果得到的菜单如图21-1所示。

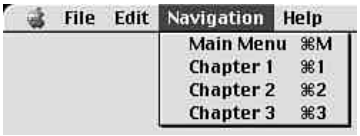

图21-1 这个自定义的菜单是由一个域演员和一个installMenu命令创建的

要真正使用该菜单，还需要在 installMenu命令中指定它的演员。该命令适合于放在 on startMovie处理程序里：

```
on startMovie
  installMenu member("menu")
end
```

在菜单域中的特殊字符都很容易混淆。实际上还有更多的字符。表 21-1将它们全部列出，并对其功能进行了描述。

表21-1 菜单的描述符号

符 号	实 例	描 述
@	menu: @	在Mac上创建苹果菜单,与当前现有的苹果菜单项共同存在
!v	!vMusic	在每一项旁边设置一个选中标记
<B	Copy<B	将此项设置为粗体
<I	Copy<I	将此项设置为斜体
<U	Copy<U	将此项设置为下划线
<O	Copy<O	将此项设置为加轮廓线
<S	Shadow	将此项设置为阴影形式
¦	Quit ¦ halt	把一个Lingo处理程序命令与此项联系起来
/	Quit/Q	添加一个Command/Ctrl快捷键
(	Copy(	淡化某个项目，使其不可选
(-	(-	在菜单中创建一个分界线

在表21-1中列出有关式样的项目只用于Mac。Windows菜单不采用这类式样。

21.1.2 菜单的控制

我们也可以通过Lingo命令对已建立好的自定义菜单进行控制。菜单条本身被视作一个对象,并由关键词menu及该菜单的名称或编号来引用。例如 ,用下面的方法可以得到从左面数的第二个菜单的名称:

```
put the name of menu 2
-- "File"
```

可以注意到，这里必须用带有“ the ”的旧句法，而不是新的Director 7的“ 点 ”句法。在Director 7里，对菜单方面的功能没有更新，因而不能使用“ 点 ”句法。

要得到菜单条中总的菜单个数，使用the number of menues：

```
put the number of menus
-- 4
```

也可以通过使用menuitem关键词得到任何一个菜单项的名称：

```
put the name of menuitem 1 of menu 4
-- "Main Menu"
```

如我们所料，可以使用number of menuitems 得到任一个菜单里的菜单项的总数。下面这个例子还演示了如何通过名称引用菜单：

```
put the number of menuitems in menu "Navigation"
-- 4
```

我们也可以设置菜单项的名称。然而，我们不能设置菜单的名称：

```
the name of menuitem 3 of menu 2 = "Exit"
```

the checkMark、the enabled和the script这三个属性使我们能够改变菜单项的另外三种特性。第一个属性可以在菜单项的旁边放置一个选中标志；第二个属性可以把某个项目淡化，使其不能使用；最后一个属性可以改变某一个项目的剧本。

另一种改变菜单的方法是通过使用Lingo字符串和文本演员命令，再重新使用installMenu，从而更新菜单的描述域。

21.2 光标的使用

在Director 7中有三种改变光标的方式。第一种是通过cursor命令采用内置的光标；第二种是采用cursor命令，再使用一个或两个代表黑 -白光标的位图；第三种是用 Cursor Xtra制作彩色的活动光标。

21.2.1 使用内置的光标

用cursor命令使用30个内置光标之一是很简单的。我们所需要做的就是给 cursor命令一个光标编号。例如，要把Mac的光标改为“ 手表 ”，或把Windows的光标改为“ 沙漏 ”，可以使用这个语句：

```
cursor(4)
```

下面这个简单的行为可以完成这个任务：当用户把光标掠过一个角色时，光标变为“ 手指 ”光标：

```
on mouseEnter me
  cursor(280)
end

on mouseLeave me
  cursor(0)
end
```

采用光标编号0，可以返回到光标的正常状态。通常这意味着光标返回箭头形状。

表21-2列出了全部可供使用的光标。图 21-2给出了这些光标在Mac上的模样。 Windows的

光标与之稍有不同。例如，在 Windows里，“ 手表 ”表现为“ 沙漏 ”。如果一致性非常重要，我们应该在跨平台的环境下测试光标。

表21-2 光标的编号

光标名称	光标编号
Arrow(箭头)	-1
I-Beam(I形光标)	1
Crosshair(十字线)	2
Crossbar(十字条)	3
Watch/Hourglass(手表/沙漏)	4
Blank(空白)	200
Help(帮助)	254
Finger(手指)	280
Hand(手)	260
Closed Hand(拳头)	290
No Drop Hand(“ 禁止放下 ”手)	291
Copy Close Hand(“ 拷贝 ”拳头)	292
Pencil(铅笔)	256
Eraser(橡皮擦)	257
Select(选择)	258
Bucket(颜料杯)	259
Lasso(套索)	272
Dropper(吸管)	281
Air Brush(气刷)	301
Zoom In(放大镜)	302
Zoom Out(缩小镜)	303
Vertical Size(纵向尺寸)	284
Horizontal Size(横向尺寸)	285
Diagonal Size(对角线尺寸)	286
White Arrow(Mac)(白色箭头)	293
Black Arrow with white outline(Windows)(带白边的黑色箭头)	293
Magnify(放大镜)	304
Wait Mouse1(Mac)(“ 等待 ”鼠标1)	282
Wait Mouse2(Mac)(“ 等待 ”鼠标2)	283

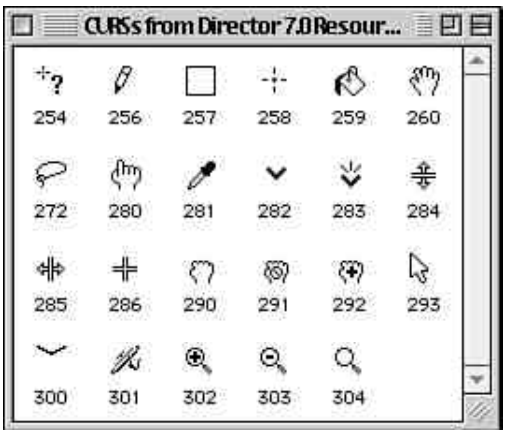

图21-2 Director所使用的一些光标。它们是用ResEdit从Director的资源文件里获得的

21.2.2 自定义的位图光标

创建光标的第二种方法是使用位图。利用位图可以定义任意的光标，只要它是黑 -白的，而且是静态的就可以。

这样做的关键是创建两个位图演员。每个演员的尺寸应该不超过 16×16个像素，且位深为1-bit。第一个演员是实际的光标，第二个演员是该光标的蒙版。在第二个演员里，黑色像素是光标的蒙版，白像素是透明的。

两个位图应该按照它们的套准点对齐。套准点的位置就是光标的实际的“ 热点 ”，因此在放置它时要特别留意。图21-3给出了两个Paint窗口，其中一个是光标，另一个是蒙版。

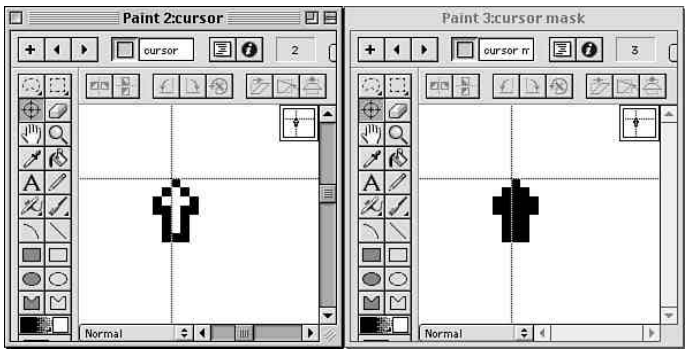

图21-3 两个Paint窗口给出了自定义光标的两部分,套准点显示了光标的热点

有了这两个演员，就可以使用 cursor命令了。使用方法与以前相同，只要以一个列表作为参数。下面是一个例子：

```
cursor([member "cursor", member "cursor mask"])
```

如果位图不是1-bit的，或者演员名称是错误的， cursor命令则不做任何事情。

21.2.3 Cursor Xtra

Director 6.5引入的Cursor Xtra使我们能够用一个或多个 8-bit位图创建自定义的光标。创建这类光标的方法是通过使用如图 21-4所示的Cursor Properties Editor。

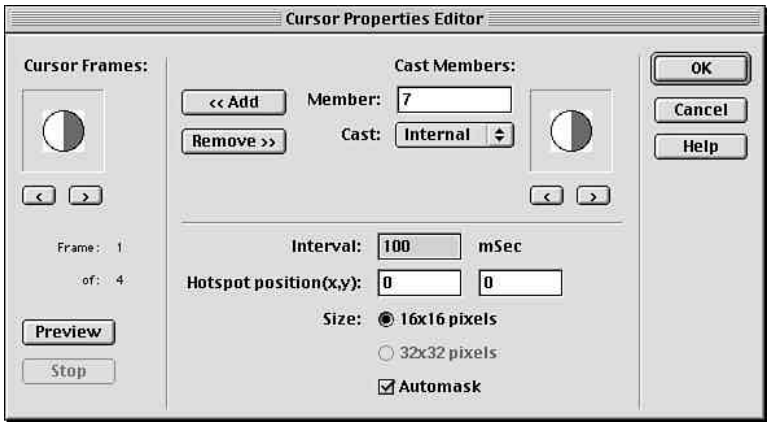

图21-4 Cursor Properties Editor使我们能够创建和修改活动的光标

为一个光标添加演员是相当容易的。只要用该对话框中的 Cast Members部分找到适当的演员，再使用Add按钮，就可以把它添加进来。我们还可以设置光标的活动速度和热点的位置。如果不想让光标呈透明状态，可以选中 Automask功能。

创建了光标后，可以使用 cursor命令，但这次只有一个演员作为参数：

```
cursor(member("Custom Cursor"))
```

参阅第8章“ 其他演员类型 ”中的 8.5节“ 添加光标 ”，以获得有关创建自定义光标的背景知识。

21.3 timeout的使用

假设我们要制作一个放在公共场所的信息站(kiosk)。用户理应走到这个计算机前，并从第一个屏幕画面开始。然而，我们不能期望每一位用户使用完时都会按下“ Im Done ”按钮。其结果是下一个用户走过来时，看到的是演了一半的演示。

处理类似情况的一种方法是当计算机已经空闲了一段时间后，对其进行检测，然后自动回到影片的开头。

Director中有些Lingo命令就可以处理此类请求。其中一个主要的处理程序是 on timeOut。这个处理程序在停止操作三分钟后被调用。如果这个处理程序不存在，就不会发生任何事情。下面是一个典型的on timeOut处理程序：

```
on timeOut
  go to frame "intro"
end
```

注释 这个处理程序需要放在影片剧本中。如果放在行为中则不生效。

可以使用一些影片属性，从而改变对 on timeout剧本的调用方法。例如，如果我们想让timeout发生在两分钟后，而不是三分钟后，可以使用 the timeOutLength属性。它的值以ticks(1/60秒)来计算：

```
the timeOutLength = 60*60*2
```

用the timeOutLapsed属性可以访问timeout计时器。它会告诉我们从上一次操作到现在所经过的时间。无论何时点击鼠标或按下某个键，这个计时器就将重新计时。然而，我们可以通过设置timeOutMouse或timeOutKeyDown属性为FALSE将这两个条件中的一个(或把这两个条件全部)关掉。

对timeOut的进一步修改就是被调用的那个处理程序。通常，它是 on timeOut处理程序。然而，我们也可以用 timeOutScript属性将其设置为其他名称。只要将它设置为一个字符串就可以了，这个字符串将被用作影片剧本处理程序的名称：

```
the timeOutScript = "myTimeOutHandler"
```

在大多数情况下，我们无须设置 timeOutScript，只使用on timeOut就可以了。然而，如果我们想要根据用户离开程序的不同位置以几种不同方式处理 timeOut，就可以把它设置为其他名称。

21.4 关于计算机

我们还可以使用其他许多影片属性，以了解有关正在运行 Director、放映机或Shockwave

短程序的计算机的一些情况。下面列出了这些属性：

■ the platform(平台) —— 根据影片播放的平台，它可以返回“ Macintosh,PowerPC ”或“ Windows,32 ”。

■ the runMode(运行模式) —— 返回“ Author ”、“ Projector ”、“ Plugin ”或“ Java Applet ”。第一项告诉我们影片正在Director中运行；第二项表示影片以放映机的形式运行；第三项是指的是Shockwave；第四项表示影片作为一个Java 短程序运行。

■ the colorDepth(颜色位深) —— 返回影片正在使用的显示器的位深，如8、16、24或32。

■ the environment(环境) —— 返回一个含有前面三个属性的值的列表。

■ the desktopRectList(桌面矩形列表) —— 返回一个列表，其内容是与计算机相连接的一个或多个显示器的矩形。

■ quickTimeVersion( )(quickTime版本) —— 这不是一个属性，而是一个函数。它返回计算机上的QuickTime的版本。如果其版本低于3.0，则返回2.1.2。

■ version(版本) —— 这也不是一个属性，而是一个当Director启动时自动创建的全局变量。无论Director引擎是Director、放映机，还是Shockwave，它都含有它的版本号，如“ 7.0 ”。

如果用户的计算机没有能力执行我们想让它执行的任务，我们可以用这些属性来决定让影片如何运行。例如，要测试计算机的设置是否为16-bit或更高的位深，我们应该这样做：

```
on startMovie
  if the colorDepth < 16 then
    alert "Please set your monitor to 16-bit. "
    halt
  end if
end
```

或者，如果我们想了解用户的显示器的宽度是否至少为 800像素，可以对the deskTop RectList的第一项的宽度进行测试。由于大多数用户只有一台显示器，而那些连接着两台显示器的计算机的主显示器的宽度极少会少于800像素，下面是一个很好的测试：

```
on startMovie
    if (the deskTopRectList)[1].width < 800 then
    alert "Please set your monitor to 800 pixels across. "
    halt
  end if
end
```

参见第34章“ 运行性能 ”里的34.1节“ 为目标计算机设计 ”，可以获得使用计算机的信息的实例。

21.5 显示时间

Lingo有几种显示时间和日期的方法。首先有一个the date属性。这个系统属性稍有些独特，因为我们可以为它加short、long或abbr等前缀。

```
put the date
-- "12/9/98"
put the short date
-- "12/9/98"
put the long date
-- "Wednesday, December 9, 1998"
put the abbr date
```

```
-- "Wed, Dec 9, 1998"
```

abbr前缀也可以写成abbrev或abbreviated。我们可以直接使用返回的字符串，也可以用子字符串表达式截取其中的部分内容。

```
put (the long date).item[1]
-- "Wednesday"
put (the long date).item[2].word[1]
-- "December"
the itemDelimiter = "/"
put integer((the date).item[2])
-- 9
```

还有一个the time属性，它与前面那个属性相似。然而，只有前缀 long会使结果有所不同。

```
put the time
-- "9:18 PM"
put the short time
-- "9:18 PM"
put the long time
-- "9:18:43 PM"
put the abbrev time
-- "9:18 PM"
```

注释 the date的字符串的格式取决于计算机的设置。如果我们查看 Mac或Windows的“控制面板”，就会意识到用户可以选择多种显示日期的方式。Director在the date里反映了这些设置。

我们可以用子字符串表达式来得到 the time里我们感兴趣的那部分内容：

```
put (the time).word[1]
-- "9:20"
the itemDelimiter = ":"
put integer((the time).word[1].item[2])
-- 20
put integer((the long time).word[1].item[3])
-- 57
```

当我们需要得到现成的可供使用的字符串或整数时，用 the time 和 the date是非常好的，不过，在处理日期方面还有一种更好的方法。 the systemDate属性可以返回一种新的结构：date。

```
Put the systemDate
-- date( 1998, 12, 9 )
```

如果我们所料，可以从date结构里抽取三个属性：year、month和day。

```
d = the systemDate
put d.day
-- 9
put d.year
-- 1998
put d.month
-- 12
```

date结构的另一个极大的优点是，我们可以把它与整数相加，它可以为我们做各种各样的计算。请看下面的例子：

```
put the systemDate
```

```
-- date( 1998, 12, 9 )
put the systemDate + 1
-- date( 1998, 12, 10 )
d = the systemDate
put d + 10
-- date( 1998, 12, 19 )
put d + 30
-- date( 1999, 1, 8 )
put d + 365*3
-- date( 2001, 12, 8 )
put d - 365
-- date( 1997, 12, 9 )
```

使用date结构，我们可以执行各种各样有趣的计算。下面这个处理程序能够计算从某一个日期开始到现在的总天数。把你的生日写成date结构，就可以计算出从你出生到现在的总天数。

```
on daysAlive birthDate
  t = 0
  repeat while TRUE
    birthDate = birthDate + 1
    t = t + 1
    if birthDate = the systemDate then
      return t
    end if
  end repeat
end
```

除了时间和日期外，还有一个更重要的 the ticks属性，它表示从 Director、放映机或Shockwave短程序开始运行到现在所经过的时间，以1/60秒为单位。

尽管把这些信息显示给用户并不十分有用，但对于定时的影片却是很重要的。下面这个短的处理程序可以创建2秒的暂停。

```
on pauseForTwo
  t = the ticks + 120
  repeat while (the ticks < t)
  end repeat
end
```

在Director 7中，出现了the ticks的一个新同伴：the milliseconds。它也是代表同一段时间，但以1/1000秒为单位。

另一个名为the timer的属性也使用tick为单位。它与the ticks之间的区别在于我们可以随时用startTimer把the timer复位。这个属性只是提供了方便，而没有添加什么新功能。凡是用 the timer能够完成的任务都能用一个变量、一些运算和 the ticks来完成。下面的处理程序可以创建2秒钟的暂停，但使用的是the timer：

```
on pauseForTwo
  startTimer
  repeat while (the timer < 120)
  end repeat
end
```

21.6 内存管理

有几个Lingo命令和属性可以帮助我们查看和控制对内存的使用。对内存的控制大都与演

员的调入和卸载有关。

21.6.1 演员的调用

出现在Director影片中的演员可以存在于内部演员表、外部演员表或作为演员表链接的外部文件中。这意味着它们存在于硬盘或 CD-ROM的文件里。要把演员显示在舞台上，首先要把它们调入计算机的内存。

Director自动完成此项任务。如果某个角色需要演员， Director会查看它是否在内存中，如果不在，则将其调入内存。由于 Director会占用越来越多的内存，因而时常需要将没有出现在舞台上的某个演员从内存中卸载，把内存空间让给即将被调入的其他演员。

尽管这是自动进行的，但我们仍可以用一套 Lingo命令和属性对调入和卸载进行控制。当速度很重要时，这样很方便，因为 Director做这些调入和卸载的工作毕竟是需要时间的。如果我们知道很快就要用到哪个演员，并且现在有时间进行调入操作，就可以用一个 Lingo命令强制调入这个演员。

在接触Lingo命令以前，先看一个典型的属性对话框。图 21-5是一个位图演员的属性对话框。

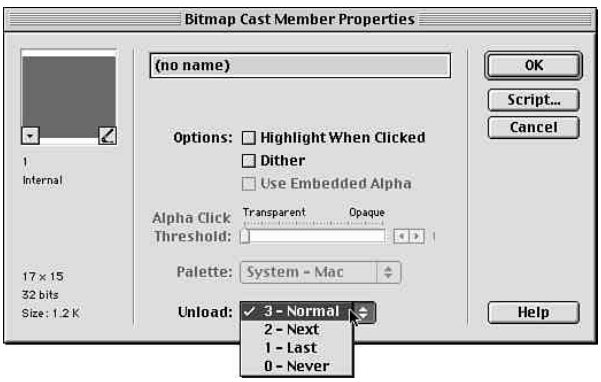

图21-5 几乎所有演员类型的属性对话框里都含有一个Unload(卸载)弹出菜单

Unload设置有四个选项：3-Normal、2-Next、1-Last或0-Never。最后一项0-Never表示把演员保留在内存中，自从演员被首次调入后，永远都不卸载； 1-Last表示把演员尽可能长时间地保留在内存中； 2-Next设置表示只要不需要，就尽快将演员卸载； 3-Normal设置表示在卸载了2-Next的全部演员，并在卸载1-Last的演员之前，卸载一些演员。

除了3-Normal之外，没有必要选择其他选项。然而，如果有一个演员经常在舞台上出现，而且它的调入和卸载会影响到影片的播放速度，把它设置为 1-Last可能是一个好方法。

用Lingo的purgePriority属性可以设置这个属性。只要使用编号就可以了，如 0、1、2或3：

```
member("large image").purgePriority = 2
```

控制调入和卸载的Lingo命令是非常直接的。为了调入一个演员到内存中，只要发出一个preLoadMember命令：

```
preLoadMember "large image "
```

我们也可以指定多个需要预调入的演员。以下这个命令可以调入所指定的两个演员，外加演员表中在这两个演员之间的其他演员：

```
preLoadMember "large image 1", "large image 7"
```

我们甚至可以只使用 preLoadMember命令本身。在这种情况下， Director试图调入所有的演员表中的演员，直至用完所有可用的内存。

还有一个preLoad命令。这个命令的参数是帧编号而不是演员名称。当独立使用该命令时，它试图调入从影片的当前帧至最后一帧中的全部演员。这只包括确实存在于剪辑室中的演员。任何不是在剪辑室中的，而是由 Lingo调用的演员(如表示按钮状态的图像)都不会被调入。

我们也可以在preLoad命令中指定两个帧，它就会调入在这两帧中使用的全部演员以及在这两帧之间的那些帧里的任何演员。如果只指定一个帧，它就调入从当前帧至指定帧之间的所有帧里的演员。

但是如果我们想用go或play命令跳到另一部影片会怎么样呢？由于那些演员不存在于当前影片中，就不能使用 preLoadMember或preLoad命令。然而，我们可以使用 preloadMovie命令。这个命令将调入另一部影片的第一帧里的全部演员。

unLoad、unLoadMember和unLoadMovie命令与前面的命令的作用相反，它们的作用是把演员从内存中卸载。

评价手工调入和卸载演员的最好方法是试验。先试着不用特殊的内存管理命令运行影片，再试着用一些命令，看看有什么不同。

21.6.2 内存信息

有几个函数可以告诉我们还有多少内存空间可供使用。简单的 freeBytes()函数可以返回Director或放映机所能使用的内存空间的多少。

另外一个更有用的函数是 freeblock()，它返回内存中的最大的连续块。由于演员需要调入连续块中，我们可以用这种方法来确认是否有这样的连续块存在。

演员的Size属性以字节为单位返回演员的尺寸。我们可以将 freeblock()与此属性组合在一起，来确定在内存中是否有足够的空间来调入演员。例如，如果我们发现一个数据量极大的声音演员无法在一台内存较少的计算机上播放，我们可以准备一个数据量较小的声音文件来替换它。下面这段程序可以确定使用哪个声音：

```
if freeBytes( ) > member("large sound").size then
  puppetSound "large sound"
else
  puppetSound "small sound"
end
```

我们也可以用ramNeeded函数确定一系列帧要用多少内存。这个处理程序将测试还有多少内存可供使用，从而决定影片是否需要跳过一些帧：

```
if ramNeeded(10,14) > the freeBytes then
  go to frame 15
else
  go to frame 10
end
```

系统属性memorySize可以返回可供Director或放映机使用的总的内存字节数。我们可以用它来检测程序是否有足够的内存来执行一个内存密集型的函数。

我们还可以使用 movieFileSize属性来获得影片的尺寸。另外一个属性 the movieFileFree

Size可以返回文件中的未用的空间。这个多余的空间就是在我们选择 File｜Save And Compact 时被扔掉的那部分空间。

参见第34章里的34.3节“ 改善运行性能 ”，可以获得调入和卸载演员的例子。

21.7 运行其他应用程序

我们可以用Director来运行用户的计算机里的其他应用程序。这是很容易的，我们所需要做的就是使用open命令：

```
open "Macintosh HD:Applications:SimpleText"
```

看起来的确简单，但要注意的是，应当该给出应用程序的完整路径名。这样有些不方便，因为即使我们想运行一些极简单的程序时 —— 如Mac的SimpleText或Windows的NotePad —— 也需要输入完整的路径。尽管有时 NotePad等简单的Windows应用程序可以不需要路径名，但我们并不能依赖于这一特点。

在使用open命令时，也可以使用预定义的启动文件来运行某个应用程序。例如，我们可以按照下面的方式用“ text file ”运行SimpleText。

```
open "Macintosh HD:text file" with "Macintosh HD:Applications:SimpleText"
```

然而，按照这种方式执行 open时存在一些问题。在 Mac上，如果SimpleText已经运行了，当我们试图通过 open用SimpleText打开一个文件时， SimpleText不会执行这个命令。SimpleText出现在屏幕上，但文件却不能被打开。在 Windows中这不成问题，因为如果Windows没有运行，就不能运行NotePad。

对于Web浏览器来说，情况要好一些，因为 gotoNetPage命令事实上可以运行用户的缺省浏览器，并进入预定义的Web网址。在第22章“ 使用Shockwave和因特网 ”中可以读到更多有关此命令的信息。

21.8 其他环境Lingo

Director还可以将自己关闭。事实上，在 Mac上，它甚至可以关闭计算机。 quit命令所做的事情正符合我们的希望，它与选择 File｜Quit/Exit的效果是相同的，这也是退出大多数应用程序的方法。执行该命令后，放映机立即退出；对于 Shockwave来说，则是影片停止播放。

quit命令工作得非常好。事实上，它好得过了头。如果我们在Director中发出一个quit命令，这个命令就会试图关闭Director。当我们创作时这可能会使我们觉得很恼火。

可以用halt命令代替quit命令。在Director中，halt命令可以使影片停止播放。在放映机中，halt命令起到了quit命令的作用，退出放映机。

如果我们要在Mac上建立一个信息查询程序，我们可能需要使用 restart或shutDown命令。它们执行的操作与Mac的Finder桌面的Special菜单中的菜单项相同。这样店主或博物馆管理员在晚间可以关闭或重新启动计算机。这样，用户就不必先退出放映机，再使用 Finder了。

我们也可以把这些命令与读时间的程序结合使用。这样就可以在某个时间自动关闭或重新启动计算机，或者如果用户在一定时间内没有进行任何操作，自动关闭或重新启动计算机。在下面这个处理程序可以在9:00p.m.关闭计算机,它可以在exitFrame处理程序内被定期调用：

```
on checkTimeShutDown
  if the time = "9:00 PM" then
    shutDown
```

```
    end if
end
```

21.9 环境Lingo的故障排除

■ 不要期盼播放影片的计算机与你的计算机的时间设置是相同的。在 the date中可以轻易地修改月份和日期，而且用户还可以很容易地修改不同项目之间的分隔符。即使目标计算机与你在同一个国家，但用户也经常会随意改变设置。如果可能，就使用 the system date。

■ 创作者浪费了很多时间调入和卸载 Lingo。在试图解决有关这些命令的速度问题之前，应当判断是否存在这类问题以及问题的确切位置。 Director的运行可能已经是优化的了。

■ 要确认自定义光标演员符合所需要的位深。普通自定义光标需要 1-bit位深,活动的光标演员需要8-bit位深，否则就不能正常工作。

■ 如果遇到了问题，应当仔细检查菜单的描述域的文本。 installMenu命令是非常严格，不允许任何错误存在。

■ 不要忘记，当为Mac添加了苹果菜单后，意味着在Windows放映机里也将添加一个奇怪的菜单。我们应该为 Mac和Windows分别编写菜单描述域。在 Windows中，应当把“ About ”菜单项放在传统的“ Help ”菜单，而不是放在苹果菜单里。

21.10 你知道吗

■ 我们可以用Lingo建立一个菜单描述域。只要使用字符串命令及其他字符串 Lingo创建或改变域，然后用installMenu使这个变化生效。这样，我们拥有了动态变化的菜单条。

■ 如果想要一个特殊尺寸的光标，只要用光标编号“ 200 ”将光标关闭，然后在最高的通道中设置一个能够跟随鼠标运动的角色就可以了。用户看不出该角色不是光标，因为它的表现和真正的光标一样。

■ 我们可以在Shockwave里使用全局变量 version重新引导拥有不同 Shockwave版本的用户。例如，我们可以制作一个 Director 5 Shockwave影片，它可以先检查版本，然后用gotoNetMovie运行Director 5、Director 6或Director 7影片。

■ Buddy API Xtra中还有许多函数能够告诉我们有关用户的系统的信息，甚至能够进行部分控制。参见第25章“ Xtra ”里的详细内容。