**运行脚本**

要运行一个现有的脚本文件，可以在Max脚本的实用程序卷栏中按下运行脚本按钮，或者在监听菜单栏中找到文件>运行脚本，或者在3ds Max菜单栏中找到*MAXScript>*运行脚本。

这将打开一个用于选择脚本的文件打开对话框。然后，MAXScript将读取并执行所选择的脚本，任何输出都会打印到监听器输出窗口。

你也可以使用filein()方法从监听器或其他脚本中运行脚本：

*fileIn* <filename\_string> [ *quiet:*<boolean> ]

<filename\_string>是一个字符串文本或者一个用于计算字符串的表达式，指定了脚本文件的名称，该文件的内容将被执行。脚本文件每次执行一次，如果在任何时候遇到错误就停止处理。默认情况下，该文件未在加载时列出；

use *quiet:*值为false时将一个正在运行脚本的清单传给监听器。

|  |
| --- |
| 例用法： |
| fileIn "my\_script.ms" |
| or |
| scriptfile="my\_script.ms"fileIn scriptfile |

脚本文件内容是在全局范文上下文中编译的，而不是在执行filein()方法时的作用域。

**在脚本中包含脚本**

MAXScript提供一个编译时源文件包含机制，允许你将大型脚本分解成更小的脚本，这些文件可以包含在脚本的任何位置，你可以在代码的任何位置使用 *include* <file>，以便在代码中有效的插入文件的内容。

形式如下：

*include* "filename\_string"

这是一个编译时构造，因此文件名规范必须是字符串文字，而不能是一个变量或表达式。

|  |
| --- |
| *EXAMPLE* |
| utility foo "Baz"  (  local a, b, c  include "foo-ui.ms"  rollout bar "Bar"  (  include "bar-rollout.ms"  )  include "foo-handlers.ms"  ) |

在代码中的该位置，*include* <file>结构被有效的替换为了以此命名的脚本文件内容。

你可以根据需要嵌套文件，包含的文件也可以包含其他文件，等等。

因为*include*是一个编译时构造，所以当前的MAXScript作用域是由包含的文件影响的。这与运行脚本中描述的*fileIn()*方法相比较，后者的脚本内容是在全局范围上下文中编译的。

*include* <file>可以在任何位置出现一个预期内容的替代记号，例如一个名字，字符串或者标点符号。这以为这你可以使用包含的文件来完成部分表达式，

|  |
| --- |
| *FOR EXAMPLE* |
| include "op1.ms"+ include "op2.ms"  if include "test2.ms" then print a else print b |

你不能在一个引用中放置*include* <file>，例如，以下是无效的：

|  |
| --- |
| *NOT VALID:* |
| timeval = 2:include "framenum.ms".0 |

**软件工具开发包和MAXScript**

*3ds Max*提供了两种编写插件的方法：使用c++的sdk，以及程序的本地脚本语言MAXScript。你选择哪种语言部分取决于你希望如何工作，部分取决于你希望你的插件来完成的工作是什么。这两种语言都有它们的优点和局限性，但是你可以使用它们中的任何一种来开发复杂的应用程序。

**MAXScript**

一般来说，MAXScript插件运行的速度比用c++编写的同类插件要慢，所以如果性能是一个问题，那么使用sdk可能更好。

另一方面，MAXScript提供了一些比c++sdk中更高级别的方法，并且支持一些没有暴露于sdk的3dsMax特性和功能。如果你的特性需要由MAXScript支持的功能，而sdk不支持，那么MAXScript是你唯一的选择。特别是，通过ole/activex/dotnet控件暴露的3ds Max的特性，在MAXScript中比在SDK中更容易编码。

MAXScript可以用于原型插件的开发，开发相对较小的特性，以及编写测试套件。