第19章　视频文件的控制

尽管在大多数情况下，我们在 Director中使用数字视频文件时，只是简单地把它放在舞台上，并使用它所附带的缺省控制条，但必要的时候也要采用 Lingo语言来对视频文件进行控制。处理数字视频文件并非像处理位图文件那样容易，但有许多属性能够用来改变视频文件的显示。本章主要探讨这些属性。

19.1　视频命令的使用

数字视频文件比其他素材有更多的演员和角色属性。由于其中一些属性属于演员，而另一些属性属于角色，因而概念较易混淆。甚至还有一些涉及 QuickTime 磁轨的功能也起到属性的作用。为了把问题分析得更透彻，本节将对演员属性和角色属性加以区分。

19.1.1　演员属性

数字视频演员有两组演员属性。第一组与视频文件的 Properties对话框中的内容对应。下面简要列出了最常用的数字演员：

■ center —— 通过设置TRUE或FALSE值决定是否把视频文件显示在角色的矩形框的中心位置。只有在crop属性设置为TRUE时该属性才有效。

■ controller —— TRUE或FALSE值确定是否显示缺省的QuickTime或AVI 控制条。

■ crop —— 如果设置为TRUE，即使角色的矩形框发生改变，影片还会保持原来的尺寸。如果设置为FALSE，影片将被调整，以适配矩形框的大小。

■ directToStage —— 这个属性确定是否直接在屏幕上显示视频文件，从而覆盖或忽略其他角色。如果直接在屏幕上显示就可以获得较流畅的图像显示效果，但其他许多特殊的功能就无法使用。

■ frameRate —— 这个属性可以被设置为某个数值，它就是播放数字视频文件的帧速率。采用特殊值-2时，会以最快的速度播放；采用特殊值 -1时，会以正常的速度播放。以这样的属性播放将不能播放出声音，若将 frameRate值设置为0将会回到Sync To Soundtrack的正常状态。

■ loop —— 这个属性确定了当播放结束时是否自动循环播放。

■ pausedAtStart —— 这个属性确定了影片是否一出现在舞台上就开始播放，或者等待控制条或Lingo告诉它如何播放。

■ sound —— 这个属性确定是否播放声音。

■ video —— 这个属性确定是否显示图像。

除了这些通过或不通过 Lingo都可以进行控制的属性外，还有一些 Lingo只能使用但却不能设置的属性：

■ digitalVideoType —— 返回#quickTime或#videoForWindows。

■ duration —— 返回视频文件的长度(单位为tick，即1/60秒)。

■ isVRMovie —— 如果是QuickTime VR影片，则返回TRUE 。

19.1.2 角色属性

还有几个角色属性也可以用Lingo进行设置，它们对于创建视频控制及效果是非常有效的。在这里将它们全部列出来：

■ loopBounds —— 这个属性可以对循环播放的开始和结束时间进行设置。采用一个简短的由两个项目构成的列表，例如：[0，240]。

■ movieRate —— 这个属性代表了视频文件前进的速度。如果设置为 0，意味着视频文件已停止播放；数值为1意味着正常播放；数值为2意味着以两倍于正常速度进行播放。也可以设置为负值而使得视频文件向回转。

■ movieTime —— 这个属性代表视频文件的当前时间，以tick为单位。可以将此值设置为0，从而返回到起点；或设置为该演员的duration，从而前进到终点。

■ rotation —— 信不信由你，我们可以旋转一个QuickTime视频文件。把该属性设置为旋转角度。这个属性只有在视频演员没有设置为 Direct to Stage时才能正常使用。

■ scale —— 可以以列表形式对此属性进行设置，来确定水平和垂直两个方向上的尺寸，例如[1.5,1.5]。

■ volume —— 只对声音演员有效，可以通过它对视频文件进行设置。

19.1.3 蒙版

数字视频演员也可以采用蒙版。蒙版是一个 1-bit的位图文件，它可以告诉Director视频文件中的哪个像素要显示，哪个像素不显示。图 19-1给出了3幅图像:一幅是视频文件的图像,一幅是1-bit位图，一幅是以那个1-bit位图为蒙版的视频文件的图像。

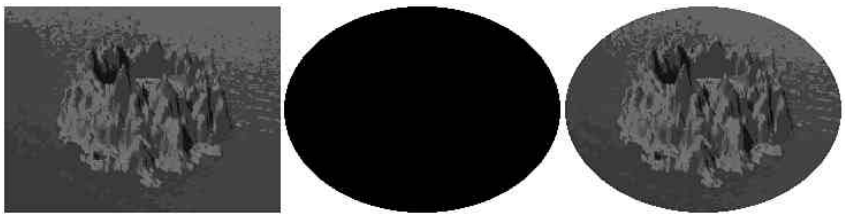

图19-1 1-bit位图可以被用作数字视频演员的蒙版

蒙版的最大好处在于当视频处于 Direct to Stage状态时也可以使用。这意味着视频播放不会因为蒙版的作用而减缓速度。

要使用一个蒙版，首先要创建一个 1-bit的位图文件并对它进行命名。然后用 mask属性把该蒙版赋予某一演员。也可以在消息窗口中进行操作，并测试它。 mask属性是一个演员属性，但这个命令需要使用角色来得到演员：

```
sprite(1).member.mask = member("myMask")
```

也可以用invertMask属性让白色像素(而不是黑色像素)代表影片中可见的像素。一定要记住蒙版的套准点应设置在左上角，而不是在中心。

参见第6章“数字视频文件”里的6.3节“数字视频文件的设置”，可以获得在Director中使用数字视频文件的背景资料。

19.2 建立视频控制

视频属性的使用在前面已经阐述过了，创建一些自定义的控制是非常简单的。事实上，一种行为通常可以处理10种不同类型的控制。

下面的行为需要知道是哪个角色包含有数字视频文件。还需要知道所用的是哪种控制类型。

```
property pControlType, pVideoSprite

on getPropertyDescriptionList me
  list = [:]
  addProp list, #pControlType, [#comment: "Control",¬
    #format: #symbol,¬
    #range: [#play, #stop, #pause, #stepForward, #stepBackward, ¬
      #start, #reverse, #fastForward, #fastReverse, #end, #loop], ¬
    #default: #stop]
  addProp list, #pVideoSprite, [#comment: "Video Sprite",¬
    #format: #sprite, #default: 1]
  return list
end
```

尽管更复杂的行为可能包括按下按钮操作这一类的处理程序，如按钮的“按下”状态和“掠过”状态等，但这个行为只做了控制视频角色时的必要工作。

对于play按钮来说，所需要做的只是将movieRate设置为1。

```
on mouseUp me
  case pControlType of
    #play:
      sprite(pVideoSprite).movieRate = 1
```

相反，对于Stop按钮来说，需要把movieRate设置为0，对于pause也是一样。为了使play按钮有所不同，也可以将 movieTime 设置为0，这样就可以停止播放，并把视频文件向回转，而不仅是停止播放。

```
    #stop:
      sprite(pVideoSprite).movieRate = 0
      sprite(pVideoSprite).movieTime = 0
    #pause:
      sprite(pVideoSprite).movieRate = 0
```

start按钮或end按钮可以使影片回到开头或结尾，并使它暂停。请注意，这意味着 start按钮及stop按钮实际上起到同样的作用。

```
    #start:
      sprite(pVideoSprite).movieRate = 0
      sprite(pVideoSprite).movieTime = 0
    #end:
      sprite(pVideoSprite).movieRate = 0
      sprite(pVideoSprite).movieTime = sprite(pVideoSprite).duration
```

reverse按钮以正常的速度播放，但方向是向回。

```
    #reverse:
    sprite (pVideoSprite). movie Rate = -1
```

有两种类型的step按钮：Forward(向前)及Backward(向回)。如果把视频文件的速度设置为典型的15帧/秒,这就意味着每帧播放的时间大约为 4/60秒或4 tick。

```
    #stepForward:
      sprite(pVideoSprite).movieTime = sprite(pVideoSprite).movieTime + 4
    #stepBackward:
      sprite(pVideoSprite).movieTime = sprite(pVideoSprite).movieTime - 4
```

fast forward及fast reverse按钮可以使得正向或反向播放的速度大于正常速度。在本例中采用了3。

```
    #fastForward:
      sprite(pVideoSprite).movieRate = 3
    #fastReverse:
      sprite(pVideoSprite).movieRate = -3
```

最后一类按钮是loop开关。它确定影片放完之后是否需要循环播放。一种好办法是把它制作为一个独立的行为，并采用一些复选框行为的程序代码，因而此按钮可以有“循环状态”和“非循环状态”。然而，为了简化程序，这个按钮只是起到切换 loop属性的作用，并不给使用者提供反馈信息。

```
    #loop:
      sprite(pVideoSprite).member.loop = ¬
        not sprite(pVideoSprite).member.loop
    end case
end
```

还可以制作其他视频控制按钮。例如，我们可以用 loopBounds在同一视频文件中切换到不同的循环；滑动条可以用来控制音量。甚至可以用滑动条设置 movieTime属性，这样的滑动条的作用就像QickTime的缺省控制条一样，但在这里我们可以使用一些自己定义的图形。

参见第6章里的6.3节“数字视频文件的设置”，可以获得更多关于数字视频属性的信息。

19.3 其他视频技术的使用

这些视频属性还可以用来制作角色表演技巧。例如，下面的行为采用了角色的 blend属性，将使视频角色渐渐显示。

```
property pSpeed

on getPropertyDescriptionList me
  list = [:]
  addProp list, #pSpeed, [#comment: "Speed", #format: #integer, ¬
    #range: [#min: 1, #max: 20], #default: 7]
  return list
end

on beginSprite me
  sprite(me.spriteNum).member.directToStage = FALSE
  sprite(me.spriteNum).member.crop = TRUE

  sprite(me.spriteNum).blend = 0
end

on exitFrame me
  if sprite(me.spriteNum).blend < 100 then
    sprite(me.spriteNum).blend = ¬
      min(sprite(me.spriteNum).blend+pSpeed,100)
  end if
```

```
end
```

请注意，这个处理程序也设置了视频角色的 directToStage和crop属性。尽管通常不必要这样，但它确保了其他行为对于这些属性的任何改变都将不起作用。

相反也是同样的，下面的行为将一个角色渐渐淡化至 blend值为0。

```
property pSpeed

on getPropertyDescriptionList me
  list = [:]
  addProp list, #pSpeed, [#comment: "Speed", #format: #integer,¬
    #range: [#min: 1, #max: 20], #default: 7]
  return list
end

on beginSprite me
  sprite(me.spriteNum).member.directToStage = FALSE
  sprite(me.spriteNum).member.crop = TRUE

  sprite(me.spriteNum) .blend = 100
end

on exitFrame me
  if sprite(me.spriteNum).blend > 0 then
    sprite(me.spriteNum).blend = ¬
      max(sprite(me.spriteNum).blend-pSpeed,0)
  end if
end
```

当然，这些行为可以被合并，以提供其中任何一种功能。注意它们也可以用于位图，甚至也可以用于文本演员。

为了实现更复杂的效果，下面的行为可以使视频图像从一个小点开始，沿着横向生长，直至成为一条线，然后向下生长，直至变成视频图像的本来面目。结果看起来像打开一台老式的电视机时的效果。

```
property pOrigRect, pSpeed

on getPropertyDescriptionList me
  list = [:]
  addProp list, #pSpeed, [#comment: "Speed", #format: #integer,¬
    #range: [#min: 1, #max: 20], #default: 7]
  return list
end

on beginSprite me
  sprite(me.spriteNum).member.directToStage = TRUE
  sprite(me.spriteNum).member.crop = FALSE

  pOrigRect = sprite(me.spriteNum).rect

  -- set rect to center point
  x = pOrigRect.left+(pOrigRect.width/2)
  y = pOrigRect.top+(pOrigRect.height/2)
  r = rect(x,y,x,y+1)
  sprite(me.spriteNum).rect = r
```

```
end

on exitFrame me
  if sprite(me.spriteNum).rect.width < pOrigRect.width then
    r = sprite(me.spriteNum).rect
    r.left = max(r.left-pSpeed, pOrigRect.left)
    r.right = min(r.right+pSpeed, pOrigRect.right)
    sprite(me.spriteNum).rect = r

  else if sprite(me.spriteNum).rect.height < pOrigRect.height then
    r = sprite(me.spriteNum).rect
    r.top = max(r.top-pSpeed, pOrigRect.top)
    r.bottom = min(r.bottom+pSpeed, pOrigRect.bottom)
    sprite(me.spriteNum).rect = r

  end if
end
```

数字视频文件还可以被旋转。下面一个行为可能没有什么实用价值，但却是一个很好的演示。

```
property pSpeed

on getPropertyDescriptionList me
  list = [:]
  addProp list, #pSpeed, [#comment: "Speed", #format: #integer, ¬
    #range: [#min: 1, #max: 20], #default: 7]
  return list
end

on beginSprite me
  sprite(me.spriteNum).member.directToStage = TRUE
  sprite(me.spriteNum).member.crop = TRUE
end

on endSprite me
  sprite(me.spriteNum).rotation = 0
end

on exitFrame me
  sprite(me.spriteNum).rotation = ¬
      sprite(me.spriteNum).rotation + pSpeed
end
```

下面还有一个更与众不同的效果。下面的行为将图像收缩，然后将其放在屏幕的左边。然后打开角色的trails属性，这样图像就会被留在所经过之处。然后向右移，留下另一幅图像。继续做这个动作，把一幅幅图像留在后面，结果看起来有些像电影胶片。当它到达屏幕右边时，又开始替换左边的图像。

```
property pOrigRect, pSize, pSpacing, pStart, pDirect

on getPropertyDescriptionList me
  list = [:]
  addProp list, #pSize, [#comment: "Size (%)", #format: #integer, ¬
    #range: [#min: 5, #max: 100], #default: 25]
  addProp list, #pSpacing, [#comment: "Spacing", #format: #integer, ¬
```

```
    #range: [#min: 0, #max: 25], #default: 5]
  addProp list, #pStart, [#comment: "Start X", #format: #integer,¬
    #default: 0]
  addProp list, #pDirect,  [#comment: "Direct To Stage", #format: #boolean,¬
    #default: TRUE]
  return list
end

on beginSprite me
  sprite(me.spriteNum).member.directToStage = pDirect
  sprite(me.spriteNum).member.crop = FALSE
   sprite(me.spriteNum).trails = FALSE

  pOrigRect = sprite(me.spriteNum).rect
  r = sprite(me.spriteNum).rect
  r = (r*pSize)/100.0
  sprite(me.spriteNum).rect = r
  sprite(me.spriteNum).locH = pStart
  sprite(me.spriteNum).trails = TRUE
end

on endSprite me
  sprite(me.spriteNum).trails = FALSE
  sprite(me.spriteNum).rect = pOrigRect
end

on exitFrame me
  x = sprite(me.spriteNum).locH
  x = x + sprite(me.spriteNum).rect.width + pSpacing

  if x > the stageRight then x = pStart

  sprite(me.spriteNum).locH = x
end
```

图19-2显示了这段程序运行时的效果。

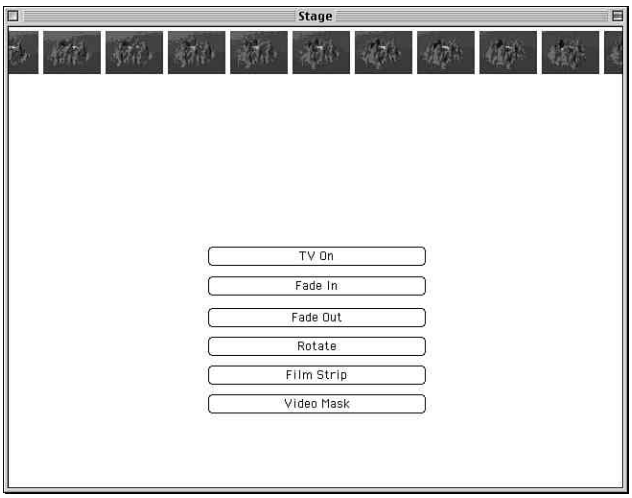

图19-2 “电影胶片”行为将视频文件的图像放置在舞台的不同位置

这些只是一些用数字视频演员和角色属性完成的简单例子。你可以根据你自己需要将它们组合并扩展，反复试验就会找到最佳方法。

参见第6章里的6.4节“处理数字视频文件”，可以获得关于使用数字视频的更多信息。

19.4 视频文件的故障排除

如果采用内置的Quicktime控制条，要检查它们在Mac和Windows上的尺寸；它们可能有一些差别，这取决于QuickTime的版本。QuickTime 2.1的Windows的控制条比Mac的要高一些。

19.5 你知道吗

■ 我们可以用视频控制剧本来控制仅有MIDI或仅有声音的QuickTime视频。这样，我们可以建立一个使用MIDI或超压缩的QuickTime音频的自动唱片点唱机。

■ 当设置movieRate属性为负值时，不仅影片会向回播放，而且声音也将向回播放。