

# 3ds MAXScript 脚本语言

## 完全学习手册

王 华 编著

兵器工业出版社

北京科海电子出版社

## 内 容 简 介

MAXScript 语言是 3ds max 自带的脚本语言，它不但拥有一般程序语言的所有特点，而且可以在程序内实现几乎所有在 3ds max 界面下的交互操作，功能非常强大，而且语法比较简单，容易上手。

本书详细、系统地介绍了 MAXScript 语言。全书分成 4 部分，共 23 章。第 1 部分详细介绍了 MAXScript 语言的基础知识、数据类型、变量、表达式、函数及程序流程控制等语法方面的基础知识；第 2 部分讲述了 MAXScript 语言如何创建和修改模型、如何对材质以及动画进行控制；第 3 部分详述了用 MAXScript 语言创建实用工具、用户界面等方面的知识；第 4 部分讲述了如何在脚本程序里控制 3ds max 用户界面、如何暂停脚本执行、如何控制渲染器、如何执行外部命令或程序以及如何退出和重置 3ds max 系统等方面的问题。

本书的最大特点是用实例程序对 MAXScript 脚本语言的功能进行演示，实用性强，特别适合有一定 3ds max 使用基础的读者阅读，对于专业动画创作人员，本书也有很高的参考价值，可作为工具书随用随查。

## 图书在版编目（CIP）数据

3ds MAXScript 脚本语言完全学习手册/王华编著. —北京：兵器工业出版社；北京科海电子出版社，2006.9  
ISBN 7-80172-733-9

I . M... II . 王... III . 三维—动画—图形软件，  
3ds max—程序设计—手册 IV . TP391.41-62

中国版本图书馆 CIP 数据核字（2006）第 095337 号

出版发行：兵器工业出版社 北京科海电子出版社

封面设计：林 陶

邮编社址：100089 北京市海淀区车道沟 10 号

责任编辑：李翠兰 陈 洁

100085 北京市海淀区上地七街国际创业园 2 号楼 14 层

责任校对：刘雪莲

www.khp.com.cn

印 数：1-3000

电 话：(010) 82896442 62630320

开 本：787×1092 1/16

经 销：各地新华书店

印 张：54.75

印 刷：北京科普瑞印刷有限责任公司

字 数：1332 千字

版 次：2006 年 9 月第 1 版第 1 次印刷

定 价：98.00 元

（版权所有 翻印必究 印装有误 负责调换）

# 前　　言

## 使用 MAXScript 脚本语言的原因

3ds max 是目前国内动画行业使用最普遍的、功能强大的三维动画设计软件，它除了具有交互操作方式外，还提供了编程开发工具——MAXScript 脚本语言。MAXScript 是 3ds max 众多插件中最有特色的一个。

MAXScript 脚本语言具有强大的功能，主要表现在以下几个方面：

- MAXScript 是一种面向对象的程序设计语言，没有很严格的格式要求，易于掌握。
- MAXScript 不但拥有一般程序设计语言的所有特点，而且几乎可以在程序内实现所有在 3ds max 界面下的交互操作，覆盖了 3ds max 的绝大部分功能，如可进行建模、动画设置、材质调制、灯光设置、渲染处理等。
- 可以建立批处理工具，这样可以把 3ds max 从业人员从一些简单重复的工作中解脱出来。例如建立一次可以渲染多个场景文件的 MAXScript 脚本语言程序。
- 用 MAXScript 进行动画设计是动画创作的较高层次，由 MAXScript 生成的动画往往有出神入化、令人惊叹的效果。例如，用 MAXScript 可以轻松地模拟有上千个球体的星系，可以生成逼真的群鸟飞翔的动画，可以使物体按某个特定的数学曲线运动，而这些恰恰是传统的交互操作方式很难完成的。
- 可以定制自己的卷展栏（Rollout）界面、鼠标工具、插件、工具按钮等。

## 本书的写作目的

作者具有十余年的 3ds max 使用经验，因为工作需要，早在五六年前就开始了对 MAXScript 脚本语言的研究，随着对其了解得越深入，就越发现其魅力无穷，同时作者的职场生涯也因 MAXScript 而受益匪浅。许多 3ds max 中文用户也很想对 MAXScript 脚本语言有系统的了解，但比较普遍的情况是：英文程度不够，直接阅读联机帮助都比较有难度；另外还有一点就是从事动画行业的人编程水平普遍都不高，所以往往半途而废。

一直以来，虽然市面上有关 3ds max 的专业书籍多如牛毛，却只有个别的书里面会有关于 MAXScript 脚本语言的比较简单的介绍，但常常都是蜻蜓点水，一带而过；网络上也仅能见到关于 MAXScript 语言某一方面功能的零星文章。对于 3ds max 的众多插件中最有特色，也是功能最强大的 MAXScript 脚本语言却没有一本完全的教材！

因此我萌生了写作本书的想法，希望通过本书的出版，完全改变 3ds max 中文用户学习 MAXScript 脚本语言的现状，让国内的 3ds max 同行真正从日常重复的工作中解放出来，轻松实现对三维建模、动画的精确控制，从而为 3ds max 的中文用户找到一条全面提升 3ds max 专业水准的捷径。我相信在本书之后将会有越来越多的 3ds max 同行开始这方面的研究，将有更多的 3ds max 同行从中受益。

## 本书的读者对象

本书实用性很强，特别适合有一定 3ds max 使用基础的读者阅读，对专业动画创作人员也有很高的参考价值。由于 MAXScript 脚本语言没有严格的格式要求且易于掌握，所以对于没有多少编程语言基础的读者亦非常适合。

## 本书的主要内容

本书从最基本的内容讲起，对 MAXScript 脚本语言的强大功能做了详细、系统、全面的介绍。本书无论对 MAXScript 脚本语言的初学者还是对有使用经验的读者都具有一定的学习和参考价值。

全书分成 4 部分，共 23 章。下面详细介绍各章的内容。

### 第 1 部分——MAXScript 语法基础

第 1 章“了解 MAXScript”讲述了如何开始 MAXScript。对新建、编辑、运行脚本程序及与 MAXScript 语言有关的两个窗口都有详细的讲解，最后通过一个简单的例子来引导读者编写简单的 MAXScript 脚本语言。通过本章，读者可以对脚本语言有一个初步的印象。

第 2 章“MAXScript 语言基础”主要介绍了关于 MAXScript 语言脚本源程序编写的一些基础知识，指明了读者需要预先掌握的知识。对一些基本概念如“字面常量”、“标识符”、“保留关键字”等做了详细介绍；本章还讲述了 MAXScript 脚本语言的表达式和赋值语句，以及语法定义的格式说明等。

第 3 章“MAXScript 数据类型”讲述了 MAXScript 脚本语言的数据的操作符和方法，并详细讲解了各种数据类型。

第 4 章“MAXScript 语言的变量和表达式”介绍了变量的赋值、求值顺序、局部变量和全局变量以及表达式等方面的内容。

第 5 章“控制 MAXScript 程序流程”介绍了程序流程控制，其中包括 if、case、while、do、for、continue、exit、try 语句。

第 6 章“MAXScript 自定义函数”集中讨论了函数，包括如何创建自定义函数和调用函数、如何向函数传递参数以及如何从函数返回值等。

### 第 2 部分——用 MAXScript 操作场景对象

第 7 章“对象超级类 MAXWrapper”介绍了 MAXWrapper 的通用属性和方法及其扩展数据。

第 8 章“创建 Node（节点）对象”详细介绍了所有场景对象的属性以及创建方法。这是 MAXScript 脚本语言在“建模”方面的应用。

第 9 章“Editable\_Mesh、SplineShape、Patch 和 Editable\_Poly”对 Editable\_Mesh、SplineShape、Patch、Editable\_Poly 的构造函数、操作符、属性、方法等方面做了详细介绍。

第 10 章“Modifier（对象空间修改器）和 Spacewarp（世界空间修改器）”讲解了 MAXScript 脚本语言在修改模型方面的应用，即对各种对象空间修改器和世界空间修改器做了详细介绍。

第 11 章“Material（材质）和 TextureMap（贴图）”讲述了如何使用 MAXScript 脚本语言对各种类型的材质以及其贴图进行调整。

第 12 章“动画控制器”详述了各种动画控制器的属性及其与控制有关的方法。这是 MAXScript 脚本语言在精确控制动画方面的应用。

第 13 章“Atmospheric（环境效果）”讲解了各种环境效果的属性和方法。

第 14 章“RenderEffect（渲染效果）”详细介绍了各种类型渲染效果的属性和方法。

### 第 3 部分——用 MAXScript 创建实用工具、用户界面

第 15 章“创建脚本工具程序 Utility”介绍了如何定制和定义脚本工具程序 Utility，对 Utility、Rollout 以及各种控件、图像按钮等做了详细介绍。

第 16 章“RcMenu（右键菜单）”讲解了 RcMenu 子句、用户界面控件、菜单项、分隔行、子菜单。

第 17 章“宏脚本（MacroScript）”讲解了宏脚本的定义和图标位图文件的创建。

第 18 章“脚本鼠标工具”介绍了脚本鼠标工具定义和 MouseTool 子句。

第 19 章“脚本插件”讲解了各类脚本插件，包括 Shape 类、Light 类、Modifier 类、Material 类、TextureMap 类等脚本插件。还介绍了 Plug-in 子句以及如何定制用户界面卷展栏等问题。

### 第 4 部分——MAXScript 的高级应用

第 20 章“在 MAXScript 里与用户界面交互”讲述如何在脚本程序里控制 3ds max 用户界面，包括按下命令按钮、打开和关闭触发器以及打开对话框等。

第 21 章“在 MAXScript 里存取文件”介绍了场景文件的装载、保存以及脚本文件的加密等方面的问题。

第 22 章“事件侦测和信号反馈机制”讲解了事件侦测和 when 构造函数、时间改变信号反馈机制、视窗刷新信号反馈机制以及通用事件反馈机制。

第 23 章“MAXScript 杂项函数”介绍了如何暂停脚本执行、如何控制渲染器、如何执行外部命令或程序、如何使用时间计算函数以及如何退出和重置 3ds max 系统等。

## 如何使用本书

不管何种学习，模仿都是最行之有效的方法。在讲述 MAXScript 脚本语言的过程中提供了一系列的实例程序，每个程序对 MAXScript 脚本语言的某个特定的功能做了例证，对学习和掌握 MAXScript 脚本语言很实用。读者应该认真阅读这些实例，并以此为参照，写出自己需要的脚本程序。

本书亦可以作为工具书，以备日常使用中查找需要了解的内容，这一点可以通过详细的目录轻松实现。其实阅读本书，也是对 3ds max 从另一个角度上的全面了解，本书几乎涉及 3ds max 的方方面面。

因作者水平有限，书中错误在所难免，读者在阅读本书时有任何问题或不同见解，请不吝赐教！作者的 E-mail：wanghuawh@hotmail.com。

最后，特别感谢刘立平先生对本书写作所作出的重大贡献。

编者

# 目 录

## 第 1 部分 MAXScript 语法基础

第 1 章 了解 MAXScript.....	2
1.1 如何开始 MAXScript .....	2
1.2 如何新建、编辑、运行脚本文件 .....	3
1.3 在 3ds max 开始运行时加载所需的脚本文件 .....	4
1.4 MAXScript Listener 窗口 .....	6
1.4.1 MAXScript Listener 窗口的功能及特点 .....	6
1.4.2 MAXScript Listener 命令 .....	7
1.4.3 宏记录器 (MacroRecorder) .....	8
1.4.4 Listener 日志文件 .....	9
1.5 MAXScript Editor 窗口 .....	10
1.5.1 MAXScript Editor 窗口功能及特点 .....	10
1.5.2 MAXScript Editor 窗口的菜单命令 .....	11
1.6 MAXScript 桌面状态 .....	12
1.7 快速学习 MAXScript 语言的两个方法.....	13
1.8 一个简单的 MAXScript 例子 .....	13
1.8.1 用 MAXScript 创建一个简单的 Box 对象 .....	13
1.8.2 修改 Box 对象.....	14
1.8.3 Box 对象的标准转换.....	17
1.8.4 Box 对象的更多转换.....	19
1.8.5 为 Box 对象创建动画.....	19
第 2 章 MAXScript 语言基础.....	21
2.1 脚本文件里命令的求值 .....	21
2.2 把脚本文件包含在另一个脚本文件中.....	21
2.3 向 MAXScript 里输入数据信息.....	22
2.4 使用“? ”号 .....	22
2.5 脚本运行过程的错误信息 .....	23
2.6 用 Esc 键中断程序运行 .....	23
2.7 在 MAXScript 中使用 3ds max 命令 .....	24
2.8 语法定义的格式说明 .....	31

2.9 MAXScript 里的数学运算 .....	32
2.10 源代码布局规则与注释 .....	33
2.11 赋值 .....	33
2.12 属性、方法、操作符、字面常量 .....	34
2.13 标识符 .....	34
2.14 保留关键字、标点、符号 .....	36
2.14.1 保留关键字 .....	36
2.14.2 标点、符号 .....	37
<b>第 3 章 MAXScript 数据类型 .....</b>	<b>38</b>
3.1 数据的操作符和方法 .....	38
3.1.1 操作符 .....	38
3.1.2 方法 .....	38
3.2 基本数据类型 .....	40
3.2.1 Number .....	40
3.2.2 String .....	43
3.2.3 Name .....	47
3.2.4 BooleanClass .....	48
3.2.5 Color .....	49
3.2.6 Point3 .....	50
3.2.7 Point2 .....	53
3.2.8 Ray .....	54
3.2.9 Quat .....	54
3.2.10 AngleAxis .....	57
3.2.11 EulerAngles .....	58
3.2.12 Matrix3 .....	59
3.2.13 BigMatrix .....	61
3.2.14 Box2 .....	63
3.2.15 BitArray .....	64
3.2.16 Time .....	65
3.2.17 Interval .....	66
3.2.18 Bitmap .....	66
3.2.19 Stream .....	73
3.3 特殊数据类型 .....	76
3.3.1 undefined 类 .....	76
3.3.2 OK .....	76
3.3.3 unsupplied .....	76
3.3.4 DontCollect .....	77
3.4 高级数据类型 .....	77

3.4.1 Structure (结构) .....	77
3.4.2 MaxKey 类 .....	80
3.4.3 NoteTrack .....	82
3.4.4 Collection (集合类数据) .....	83
3.5 集合类数据类型 .....	84
3.5.1 Array (数组) .....	84
3.5.2 PathName (路径名) .....	86
3.5.3 ObjectSet (对象集) .....	89
3.5.4 SelectionSet (选择集) .....	90
3.5.5 SelectionSetArray (选择集数组) .....	90
3.5.6 NodeChildrenArray (子对象数组) .....	92
3.5.7 VertexSelection (顶点选择集) .....	92
3.5.8 FaceSelection (面选择集) .....	94
3.5.9 EdgeSelection (边选择集) .....	95
3.5.10 MaxKeyArray (关键帧数组) .....	97
3.5.11 MaxNoteKeyArray (Note 轨迹关键帧数组) .....	98
3.5.12 ModifierArray (修改器数组) .....	99
3.5.13 MaterialLibrary (材质库) .....	99
3.5.14 ArrayParameter (数组参数类) .....	101
<b>第 4 章 MAXScript 语言的变量和表达式 .....</b>	<b>103</b>
4.1 变量赋值 .....	103
4.2 变量的求值顺序 .....	104
4.3 变量的作用域 .....	104
4.4 局部变量和全局变量 .....	108
4.5 保留全局变量 .....	110
4.5.1 预定义全局变量 .....	110
4.5.2 3ds max 系统变量 .....	110
4.5.3 MAXScript 系统变量 .....	115
4.6 持续型全局变量 .....	116
4.7 变量的几个特性 .....	117
4.8 表达式 .....	119
4.9 简单表达式 .....	120
4.9.1 数学表达式 .....	120
4.9.2 比较表达式 .....	121
4.9.3 逻辑表达式 .....	121
4.9.4 函数调用表达式 .....	121
4.9.5 块表达式 .....	122
4.10 关联表达式 .....	122

4.10.1	animate.....	123
4.10.2	at level、in.....	123
4.10.3	at time .....	124
4.10.4	coordsys .....	124
4.10.5	about .....	125
4.10.6	undo .....	125
4.10.7	关联语句的迭加 .....	126
4.10.8	关联语句嵌套 .....	126
4.10.9	持续关联语句 .....	127
<b>第 5 章</b>	<b>控制 MAXScript 程序流程.....</b>	<b>128</b>
5.1	if 表达式 .....	128
5.2	case 表达式 .....	128
5.3	while 循环和 do 循环 .....	129
5.4	for 循环 .....	130
5.5	continue 语句 .....	131
5.6	exit 语句 .....	131
5.7	try 表达式 .....	132
<b>第 6 章</b>	<b>MAXScript 自定义函数 .....</b>	<b>133</b>
6.1	创建自定义函数 .....	133
6.2	函数变量 .....	135
6.3	函数参数 .....	136
6.4	return 表达式 .....	138
6.5	函数调用的优先级 .....	138

## 第 2 部分 用 MAXScript 操作场景对象

<b>第 7 章</b>	<b>对象超级类 MAXWrapper.....</b>	<b>140</b>
7.1	MAXWrapper 的通用属性和方法 .....	140
7.1.1	MAXWrapper 值和类的通用属性.....	140
7.1.2	MAXWrapper 值和类的通用方法.....	141
7.2	MAXWrapper 的扩展数据 .....	143
<b>第 8 章</b>	<b>创建 Node（节点）对象.....</b>	<b>145</b>
8.1	Node 类构造函数 .....	145
8.2	Node 类方法 .....	146
8.2.1	通用方法 .....	146
8.2.2	与渲染有关的方法 .....	149

8.2.3 与组 (Group) 有关的方法 .....	150
8.2.4 与视窗状态有关的方法.....	151
8.2.5 与对象选择有关的方法.....	152
8.2.6 与修改器堆栈 (Modifier Stack) 有关的方法.....	152
8.2.7 与对象 Modifier 关联转换有关的方法.....	153
8.2.8 与对象转换有关的方法.....	154
8.2.9 与用户定制属性有关的方法.....	155
8.2.10 与 IK 属性有关的方法 .....	156
8.2.11 Node 类其他方法 .....	157
8.3 Node 类对象属性 .....	157
8.3.1 Node 通用属性.....	157
8.3.2 与 Target/LookAt 有关的属性 .....	159
8.3.3 与视窗有关的属性 .....	159
8.3.4 与层有关的属性 .....	160
8.3.5 与渲染有关的属性 .....	161
8.3.6 与转换有关的属性 .....	162
8.3.7 使用 Node 对象的转换属性 .....	164
8.3.8 定制 Node 属性.....	168
8.4 Node 子类 .....	169
8.4.1 GeometryClass: Node .....	169
8.4.2 GeometryClass 的操作符和方法 .....	170
8.4.3 Geometry-Standard Primitives (标准基本体) .....	170
8.4.4 Geometry-Extended Primitives (扩展基本体) .....	176
8.4.5 Geometry-Dynamics Objects (动力学对象) .....	186
8.4.6 Geometry-Compound Objects (复合对象) .....	190
8.4.7 Geometry-Door 和 Window (门窗建筑对象) .....	198
8.4.8 Stair: GeometryClass (楼梯建筑对象) .....	203
8.4.9 Geometry-Patch Objects (面片栅格对象) .....	211
8.4.10 Geometry-Particle Systems (粒子系统) .....	211
8.4.11 Geometry-NURBS Objects (NURBS 对象) .....	236
8.5 Shape: Node .....	237
8.5.1 Shape 类方法.....	237
8.5.2 Shape-Spline (样条曲线) .....	239
8.5.3 Spline 类 Shape 对象通用属性和方法 .....	239
8.5.4 NURBS 曲线 .....	251
8.6 Light: Node.....	251
8.6.1 Light 通用属性、操作符和方法 .....	252
8.7 Camera: Node.....	263

8.7.1 Camera 通用属性 .....	263
8.8 Helper: Node .....	265
8.8.1 Bone: Helper (骨骼系统) .....	266
8.8.2 Compass: Helper (指南针辅助对象) .....	266
8.8.3 Dummy: Helper (虚拟辅助对象) .....	267
8.8.4 Grid: Helper (栅格辅助对象) .....	267
8.8.5 Point: Helper (点辅助对象) .....	267
8.8.6 Protractor: Helper (量角器辅助对象) .....	268
8.8.7 Tape: Helper (卷尺辅助对象) .....	268
8.8.8 Helper-Atmospheric (大气装置) .....	268
8.8.9 Helper-Camera Match (摄影机匹配) .....	269
8.8.10 Helper-VRML 1.0/VRBL .....	270
8.8.11 Anchor: Helper .....	270
8.8.12 AudioClip: Helper (锚定 VRML97 辅助对象) .....	271
8.8.13 Background: Helper (背景 VRML97 辅助对象) .....	271
8.8.14 Billboard: Helper (布告牌 VRML97 辅助对象) .....	271
8.8.15 FogHelper: Helper (雾 VRML97 辅助对象) .....	271
8.8.16 InlineHelper: Helper (内嵌 VRML97 辅助对象) .....	271
8.8.17 LODHelper: Helper (LOD VRML97 辅助对象) .....	272
8.8.18 NavInfo: Helper (漫游信息 VRML97 辅助对象) .....	272
8.8.19 ProxSensor: Helper (范围感应器 VRML97 辅助对象) .....	272
8.8.20 Sound: Helper (音频剪辑 VRML97 辅助对象) .....	272
8.8.21 TimeSensor: Helper (时间感应器 VRML97 辅助对象) .....	273
8.8.22 TouchSensor: Helper (触动感应器 VRML97 辅助对象) .....	273
8.9 System: Node (系统) .....	273
8.9.1 Bones: System (骨骼系统) .....	273
8.9.2 Sunlight: System (太阳光系统) .....	274
8.9.3 RingArray: System (环形阵列系统) .....	274
8.10 SpacewarpObject: Node (空间扭曲) .....	274
8.10.1 Bomb: SpacewarpObject (爆炸空间扭曲) .....	275
8.10.2 ConformSpaceWarp: SpacewarpObject (一致空间扭曲) .....	276
8.10.3 SpaceDisplace: SpacewarpObject (位移空间扭曲) .....	277
8.10.4 SpaceFFDBox: SpacewarpObject (FFD 长方体空间扭曲) .....	278
8.10.5 SpaceFFDCyl: SpacewarpObject (FFD 柱体空间扭曲) .....	279
8.10.6 SpaceRipple: SpacewarpObject (涟漪空间扭曲) .....	280
8.10.7 SpaceWave: SpacewarpObject (波浪空间扭曲) .....	281
8.10.8 Gravity: SpacewarpObject (重力空间扭曲) .....	282
8.10.9 Motor: SpacewarpObject (马达空间扭曲) .....	282

8.10.10	PBomb: SpacewarpObject (粒子爆炸空间扭曲) .....	284
8.10.11	PushSpaceWarp: SpacewarpObject (推力空间扭曲) .....	285
8.10.12	Wind: SpacewarpObject (风力空间扭曲) .....	286
8.10.13	SpaceBend: SpacewarpObject (弯曲修改器) .....	287
8.10.14	SpaceNoise: SpacewarpObject (噪波修改器) .....	288
8.10.15	SpaceSkew: SpacewarpObject (倾斜修改器) .....	289
8.10.16	SpaceStretch: SpacewarpObject (挤出修改器) .....	289
8.10.17	SpaceTaper: SpacewarpObject (锥化修改器) .....	290
8.10.18	SpaceTwist: SpacewarpObject (扭曲修改器) .....	291
8.10.19	PDynaFlect: SpacewarpObject (动力学导向板空间扭曲) .....	292
8.10.20	SDynaFlect: SpacewarpObject (动力学导向球空间扭曲) .....	293
8.10.21	UDynaFlect: SpacewarpObject (通用动力学导向器空间扭曲) .....	294
8.10.22	Deflector: SpacewarpObject (导向器空间扭曲) .....	295
8.10.23	Path_Follow: SpacewarpObject (路径跟随空间扭曲) .....	296
8.10.24	POmniFlect: SpacewarpObject (泛方向导向板空间扭曲) .....	297
8.10.25	SDeflector: SpacewarpObject (导向球空间扭曲) .....	299
8.10.26	SOmniFlect: SpacewarpObject (泛方向导向球空间扭曲) .....	300
8.10.27	UDeflector: SpacewarpObject (通用导向器空间扭曲) .....	301
8.10.28	UOmniFlect: SpacewarpObject (通用泛方向导向器空间扭曲) .....	302
8.11	XRefObject: Node (外部参照对象) .....	304
8.11.1	XRefScene Values.....	305
8.12	Track View Node (轨迹视窗节点) .....	307
8.13	NURBS Node 属性和方法 .....	309
8.13.1	NURBS 类 .....	311
8.13.2	NURBSCurveshape: Shape.....	313
8.13.3	NURBSObject 通用属性.....	313
8.13.4	NURBSPoint: NURBSObject (点子对象) .....	314
8.13.5	NURBSCurveConstPoint: NURBSPoint (曲线点) .....	314
8.13.6	NURBSCurveIntersectPoint: NURBSPoint (曲线-曲线相交点) .....	314
8.13.7	NURBSCurveSurfaceIntersectPoint: NURBSPoint (曲面-曲线相交点) .....	315
8.13.8	NURBSIndependentPoint: NURBSPoint (独立点) .....	315
8.13.9	NURBSPointConstPoint: NURBSPoint (偏移点) .....	316
8.13.10	NURBSSurfConstPoint: NURBSPoint (曲面点) .....	316
8.13.11	NURBSControlVertex: NURBSObject (控制顶点对象) .....	317
8.13.12	NURBSCurve: NURBSObject (曲线子对象) .....	317
8.13.13	NURBSBlendCurve: NURBSCurve (混合曲线) .....	318
8.13.14	NURBSChamferCurve: NURBSCurve (切角曲线) .....	318
8.13.15	NURBSCVCurve: NURBSCurve (CV 曲线子对象) .....	319

8.13.16	NURBSCurveOnSurface: NURBSCVCurve (曲面上的 CV 曲线) .....	320
8.13.17	NURBSFilletCurve: NURBSCurve (圆角曲线) .....	320
8.13.18	NURBSIsoCurve: NURBSCurve (U 向和 V 向等参曲线) .....	321
8.13.19	NURBSMirrorCurve: NURBSCurve (镜像曲线) .....	321
8.13.20	NURBSOffsetCurve: NURBSCurve (偏移曲线) .....	322
8.13.21	NURBSPointCurve: NURBSCurve (点曲线子对象) .....	322
8.13.22	NURBSPointCurveOnSurface: NURBSPointCurve (曲面上的点曲线) .....	323
8.13.23	NURBSProjectNormalCurve: NURBSCurve (法向投射曲线) .....	323
8.13.24	NURBSProjectVectorCurve: NURBSCurve (矢量投射曲线) .....	323
8.13.25	NURBSSurfaceEdgeCurve: NURBSCurve (曲面边曲线) .....	324
8.13.26	NURBSSurfaceNormalCurve: NURBSCurve (曲面法线曲线) .....	324
8.13.27	NURBSSurfSurfIntersectionCurve: NURBSCurve (曲面-曲面相交曲线) .....	324
8.13.28	NURBSXFormCurve: NURBSCurve (变换曲线) .....	325
8.13.29	NURBSSurface: NURBSObject (曲面子对象) .....	325
8.13.30	NURBS1RailSweepSurface: NURBSSurface (单轨扫描曲面) .....	327
8.13.31	NURBS2RailSweepSurface: NURBSSurface (双轨扫描曲面) .....	328
8.13.32	NURBSBlendSurface: NURBSSurface (混合曲面) .....	329
8.13.33	NURBSCapSurface: NURBSSurface (封口曲面) .....	330
8.13.34	NURBSCVSurface: NURBSSurface (CV 曲面子对象) .....	331
8.13.35	NURBSExtrudeSurface: NURBSSurface (挤出曲面) .....	332
8.13.36	NURBSFilletSurface: NURBSSurface (圆角曲面) .....	332
8.13.37	NURBSLatheSurface: NURBSSurface (车削曲面) .....	333
8.13.38	NURBSMirrorSurface: NURBSSurface (镜像曲面) .....	333
8.13.39	NURBSMultiCurveTrimSurface: NURBSSurface (多重曲线修剪曲面) .....	334
8.13.40	NURBSNBlendSurface: NURBSSurface (混合曲面) .....	334
8.13.41	NURBSOffsetSurface: NURBSSurface (偏移曲面) .....	335
8.13.42	NURBSPointSurface: NURBSSurface (点曲面子对象) .....	335
8.13.43	NURBSRuledSurface: NURBSSurface (规则曲面) .....	336
8.13.44	NURBSULoftSurface: NURBSSurface (U 向放样曲面) .....	337
8.13.45	NURBSUVLoftSurface: NURBSSurface (UV 放样曲面) .....	338
8.13.46	NURBSXFormSurface: NURBSSurface (变换曲面) .....	339
8.13.47	NURBSTexturePoint: NURBSObject (纹理曲面) .....	339
8.13.48	NURBSDisplay: Value .....	339
8.13.49	NURBSSelection: Value .....	340
8.13.50	NURBSSet: Value .....	342
8.13.51	NURBSSurfaceApproximation: Value .....	344
<b>第 9 章 Editable_Mesh、SplineShape、Patch 和 Editable_Poly .....</b>		<b>347</b>
9.1	Editable_Mesh 和 TriMesh: GeometryClass (可编辑网格和三角网格) .....	347

9.1.1 Editable_Mesh 和 TriMesh 构造函数、操作符、属性 .....	347
9.1.2 Mesh 通用方法 .....	350
9.1.3 Mesh Vertex 方法 .....	352
9.1.4 Meshop Vertex 方法 .....	354
9.1.5 Meshop Vertex 数据方法 .....	357
9.1.6 Mesh Edge 方法 .....	359
9.1.7 Meshop Edge 方法 .....	359
9.1.8 Mesh Face 方法 .....	361
9.1.9 Meshop Face 方法 .....	364
9.1.10 Mesh 方法 .....	367
9.1.11 Meshop Mapping 通用方法 .....	369
9.1.12 Meshop Editable_Mesh 用户界面控件方法 .....	372
9.1.13 Mesh Texture Vertex 方法 .....	374
9.1.14 Mesh Color-Per-Vertex 方法 .....	375
9.1.15 Subdivision Displacement Surface 方法 .....	376
9.1.16 Editable_Mesh Modify 面板命令的操作方法 .....	376
9.1.17 使用 Editable_Mesh 的示例 .....	379
9.2 SplineShape: Shape .....	381
9.2.1 SplineShape 属性 .....	381
9.2.2 使用 SplineShape 方法的注意事项 .....	382
9.2.3 Shape 方法 .....	383
9.2.4 Spline 方法 .....	384
9.2.5 Segment 方法 .....	385
9.2.6 Knot 方法 .....	386
9.2.7 Editable_Spline Modify 面板命令的操作方法 .....	387
9.3 Patch: GeometryClass .....	389
9.3.1 Editable_Patch Modify 面板命令的操作方法 .....	390
9.4 Editable_Poly: GeometryClass .....	395
9.4.1 Editable_Poly 方法 .....	403
9.4.2 Editable_Poly Modify 面板命令的操作方法 .....	406
<b>第 10 章 Modifier (对象空间修改器) 和 Spacewarp (世界空间修改器) .....</b>	<b>421</b>
10.1 Modifier: MAXWrapper .....	421
10.2 Modifier 通用属性和方法 .....	422
10.3 Modifier 子对象转换属性 .....	423
10.4 对象空间修改器分类 .....	424
10.4.1 Affect_Region: Modifier (影响区域修改器) .....	427
10.4.2 Bend: Modifier (弯曲修改器) .....	427
10.4.3 Bevel: Modifier (倒角修改器) .....	428

10.4.4	Bevel_Profile: Modifier (倒角截面修改器) .....	429
10.4.5	CameraMap: Modifier (摄影机贴图修改器) .....	430
10.4.6	Cap_Holes: Modifier (补洞修改器) .....	430
10.4.7	CrossSection: Modifier (交叉连线修改器) .....	430
10.4.8	DeleteMesh: Modifier (删除网格修改器) .....	430
10.4.9	DeleteSplineModifier: Modifier (删除样条线修改器) .....	431
10.4.10	Disp_Approx: Modifier (置換近似修改器) .....	431
10.4.11	Displace: Modifier (位移修改器) .....	431
10.4.12	Edit_Mesh: Modifier (编辑网格修改器) .....	432
10.4.13	Edit_Patch: Modifier (编辑面片修改器) .....	433
10.4.14	Edit_Spline: Modifier (编辑样条线修改器) .....	433
10.4.15	Extrude: Modifier (挤出修改器) .....	433
10.4.16	FFDBox: Modifier (自由形式变形长方体修改器) .....	434
10.4.17	FFDCyl: Modifier (自由形式变形圆柱体修改器) .....	435
10.4.18	FFD_2x2x2: Modifier (自由形式变形) .....	437
10.4.19	FFD_3x3x3: Modifier (自由形式变形) .....	438
10.4.20	FFD_4x4x4: Modifier (自由形式变形) .....	439
10.4.21	FFD_Select: Modifier (自由形式变形选择修改器) .....	440
10.4.22	Face_Extrude: Modifier (面挤出修改器) .....	440
10.4.23	Fillet_Chamfer: Modifier (圆角/切角修改器) .....	441
10.4.24	Flex: Modifier (柔体修改器) .....	441
10.4.25	HSDS_Modifier: Modifier (HSDS 修改器) .....	446
10.4.26	HSDSObject: Modifier (HSDS 对象修改器) .....	447
10.4.27	Lathe: Modifier (车削修改器) .....	447
10.4.28	Lattice: Modifier (晶格修改器) .....	448
10.4.29	Linked_XForm: Modifier (链接变换修改器) .....	449
10.4.30	LS_Mesh: Modifier (LS 网格修改器) .....	449
10.4.31	MaterialByElement: Modifier (按元素分配材质修改器) .....	450
10.4.32	MaterialModifier: Modifier (材质修改器) .....	450
10.4.33	Melt: Modifier (融化修改器) .....	451
10.4.34	MeshSmooth: Modifier (网格平滑修改器) .....	451
10.4.35	Mesh_Select: Modifier (网格选择修改器) .....	453
10.4.36	Mirror: Modifier (镜像修改器) .....	454
10.4.37	Morpher: Modifier (变形器修改器) .....	454
10.4.38	MultiRes: Modifier (多分辨率修改器) .....	457
10.4.39	NCurve_Sel: Modifier (NURBS 曲线选择修改器) .....	459
10.4.40	NoiseModifier: Modifier (噪波修改器) .....	459
10.4.41	Normalize_Spl: Modifier (规格化样条线修改器) .....	460

10.4.42	NormalModifier: Modifier (法线修改器) .....	461
10.4.43	NSurf_Sel: Modifier (NURBS 曲面选择修改器) .....	461
10.4.44	Optimize: Modifier (优化修改器) .....	461
10.4.45	PatchDeform: Modifier (面片变形修改器) .....	462
10.4.46	Patch_Select: Modifier (面片选择修改器) .....	463
10.4.47	PathDeform: Modifier (路径变形修改器) .....	463
10.4.48	Point_Cache: Modifier (点缓存修改器) .....	464
10.4.49	Poly_Select: Modifier (多边形选择修改器) .....	465
10.4.50	Preserve: Modifier (保留修改器) .....	466
10.4.51	Push: Modifier (推动修改器) .....	467
10.4.52	Relax: Modifier (松弛修改器) .....	467
10.4.53	Ripple: Modifier (涟漪修改器) .....	467
10.4.54	Skew: Modifier (倾斜修改器) .....	468
10.4.55	Skin: Modifier (蒙皮修改器) .....	469
10.4.56	Skin_Morph: Modifier (蒙皮变形修改器) .....	478
10.4.57	Skin_Wrap: Modifier (蒙皮包裹修改器) .....	484
10.4.58	Skin_Wrap_Patch: Modifier (蒙皮包裹面片修改器) .....	486
10.4.59	SliceModifier: Modifier (切片修改器) .....	486
10.4.60	smooth: Modifier (平滑修改器) .....	487
10.4.61	Spherify: Modifier (球形化修改器) .....	488
10.4.62	Spline_IK_Control: Modifier (样条线 IK 控制修改器) .....	488
10.4.63	SplineSelect: Modifier (样条线选择修改器) .....	489
10.4.64	Squeeze: Modifier (挤压修改器) .....	489
10.4.65	STL_Check: Modifier (STL 检查修改器) .....	490
10.4.66	Stretch: Modifier (拉伸修改器) .....	491
10.4.67	Subdivide: Modifier (细分修改器) .....	491
10.4.68	Substitute: Modifier (替代修改器) .....	492
10.4.69	Surface: Modifier (曲面修改器) .....	492
10.4.70	SurfDeform: Modifier (曲面变形修改器) .....	492
10.4.71	Symmetry:Modifier (对称修改器) .....	493
10.4.72	Taper: Modifier (锥化修改器) .....	494
10.4.73	Tessellate: Modifier (细化修改器) .....	495
10.4.74	Trim_Extend: Modifier (修剪/延伸修改器) .....	495
10.4.75	TurboSmooth: Modifier (涡轮平滑修改器) .....	496
10.4.76	Turn_to_Mesh: Modifier (转化为网格修改器) .....	497
10.4.77	Turn_to_Patch: Modifier (转化为面片修改器) .....	497
10.4.78	Turn_to_Poly: Modifier (转化为多边形修改器) .....	498
10.4.79	Twist: Modifier (扭曲修改器) .....	499

10.4.80 Unwrap_UVW: Modifier (展开 UVW 修改器) .....	500
10.4.81 UVW_Xform: Modifier (UVW 变换修改器) .....	510
10.4.82 UVWmap: Modifier (UVW 贴图修改器) .....	510
10.4.83 Vertex_Colors: Modifier (顶点颜色修改器) .....	512
10.4.84 VertexPaint: Modifier (顶点绘制修改器) .....	512
10.4.85 Vertex_Weld: Modifier (顶点焊接修改器) .....	513
10.4.86 VolumeSelect: Modifier (体积选择修改器) .....	513
10.4.87 Wave: Modifier (波浪修改器) .....	515
10.4.88 XForm: Modifier (变换修改器) .....	516
10.5 世界空间修改器构造函数和属性 .....	516
10.5.1 世界空间修改器分类 .....	518
10.5.2 Displace_Mesh: SpacewarpModifier (位移网格修改器) .....	518
10.5.3 Displace_NURBS: SpacewarpModifier (位移 NURBS 修改器) .....	518
10.5.4 LS_Colors: SpacewarpModifier (LS 颜色修改器) .....	518
10.5.5 MapScaler: SpacewarpModifier (贴图缩放修改器) .....	519
10.5.6 SpaceCameraMap: SpacewarpModifier (摄影机贴图修改器) .....	519
10.5.7 SpacePatchDeform: SpacewarpModifier (面片变形修改器) .....	520
10.5.8 SpacePathDeform: SpacewarpModifier (路径变形修改器) .....	520
10.5.9 SpaceSurfDeform: SpacewarpModifier (曲面变形修改器) .....	521
10.5.10 SubdivideSpacewarpModifier:SpacewarpModifier (细分修改器) .....	521
10.5.11 Surface_Mapper: SpacewarpModifier (曲面贴图修改器) .....	522
第 11 章 Material (材质) 和 TextureMap (贴图) .....	523
11.1 Material 类通用属性和方法 .....	523
11.2 Material 材质类型 .....	524
11.2.1 Advanced_Lighting_Override: Material (高级照明覆盖材质) .....	525
11.2.2 Architectural: Material (建筑材质) .....	525
11.2.3 Blend: Material (混合材质) .....	529
11.2.4 CompositeMaterial: Material (合成材质) .....	530
11.2.5 DoubleSided: Material (双面材质) .....	531
11.2.6 InkNPaint: Material (卡通材质) .....	531
11.2.7 Lightscape_Mtl: Material (Lightscape 材质) .....	536
11.2.8 MatteShadow: Material (不可见/投影材质) .....	537
11.2.9 MorpherMaterial: Material (变形材质) .....	538
11.2.10 MultiMaterial: Material (多维材质) .....	539
11.2.11 NoMaterial: Material (无材质) .....	540
11.2.12 RayTraceMaterial: Material (光线跟踪材质) .....	540
11.2.13 StandardMaterial: Material (标准材质) .....	549
11.2.14 Shellac: Material (胶合材质) .....	556

11.2.15 TopBottom: Material (顶/底材质) .....	556
11.3 TextureMap: Material (贴图) .....	557
11.3.1 TextureMap 类通用属性和方法 .....	557
11.3.2 TextureMap 的三个共享类 .....	559
11.3.3 UVGenClass: Material.....	559
11.3.4 StandardXYZGen: Material.....	561
11.3.5 TextOutputClass: Material .....	561
11.4 贴图类型 .....	562
11.4.1 Adobe_Photoshop_Plug_In_Filter: TextureMap.....	563
11.4.2 Adobe_Premiere_Video_Filter: TextureMap .....	564
11.4.3 BitmapTexture: TextureMap (位图贴图) .....	564
11.4.4 Bricks: TextureMap (砖块贴图) .....	567
11.4.5 Cellular: TextureMap (细胞贴图) .....	568
11.4.6 Checker: TextureMap (方格贴图) .....	570
11.4.7 CompositeTextureMap: TextureMap (合成贴图) .....	570
11.4.8 Dent: TextureMap (凹痕贴图) .....	571
11.4.9 Falloff: TextureMap (衰减贴图) .....	571
11.4.10 FalloffTextureMap: TextureMap (衰减纹理贴图) .....	573
11.4.11 FlatMirror: TextureMap (平面镜贴图) .....	574
11.4.12 Gradient: TextureMap (渐变贴图) .....	575
11.4.13 Gradient_Ramp: TextureMap (渐变坡度贴图) .....	576
11.4.14 Marble: TextureMap (大理石贴图) .....	578
11.4.15 Mask: TextureMap (遮罩贴图) .....	578
11.4.16 Mix: TextureMap (混合贴图) .....	579
11.4.17 Noise: TextureMap (躁波贴图) .....	580
11.4.18 NoTexture: TextureMap (无纹理贴图) .....	580
11.4.19 Output: TextureMap (输出贴图) .....	581
11.4.20 Paint: TextureMap (绘制贴图) .....	581
11.4.21 Particle_Age: TextureMap (粒子年龄贴图) .....	581
11.4.22 Particle_MBlur: TextureMap (粒子运动模糊贴图) .....	582
11.4.23 Perlin_Marble: TextureMap (Perlin 大理石贴图) .....	583
11.4.24 Planet: TextureMap (行星贴图) .....	584
11.4.25 Raytrace: TextureMap (光线跟踪贴图) .....	585
11.4.26 Reflect_Refraction: TextureMap (反射和折射贴图) .....	588
11.4.27 RGB_Multiply: TextureMap (RGB 倍增贴图) .....	589
11.4.28 RGB_Tint: TextureMap (RGB 色彩贴图) .....	590
11.4.29 Smoke: TextureMap (烟雾贴图) .....	591
11.4.30 Speckle: TextureMap (斑纹贴图) .....	591

11.4.31 Splat: TextureMap (泼溅贴图) .....	592
11.4.32 Stucco: TextureMap (灰泥贴图) .....	592
11.4.33 Swirl: TextureMap (旋涡贴图) .....	593
11.4.34 Thin_Wall_Refraction: TextureMap (薄壁折射贴图) .....	594
11.4.35 Vertex_Color: TextureMap (顶点颜色贴图) .....	595
11.4.36 Water: TextureMap (波浪贴图) .....	595
11.4.37 Wood: TextureMap (木材贴图) .....	596
<b>第 12 章 动画控制器 .....</b>	<b>598</b>
12.1 Controller (控制器) 类 .....	598
12.2 控制器通用属性 .....	598
12.3 控制器通用方法 .....	599
12.4 与控制器时间有关的方法 .....	601
12.5 与控制器关键帧有关的方法 .....	602
12.6 与控制器 ORT 有关的方法 .....	604
12.7 与控制器 Ease 曲线和 Multiplier 曲线有关的方法 .....	605
12.8 与控制器关键帧衰减有关的方法 .....	606
12.9 与对象层级有关的时间和关键帧方法 .....	607
12.10 控制器类型 .....	608
12.10.1 控制器超类级 .....	608
12.10.2 Attachment: PositionController (附着点约束控制器) .....	609
12.10.3 Audio Controller (音频控制器) .....	610
12.10.4 Barycentric_Morph_Controller: MorphController (重心变形控制器) .....	610
12.10.5 Bezier Controller (贝塞尔控制器) .....	612
12.10.6 Block: FloatController (块控制器) .....	614
12.10.7 Block_Control: MasterBlockController .....	614
12.10.8 Dynamics Controller (运动控制器) .....	614
12.10.9 Expression Controller (表达式控制器) .....	614
12.10.10 IK_ControllerMatrix3Controller: Matrix3Controller (反向动力学控制器) .....	615
12.10.11 Linear Controller (线性控制器) .....	616
12.10.12 Link_Control: Matrix3Controller (链接控制器) .....	617
12.10.13 List Controller (列表控制器) .....	617
12.10.14 LOD_Controller: FloatController .....	619
12.10.15 LookAt: Matrix3Controller (注视约束控制器) .....	620
12.10.16 MasterBlock: MasterBlockController .....	620
12.10.17 Motion Capture Controllers (运动捕捉控制器) .....	621
12.10.18 Noise Controllers (噪波控制器) .....	621
12.10.19 On_Off: FloatController (开关控制器) .....	622
12.10.20 Path: PositionController (路径约束控制器) .....	623

12.10.21 PRS: Matrix3Controller (PRS 控制器) .....	624
12.10.22 Reactor Controller (连锁反应控制器) .....	624
12.10.23 Script Controller (脚本控制器) .....	626
12.10.24 Slave_Control: Matrix3Controller (附属控制器) .....	629
12.10.25 Slave Controller (附属控制器) .....	630
12.10.26 Surface_position: PositionController (表面约束控制器) .....	630
12.10.27 TCB Controller (TCB 控制器) .....	630
12.10.28 Waveform_Float: FloatController (波形控制器) .....	631
12.10.29 XYZ Controller (XYZ 控制器) .....	631
<b>第 13 章 Atmospheric (环境效果) .....</b>	<b>634</b>
13.1 Atmospheric 类通用属性和方法.....	634
13.2 Atmospheric Effect (环境效果类型) .....	636
13.3 Fire_Effect: Atmospheric (火焰环境效果) .....	636
13.4 Fog: Atmospheric (雾环境效果) .....	638
13.5 Volume_Fog: Atmospheric (体积雾环境效果) .....	639
13.6 Volume_Light: Atmospheric.....	640
13.7 使用 Atmospheric 的示例.....	642
<b>第 14 章 RenderEffect (渲染效果) .....</b>	<b>644</b>
14.1 渲染效果通用属性和方法 .....	644
14.2 渲染效果类型 .....	645
14.3 Blur:RenderEffect (模糊渲染效果) .....	645
14.4 Brightness_and_Contrast: RenderEffect (亮度和对比度渲染效果) .....	648
14.5 Color_Balance: RenderEffect (颜色平衡渲染效果) .....	649
14.6 Depth_of_Field: RenderEffect (景深渲染效果) .....	649
14.7 File_Output: RenderEffect (文件输出渲染效果) .....	651
14.8 Film_Grain: RenderEffect (胶片颗粒渲染效果) .....	652
14.9 Lens_Effects: RenderEffect (镜头渲染效果) .....	652
14.9.1 Lens_Effects-Auto_Secondary (自动二级光斑镜头效果) .....	655
14.9.2 Lens_Effects-Glow (光晕镜头效果) .....	661
14.9.3 Lens_Effects-Manual_Secondary (手动二级光斑镜头效果) .....	666
14.9.4 Lens_Effects-Ray (射线镜头效果) .....	671
14.9.5 Lens_Effects-Ring (光环镜头效果) .....	676
14.9.6 Lens_Effects-Star (星形镜头效果) .....	680
14.9.7 Lens_Effects-Streak (条纹镜头效果) .....	685
14.10 Motion_Blur: RenderEffect (运动模糊渲染效果) .....	689

## 第3部分 用 MAXScript 创建实用工具、用户界面

第 15 章 创建脚本工具程序 Utility.....	691
15.1 关于定制脚本工具程序 Utility.....	691
15.2 定义脚本工具程序 Utility.....	691
15.3 Utility 子句.....	692
15.4 在一个脚本工具 Utility 里定义多个卷展栏.....	694
15.5 Rollout 子句.....	696
15.5.1 局部变量、全局变量声明和函数、结构定义.....	696
15.5.2 用户界面控件<user_interface_item>.....	697
15.5.3 用户界面控件组<item_group>.....	697
15.5.4 控件事件处理程序<event_handler>.....	698
15.6 Utility 和 Rollout 的属性、方法和事件处理程序.....	699
15.7 Rollout 浮动窗口.....	701
15.8 局部变量、函数、结构和用户界面控件的定义顺序.....	702
15.9 从外部代码里存取 Utility 内部局部变量和控件.....	704
15.10 Rollout 用户界面控件.....	705
15.10.1 控件通用属性.....	705
15.10.2 控件通用布局参数.....	706
15.10.3 控件类型.....	707
15.10.4 Angle (角填充) .....	708
15.10.5 Bitmap (图像框) .....	710
15.10.6 Button (按钮) .....	711
15.10.7 Checkbox (复选框) .....	711
15.10.8 Checkbutton (复选按钮) .....	712
15.10.9 Colorpicker (颜色拾取器) .....	713
15.10.10 Combobox (组合框) .....	713
15.10.11 CurveControl (曲线控件) .....	715
15.10.12 Dropdownlist (下拉列表) .....	721
15.10.13 Edittext (编辑框) .....	722
15.10.14 GroupBox (组合框) .....	723
15.10.15 HyperLink (超级链接) .....	724
15.10.16 ImgTag (图像) .....	724
15.10.17 Label (标签) .....	725
15.10.18 Listbox (列表框) .....	726
15.10.19 Mapbutton (贴图按钮) .....	726
15.10.20 Materialbutton (材质按钮) .....	727

15.10.21 MultiListbox (多选列表框) .....	728
15.10.22 Pickbutton (对象拾取按钮) .....	729
15.10.23 PopupMenu (右键弹出菜单) .....	730
15.10.24 ProgressBar (进度栏) .....	731
15.10.25 Radiobuttons (单选按钮) .....	731
15.10.26 Slider (滑标) .....	732
15.10.27 Spinner (数值微调器) .....	733
15.10.28 SubRollout .....	734
15.10.29 Timer (计时器) .....	736
15.11 图像按钮 .....	737
<b>第 16 章 RcMenu (右键菜单) .....</b>	<b>738</b>
16.1 RcMenu 子句 .....	739
16.2 RcMenu 用户界面控件 .....	740
16.2.1 MenuItem (菜单项) .....	740
16.2.2 Separator (分隔行) .....	741
16.2.3 Submenu (子菜单) .....	742
<b>第 17 章 宏脚本 (MacroScript) .....</b>	<b>743</b>
17.1 定义宏脚本 .....	743
17.2 创建图标位图文件 .....	750
<b>第 18 章 脚本鼠标工具 .....</b>	<b>752</b>
18.1 脚本鼠标工具定义 .....	752
18.2 MouseTool 子句 .....	753
<b>第 19 章 脚本插件 .....</b>	<b>757</b>
19.1 Plug-in 子句 .....	759
19.1.1 局部变量声明和函数、结构定义 .....	759
19.1.2 参数块<parameters> .....	761
19.1.3 鼠标工具<tools> .....	765
19.1.4 定制用户界面卷展栏<Rollouts> .....	765
19.1.5 事件处理程序<event_handler> .....	765
19.2 脚本插件方法 .....	766
19.3 脚本插件的更新 .....	768
19.4 Geometry (几何体) 类脚本插件 .....	769
19.5 SimpleObject 类脚本插件 .....	771
19.6 Shape 类脚本插件 .....	774
19.7 Light 类脚本插件 .....	775

19.8 Helper 类脚本插件 .....	775
19.9 Modifier 类脚本插件 .....	775
19.10 SimpleMod 类脚本插件 .....	776
19.11 Material 类脚本插件 .....	778
19.12 TextureMap 类脚本插件 .....	779
19.13 RenderEffect 类脚本插件 .....	779
19.14 Atmospheric 类脚本插件 .....	781

## 第 4 部分 MAXScript 的高级应用

第 20 章 在 MAXScript 里与用户界面交互 .....	783
20.1 Main Toolbar (主工具栏) .....	783
20.2 Status Bar (状态栏) .....	786
20.2.1 Prompt Line (提示栏) .....	786
20.2.2 Coordinate Display (坐标显示) .....	786
20.2.3 Progress Bar (进度栏) .....	787
20.2.4 Status Bar Button (状态栏) .....	787
20.3 Time Control (时间控制) .....	788
20.4 Trackbar (轨迹栏) .....	788
20.5 Viewport (视窗) .....	789
20.5.1 存取当前视窗信息、类型和 Transform 信息 .....	789
20.5.2 刷新视窗 .....	793
20.5.3 视窗背景图像操作 .....	793
20.5.4 视窗网格 (Viewport Grid) .....	794
20.5.5 鼠标光标 (Mouse Cursor) .....	794
20.5.6 在视窗里拾取点 .....	795
20.5.7 3ds max 图形系统的低级存取方法 .....	797
20.5.8 其他视窗方法和系统变量 .....	804
20.6 3ds max 用户界面颜色 .....	804
20.7 Material Editor .....	808
20.8 轨迹视图 (Track View) .....	809
20.9 渲染场景 (Render Scene) 对话框 .....	811
20.10 图解视图 (Schematic View) .....	813
20.11 Time Configuration 对话框 .....	813
20.12 RAMPlayer .....	814
20.13 Track View Pick 对话框 .....	815
20.14 选择场景对象 .....	815
20.14.1 点击选取场景对象 .....	816

20.14.2 用对象名选择场景对象.....	816
20.14.3 用区域来选择场景对象.....	817
20.15 提示信息框和询问对话框 .....	818
20.16 其他对话框.....	819
20.17 键盘输入 .....	820
20.18 3ds max 系统路径.....	820
20.19 3ds max 场景文件属性.....	822
<b>第 21 章 在 MAXScript 里存取文件.....</b>	<b>825</b>
21.1 3ds max 场景文件的装载和保存 .....	825
21.2 与 Bitmap 文件有关的方法.....	827
21.3 标准文件打开、存储对话框 .....	828
21.4 文件名提取 .....	828
21.5 外部文件方法 .....	829
21.6 加密文件 .....	830
21.7 存取.INI 文件 .....	831
21.8 存取.CUI 文件 .....	831
<b>第 22 章 事件侦测和信号反馈机制.....</b>	<b>832</b>
22.1 事件侦测和 when 构造函数.....	832
22.2 时间改变信号反馈机制 .....	835
22.3 视窗刷新信号反馈机制 .....	836
22.4 通用事件反馈机制 .....	836
<b>第 23 章 MAXScript 杂项函数.....</b>	<b>844</b>
23.1 暂停脚本执行 .....	844
23.2 时间计算函数 .....	844
23.3 控制渲染器 .....	844
23.4 执行外部命令或程序 .....	848
23.5 退出和重置 3ds max 系统.....	849
23.6 其他函数 .....	849

第  
1  
部  
分

MAXScript  
语法基础

# 第1章 了解 MAXScript

本章讲述了从哪里开始 MAXScript 脚本语言, 新建、编辑、运行脚本程序及与 MAXScript 语言有关的两个窗口, 最后通过一个简单的 MAXScript 例题演示了用脚本语言可以做哪些事情。通过本章的阅读, 将对脚本语言有一个初步的印象。在本章中会涉及到一些函数和专业术语, 你可能会感到很陌生, 可以先跳过它们, 在以后的章节中会一一介绍到。

## 1.1 如何开始 MAXScript

MAXScript 是 3ds max R2.0 版以后增加的一种内嵌的脚本语言, 因此它只能在 3ds max 界面下使用。有两种方法可以在 3ds max 里打开 MAXScript。

最简单的方法是在 MAXScript 菜单栏里选择 MAXScript Listener, 可以打开 MAXScript Listener 窗口, 如图 1-1 所示。

在一个 3ds max 窗口里, 一次只能打开一个 MAXScript Listener 窗口, 该窗口是一个大小可调的无模式窗口, 用户可以在其和别的软件窗口之间任意切换。现在我们来认识一下 Listener 窗口。

Listener 窗口分成上下两部分: 上部分(粉红色)为宏记录区域, 下部分(白色)为输出区域。如果宏记录区域在 Listener 窗口打开后不可见, 用鼠标按住分隔条向下拖拉就可以打开该区域。用户操作 3ds max 过程中每一条可记录的命令都会作为一行脚本语言显示在 Listener 窗口的宏记录区域里。用户在两个区域里都可以进行文本的复制、剪切、粘贴、拖拉、编辑和选择运行源程序代码。

另一种方法是在命令面板里按下  , 选择 UTILITY | MAXScript, 可以打开 MAXScript 卷展栏, 显示如图 1-2 所示。

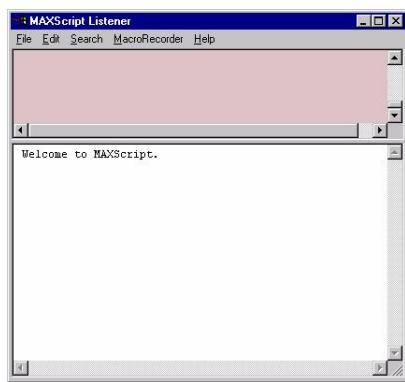


图 1-1 MAXScript Listener 窗口



图 1-2 MAXScript 卷展栏

MAXScript 卷展栏各按钮说明如下：

- ◆ Open Listener 打开 MAXScript Listener 窗口。
- ◆ New Script 打开一个新的 MAXScript Editor 窗口来写入新的脚本。
- ◆ Open Script 打开一个已存储的脚本文件，该脚本文件会被显示在一个新的 MAXScript Editor 窗口里。
- ◆ Run Script 打开并运行脚本文件，运行过程的所有输出都会显示在 Listener 窗口的输出区域里。
- ◆ Utilities 显示当前可用的脚本工具清单，只有显示在该清单里的脚本工具，MAXScript 才能执行它。有关脚本工具的更多内容请参见本书第 15 章。

## 1.2 如何新建、编辑、运行脚本文件

我们已经知道如何开始 MAXScript 了，接下来将学习如何新建、编辑、运行已有的脚本文件。脚本文件后缀为\*.ms 或\*.mse。

### 新建脚本文件

新建一个脚本文件有下面两种方法：

方法一：在 MAXScript 菜单栏下选择 New Script。

方法二：在  UTILITY 命令面板里展开 MAXScript 卷展栏，按下 New Script 按钮。

### 编辑脚本文件

同样，编辑一个已有的脚本文件也有下面两种方法：

方法一：在 MAXScript 菜单栏下选择 Open Script。

方法二：在  UTILITY 命令面板里展开 MAXScript 卷展栏，按下 Open Script 按钮，在弹出的文件选择对话框里选择要编辑的脚本文件。

**注意** 用别的文本编辑工具如写字板也可以新建、编辑脚本文件。

### 运行脚本文件

运行 3ds max 脚本文件的方法有下面四种：

方法一：在 MAXScript 菜单栏里选择 Run Script。在弹出的文件选择对话框中选择要运行的脚本文件后，单击 Open 按钮，MAXScript 就立刻开始执行指定的脚本文件。

方法二：在  UTILITY 命令面板里，展开 MAXScript 卷展栏，单击 Run Script 按钮。

方法三：从 Windows 命令行里直接运行脚本。

可以在启动 3ds max 的同时运行一个指定的脚本文件。为了达到此目的，要用到 3ds max 的一个命令行开关 “-U”，如下例：

```
c:\3ds max5\3dsMAX -U MAXScript rendercams.ms
```

脚本文件 rendercams.ms 自动装入两个场景，并对每个摄影机进行渲染：

```

loadMaxFile "foo.max"
for c in cameras do render camera:c outputFile:( "foo"+c.name+".bmp")
loadMaxFile "baz.max"
for c in cameras do render camera:c outputFile:( "baz_"+c.name+".bmp")
quitMax #noPrompt

```

要实现在 Windows 命令行里直接运行上面的脚本文件，有下面两种方法：

- (1) 选择 3ds max 桌面图标，按右键选择“属性 | 快捷方式”标签，修改“目标 (T)”为“c:\3dsmax8\3dsmax -U MAXScript rendercams.ms”，如图 1-3 所示。
- (2) 选择“开始 | 运行”在“打开 (O)”栏里输入“c:\3dsmax8\3dsmax -U MAXScript rendercams.ms”，如图 1-4 所示。



图 1-3 3ds max 快捷方式建立窗口

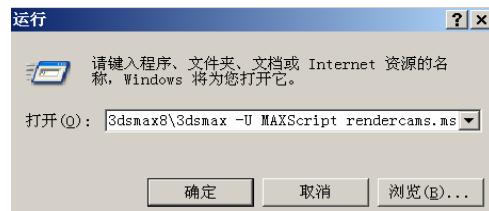


图 1-4 选择 Windows 运行窗口

方法四：可以在 Listener 窗口或其他脚本文件中用下面函数运行脚本：

```
filein <filename_string>[quiet:<boolean>]
```

其中参数<filename\_string>为指定脚本文件的字符串或求值结果为字符串的表达式，参数quiet:为可选参数，指定当装载文件时，是否在 Listener 窗口里列出程序代码清单，默认值为 quiet:True，表示不列出清单。例如：

```
filein "my_script.ms"
```

### 1.3 在 3ds max 开始运行时加载所需的脚本文件

熟悉 AutoCAD 的内嵌语言 AutoLISP 的读者应该知道：AutoCAD 在每次启动时都会在文件搜索路径里查找文件 acad.lsp，一旦找到该文件，就会自动装载它。在 3ds max 里也有类似功能。

假如我们有一个函数库，并且每次在 MAXScript 里都需要预先装载它们用来对界面作一些设置、装载一些工具卷展栏脚本时，如果这些工作都能在启动 3ds max 时由系统自动

进行，将会大大简化我们的工作。在3ds max里，启动脚本文件startup.ms就可以实现这种功能。

设置自动查找启动脚本文件的功能的方法为：在3ds max菜单中选择Customize | Preference，在弹出的Preference Settings对话框里选择MAXScript选项卡，然后按如图1-5所示进行设置。

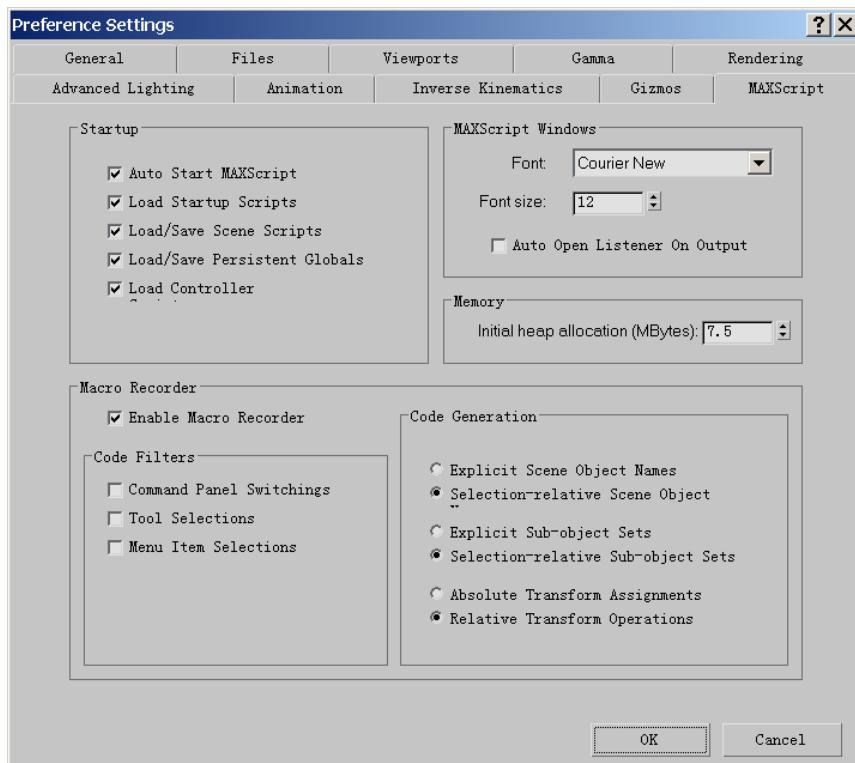


图1-5 Preference Settings对话框

按图1-5设置好后，每次启动3ds max时系统会按下面的顺序查找脚本文件：

(1) 首先，MAXScript按下面路径顺序查找名为startup.ms的文件并把它装入系统：

Scripts路径→Startup Scripts路径→3ds max主路径→32位Windows NT系统路径(system32)→16位Windows NT系统路径(system)→Windows路径→在Windows系统PATH环境变量里列出的路径。

其中Scripts路径与Startup Scripts路径可以在3ds max的Customize | Configure Paths对话框里进行定义，如图1-6所示。

(2) 当找到第一个startup.ms文件后，MAXScript会停止查找，然后MAXScript会对Plug-ins路径和Startup Scripts路径以及其下级路径进行扫描，所有以\*.ms和\*.mse为后缀的脚本文件都会被自动装入系统。在此过程中，如果又发现名为startup.ms的文件，系统会自动跳过它。如果我们不希望某一个下级路径里的文件被系统自动装入，可以把路径名用括号括起来，如“(old-version)”。

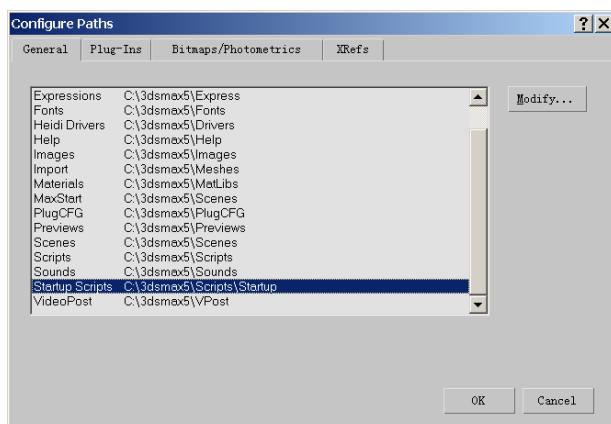


图 1-6 Configure Paths 设置窗口

## 1.4 MAXScript Listener 窗口

### 1.4.1 MAXScript Listener 窗口的功能及特点

MAXScript Listener 窗口兼有文本编辑和命令提示的功能，它既可以执行用户窗口里输入的命令，也可以对该命令进行编辑然后执行修改后的命令。

关于 Listener 窗口的使用，有下面几点提示：

#### 1. 编辑、执行窗口中间的命令

可以执行或编辑 Listener 窗口中已有的文本，区分这两者的方法为：

(1) 在当前鼠标位置按下 Enter 键，插入一个空行。

(2) 按 Shift+Enter 键或数字键盘的 Enter 键执行鼠标所在行，执行命令产生的输出信息或错误信息会显示在 Listener 窗口输出区域的当前行之后。

#### 2. 文本末的命令

使用 Listener 窗口最简单的方法就是在已有文本的末尾输入命令，按 Enter 键执行，这叫做“end-of-text compilation”（文本末编译）。

#### 3. 选择并执行命令

我们可以选择多行文本，然后按 Shift+Enter 键来逐行执行所选择的文本；也可以仅选择一行命令里能构成一个有效的 MAXScript 表达式的一部分，按 Shift+Enter 键来执行。

#### 4. 执行表达式块

MAXScript 命令可能是单行或多行命令，也可以是一个由许多命令组成的表达式块。如果用户想在已有的命令中加入一个表达式块或多行命令，可以先按 Enter 键插入所有行后选择它们，然后按 Shift+Enter 键来执行。

如果用户想在 Listener 窗口的文本末尾输入一个多行命令或表达式块，当每次按下 Enter 键时系统将编译每一行，但只有在多行命令或表达式块输入完整后，才会被执行。我们仅能在按下 Enter 键之前修改本行命令，而不能对多行命令或表达式块的前面部分进行

修改。用户也可以按 Esc 键取消本次多行命令或表达式块的编译。

### 5. 将表达式块定制成宏脚本

可以选择一行或多行命令，将它们拖至 3ds max 工具栏，创建一个包含这些命令的宏脚本。有关宏脚本的更多信息，请参见本书第 17 章。

Listener 窗口具有以下特点：

- ◆ 当用户把鼠标移到左边边框边缘时，会变成一个指向右边的箭头，单击鼠标可以选取整行文本；按住鼠标拖动可以进行多行选择。
- ◆ 可以在 Listener 窗口内或与 MAXScript Editor 窗口之间进行文本的拖拉复制。
- ◆ 当用户执行 Listener 窗口下 Search | Find or Search 或 Replace 命令时，用户刚刚选择的文本将自动作为查找内容。
- ◆ 对输入文本、输出文本、错误信息都使用不同的颜色显示，以便于区别。有三个 MAXScript 系统变量控制这三种颜色，用户可以将它们改变成自己喜欢的颜色。如下表所示。

文本类型	系统变量	默认值
输入文本	inputTextColor	Black (黑色)
输出文本	outputTextColor	Blue (蓝色)
错误信息	messageTextColor	Red (红色)

### 1.4.2 MAXScript Listener 命令

下表是 Listener 窗口里的菜单命令和快捷键，其中 Edit 命令也可以在单击鼠标右键的弹出菜单里找到。

命令名称	快捷键	说明
File   Close	Ctrl+W	关闭当前 Listener 窗口
File   Save As	Ctrl+S	将当前 Listener 窗口里的脚本存为另一个文件
File   New Script	Ctrl+N	打开一个新的脚本文件
File   Run Script	Ctrl+R	运行 Listener 窗口里的脚本文件
Edit   Undo	Ctrl+Z	取消上一次操作
Edit   Cut	Ctrl+X	文本剪切
Edit   Copy	Ctrl+C	文本复制
Edit   Paste	Ctrl+V	文本粘贴
Edit   Delete	Del	文本删除
Edit   Clear All		清除活动 Listener 窗口里所有内容
Edit   Select All	Ctrl+A	选取 Listener 窗口里的全部文本
Search   Find	Ctrl+F	字符查找
Search   Find Next	Ctrl+G	再次查找字符
Search   Replace	Ctrl+H	字符替换

(续表)

命令名称	快捷键	说明
Help   Help	F1	显示 MAXScript 在线帮助
Help   About MAXScript		显示 About MAXScript 对话框
	Ctrl+B	选择当前括号内的文本。这样可以帮助我们检查一段很长的源程序的括号匹配情况。括号可以是(),[],{}。将鼠标置于源程序任意位置,按下 Ctrl +B, 如果鼠标处于一个括号之后,那么从该括号开始直至与之匹配的括号结束的程序代码将被选择。如果鼠标没有紧跟在一个括号之后,最近的一个括号对里包含的程序代码被选择;第二次按下 Ctrl +B, 上一层括号对里的代码将被选择;反复按下 Ctrl+B, 将遍历程序里所有的括号嵌套。如果在某一点发现括号不匹配,系统会给出蜂鸣提示,不选择任何文本

### 1.4.3 宏记录器 (MacroRecorder)

Listener 窗口还有一个很重要的功能: 记录用户在 3ds max 界面里的大部分操作。每一个可记录的命令会作为一个命令行显示在 Listener 窗口的宏记录区域里。这种功能类似于录音机的录音功能, 我们称之为“宏记录”(MacroRecorder)。对绝大多数菜单栏上的按钮、工具栏、状态栏、Create 命令面板、Modify 命令面板的操作都会产生 MacroRecorder 输出。但是如果这些按钮激活一个对话框, 在这个对话框里进行的修改设置 MacroRecorder 将不作记录。在 Create 面板和 Modify 面板里, 用 MAXScript 能创建的所有对象类型的操作都会产生 MacroRecorder 输出。

下面列出了 Listener 窗口里 MacroRecorder 菜单项下的所有开关选项。

#### 1. Enable

如果 Enable 被选择, 宏记录才会在 Listener 窗口里生成相应 MAXScript 命令。

#### 2. Explicit Scene Object Names/Selection-Relative Scene Object Names

一对互斥开关, 指定在生成的宏记录中使用对象名还是选择符“\$”。

如果选择 Explicit Scene Object Names, 生成的宏记录为如下形式:

```
move $Sphere03 [10,10,0]
```

如果选择 Selection-Relative Scene Object Names, 生成的宏记录为如下形式:

```
move $ [10,10,0]
```

#### 3. Absolute Transform Assignments/Relative Transform Operations

一对互斥开关, 指定在生成的宏记录中使用绝对转换赋值还是相对转换操作。

如果选择 Absolute Transform Assignments, 当在视窗里移动一个对象选集时, 生成的宏记录为如下形式:

```
$ .position = [55.6739,23.5,0]
```

如果选择 Relative Transform Operations, 相应的宏记录为如下形式:

```
move $ [0,-47.8044,0]
```

当选择 Absolute Transform Assignments 选项时，绝对转换赋值仅用于单个对象被转换的情况，如果对象选择集里有多个对象，产生的宏记录仍然使用相对转换操作。

#### 4. Explicit Sub-object Sets/Selection-Relative Sub-object Sets

一对互斥开关，指定在生成的宏记录中使用子对象标识符还是子对象选择集属性。

如果选择 Explicit Sub-object Sets，生成的宏记录为如下形式：

```
move $Sphere02.verts[#{20..32, 51..65}] [40.0986,10.3648,0]
```

如果选择 Selection-Relative Sub-object Sets，生成的宏记录为如下形式：

```
move $Sphere02.selectedVerts [40.0986,10.3648,0]
```

如果使用 Selection-Relative Sub-object Sets 选项，记录下来的脚本也可以用于别的选择集，这样可以生成一些通用的脚本。如果用户希望脚本总是对同一子对象进行操作，而不是当前的选择集，就可以使用 Explicit Sub-object Sets 选项。

#### 5. Command Panel Switchings

如果选择 Command Panel Switchings 选项，宏记录会为不同命令面板之间的切换生成脚本命令。在绝大多数情况下，记录命令面板之间的切换是多余的，因为大多数脚本并不依赖用户界面的状态。

#### 6. Tool Selections

如果选择 Tool Selections 选项，当用户按下 3ds max 的工具栏里的某一按钮时，宏记录会生成一条命令。在绝大多数情况下，记录工具栏里的按钮状态是多余的。

#### 7. Menu Item Selections

如果选择 Menu Item Selections 选项，当用户按下 3ds max 的某一菜单项时，宏记录会生成一条命令。

### 1.4.4 Listener 日志文件

当用 Listener 窗口工作时，我们可以使用日志文件来记录所有输入、输出文本。日志文件可以记录输入到 Listener 窗口的文本（上下两个区域都可以）和显示在输出区域的输出结果。但宏记录器（MacroRecorder）的输出结果不能被记录。

用下面的函数可以打开一个日志文件：

```
openLog <filename_string> [ mode : " w" | " a" ] [ outputOnly:<boolean> ]
```

例如：

```
openLog "my_log.txt" mode: "a" outputOnly: True
```

函数的几个参数说明如下：

- ◆ <filename\_string> 是一个字符串或返回字符串的表达式，指定了要创建日志文件的文件名。
- ◆ mode 是一个可选参数，指定了文件打开的模式，默认模式“w”将创建一个新文件或覆盖已有的同名文件。模式“a”把记录加到文件的末尾，如果指定文件不存在，将导致一个错误信息。

- ◆ `outputOnly` 指定了记录的内容，默认值为 `False`，记录输入和输出。如果指定 `outputOnly:True` 将仅记录输出内容。

日志文件数据并不被连续地写入日志文件，而是先被写入内存缓冲，当内存缓冲被写满时，系统再把缓冲里的数据写入日志文件，可以用函数 `flushLog()` 来保证所有数据都被写入文件。

函数 `closeLog()` 可以做下面三件事情：停止记录、刷新日志缓冲、关闭日志文件。

如果在调用函数 `flushLog()` 或 `closeLog()` 时并没有打开日志文件，系统不会给出错误信息。

## 1.5 MAXScript Editor 窗口

### 1.5.1 MAXScript Editor 窗口功能及特点

MAXScript Editor 窗口是一个文本编辑窗口，用户可以在 3ds max 里用它进行文本文件（主要是 MAXScript 脚本文件）的创建、编辑。如图 1-7 所示。

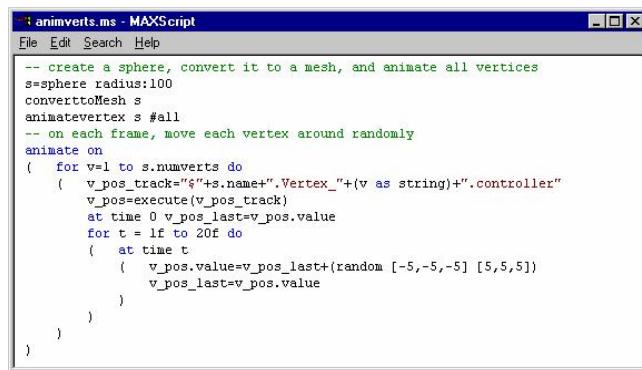


图 1-7 MAXScript Editor 窗口

MAXScript Editor 窗口的功能和菜单项与 Windows 自带的记事本相似，可以同时打开任意数量的 Editor 窗口，适合于编制复杂的脚本文件、工具和函数库，其功能如下：

- ◆ 可以选择 Editor 窗口里一行或几行脚本，将其拖拉至 3ds max 的工具栏，创建一个脚本宏。
- ◆ 可以通过调用 `edit()` 函数在 Listener 窗口中或其他正在运行的脚本文件中打开一个 MAXScript Editor 窗口，其语法为：

```
edit <filename_string>
```

其中参数 `<filename_string>` 为一个字符串或求值结果为字符串的表达式，它指定了要装入 MAXScript Editor 窗口的脚本文件名，例如：

```
Scriptfile= "my_script.ms"
```

- ◆ 可以在 Listener 窗口或正在运行的脚本文件中创建一个新的脚本文件，其语法为：

```
newScript()
```

下面的例子将创建一个新的脚本文件，然后把一些中间结果写入文件中：

```
debug = newScript()
...
print $foo to:debug
...
format "name is % | n" obj.name to:debug
```

- ◆ 如果我们需要查找某一个脚本函数在哪一个脚本文件中被定义，可以使用下面的方法：

```
showSource <fn>
```

系统会打开一个新的 MAXScript Editor 窗口，显示定义该函数的脚本文件，并将光标定位在该函数定义的起点。

MAXScript Editor 窗口具有与 Listener 窗口相同的特点，请参见 1.4.1 节。

### 1.5.2 MAXScript Editor 窗口的菜单命令

下表是 MAXScript Editor 窗口的所有菜单命令和快捷键，其中 Edit 菜单命令也可以用右键弹出菜单得到，如图 1-8 所示。

命令名称	快捷键	说明
File   New	Ctrl+N	打开一个新 Editor 窗口
File   Open	Ctrl +O	打开一个 File Open 对话框，让用户选择一个已有的脚本文件，并在一个新的 Editor 窗口里打开该文件
File   Close	Ctrl+W	关闭当前 MAXScript Editor 窗口
File   Save As	Ctrl+S	存储 MAXScript Editor 窗口里的内容到当前文件名里，然后关闭 MAXScript Editor 窗口。如果还没有文件名，会打开一个 File Save 对话框
File   Evaluate All	Ctrl+E	对当前窗口内全部内容求值
Edit   Undo	Ctrl+Z	取消上一次操作，仅支持一级取消
Edit   Cut	Ctrl+X	文本剪切
Edit   Copy	Ctrl+C	文本复制
Edit   Paste	Ctrl+V	文本粘贴
Edit   Delete	Del	文本删除
Edit   Select All	Ctrl+A	选取 Editor 窗口里的全部文本
Search   Find	Ctrl+F	字符查找
Search   Find Next	Ctrl+G	再次查找字符
Search   Replace	Ctrl+H	字符替换
Help   Help	F1	显示 MAXScript 在线帮助

(续表)

命令名称	快捷键	说明
Help   About MAXScript		显示 About MAXScript 提示信息
	Shift+Enter	执行选择的命令行, 如果没有选择文本, 则执行光标所在行
	Ctrl+右击鼠标	显示一个弹出菜单, 里面包含当前脚本中定义的所有工具(UTILITY)、结构(structure)、用户界面、函数、插件、工具、脚本宏、右键菜单, 选择其中一个可以把鼠标定位在定义该项的脚本起始处, 这点可以帮助我们快速定位, 如图 1-8 所示
	Ctrl+D	在 MAXScript Editor 窗口里执行一次语法着色, 每按一次 Ctrl+D, 窗口就被刷新一次。分别使用下面的颜色: 注释用绿色, MAXScript 保留字用蓝色, 其他字符串用浅红色。这在阅读大型、复杂的程序时很有帮助, 新输入的文本总是用黑色显示, 所以要用 Ctrl+D 快捷键重新着色。语法着色仅作为一个编程的辅助手段, 它不会影响脚本的执行
	Ctrl+R	定位鼠标至上一个单击的位置或查找操作的位置
	Ctrl+B	选择当前括号内的文本。这样可以帮助我们检查一段很长的源程序的括号匹配情况。括号可以是(), [], {}。将鼠标置于源程序任意位置, 按下 Ctrl+B, 如果鼠标处于一个括号之后, 那么从该括号开始直至与之匹配的括号结束的程序代码将被选择。如果鼠标没有紧跟在一个括号之后, 最近的一个括号对里包含的程序代码被选择; 第二次按下 Ctrl+B, 上一层括号对里的代码将被选择; 反复按下 Ctrl+B, 将遍历程序里所有的括号嵌套。如果在某一点发现括号不匹配, 系统会给出蜂鸣提示, 不选择任何文本

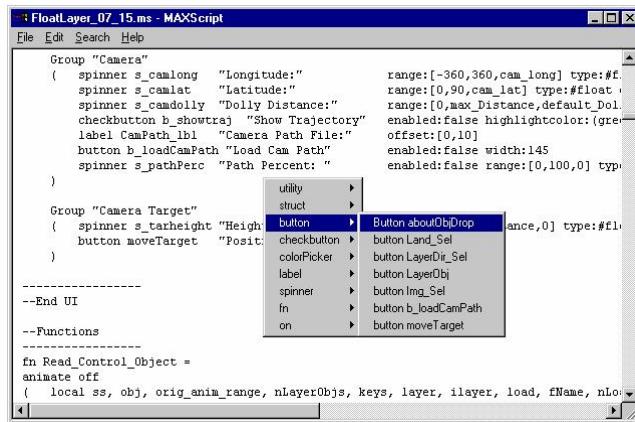


图 1-8 Ctrl+右击鼠标弹出窗口

## 1.6 MAXScript 桌面状态

MAXScript 可以保存上一次退出 3ds max 时的桌面状态, 包括活动 Editor 窗口和 Listener

窗口的位置，当用户再一次启动 3ds max 时，这些窗口会自动被恢复。桌面状态存储在一个名为 MAXScript.dsk 的文件里，该文件在 3ds max 的执行路径里，用户可以改变它的名字或以另外一个文件取代它，也可以简单地把它删除，那样桌面状态将不会被自动恢复。

## 1.7 快速学习 MAXScript 语言的两个方法

### 通过一个脚本文件学习 MAXScript

最好的学习 MAXScript 命令和语法的方法是在 MAXScript Editor 窗口中调入一个脚本文件，逐行运行它，然后观察 Listener 窗口和用户界面的反应：

- (1) 在 MAXScript Editor 窗口中打开一个脚本文件。
- (2) 把光标放在第一个命令行，按下数字键盘区的 Enter 键运行当前行，输出信息会在 Listener 窗口的输出区域显示。
- (3) 移动鼠标至下一行，重复步骤 (2)。

### 通过宏记录器 (MacroRecorder) 学习 MAXScript

如果你想知道 MAXScript 怎样执行一个任务，你可以从 MacroRecorder 开始。宏记录器捕捉绝大多数命令执行的操作，并产生与这些操作相应的 MAXScript 命令。宏记录器输出在 MAXScript Listener 窗口的粉红色区域。

## 1.8 一个简单的 MAXScript 例子

通过前面的讲述，读者应该对 MAXScript 脚本语言已经有一个比较初步的认识了，通过下面的例题将对 MAXScript 语言如何创建和修改场景对象有一个比较具体的了解。这些例题中有一些内容可能读者觉得很陌生，不要紧，这些内容在以后的章节里会慢慢涉及到。

### 1.8.1 用 MAXScript 创建一个简单的 Box 对象

用 MAXScript 可以创建常见的对象，如 Box 和 Cylinder 等，比如在 Listener 窗口输入：

```
Box()
```

将在 3ds max 视窗里用参数默认值创建一个 Box 对象。也可以将该对象赋给一个变量，以便在随后的运算中引用它：

```
myBox=Box()
```

**注意** 如果在创建对象时不指定任何参数，必须在函数名后加一对括号 “()”，这样 MAXScript 才能知道使用参数的默认值来创建对象，如果在调用函数时指定了一个或多个参数的值，则不需要再在后面加括号了，如：

```
myBox=Box length :20 width :20 height :20
```

请读者注意上面参数指定的格式：参数名后跟一个冒号“：“，再接着参数值，有关参数调用请读者参见本书 6.3 节。

对上面的语句，Listener 窗口在输出区域里返回下面的语句：

```
$Box:Box01 @[0.000000,0.000000,0.000000]
```

该语句的第一部分为路径名。路径名与 Windows 系统里的路径名相似，如：“C:\3ds max\examples\file.max”，表示一个特定文件所处的路径层级结构。在 MAXScript 里采用相似的路径概念，但是 MAXScript 的路径指向一个特定的对象，而不是文件。路径名总是以“\$”开始，更详细的有关路径名的信息，请参见 3.5.2 节。

该语句的第二部分是对象名：Box01。当用户用 Select by Name 工具栏选择对象时，出现在 Select Object 对话框里的是该对象名称 Box01，而不是变量名 myBox。

方括号里的数字代表 Box 几何中心的 x、y、z 坐标。

这时在视窗里会画出该 Box 对象，如图 1-9 所示。

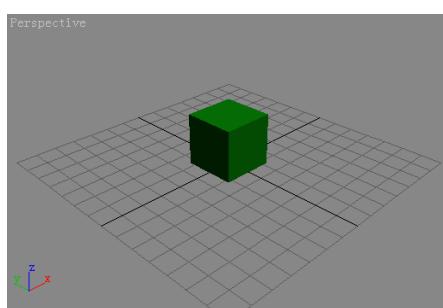


图 1-9 创建 Box 窗口

### 1.8.2 修改 Box 对象

在上一节里，我们创建了一个 Box 对象，并把它赋给了变量 myBox，现在我们能很容易地获取 Box 对象的所有属性，获取属性的方法是在变量名后加一个点“.”号，再紧跟着属性名，如 myBox.height，可以看作对象 myBox 的高度属性。

对象属性有以下几类：创建类属性，如 Box 对象的.height、.width、.length，Circle 对象的.radius；变形类属性，如.scale、.rotation、.position；通用属性，如.wireColor 等。如果想改变对象属性，用户可以给它们赋一个新值。如果不喜欢单独的名字 Box01，可以重新给.name 属性赋值，在 Listener 窗口输入：

```
myBox.name= "Blue Box"
```

#### 获取对象的参数

在 Modify 面板的 Parameters 卷展栏里，用户可以看到对象的所有创建参数，并可以改变它们。但如果我们希望在 MAXScript 里改变这些参数，就需要先知道这些参数的语法，有两种方法可以获得对象的所有参数设置。

##### 1. showClass()函数

showClass()函数可用来显示指定类的全部属性或指定属性的数据类型。格式为：

```
showClass <pattern_string>[ :<stream> ]
```

其中参数<pattern\_string>为一个可以包含通配符的字符串，可以为3ds max的类名、超类名和属性名，其格式为：

```
"<class_name>[ :<superclass_name> ][ .<property_name> ] "
```

可选参数:<stream>必须指定一个stream类（数据流）值，用来指定输出信息显示在哪里。例如：

```
showClass "Box.*"
```

MAXScript会在Listener窗口的输出区域里显示Box对象的所有属性。这些属性与用户在Modify面板的Parameters卷展栏里看到的属性相对应，内容包括Box对象所属的类和属性的数据类型，如图1-10所示。

我们可以用匹配符“\*”来一次查看多个类的参数，如果输入：

```
showClass "Box*.*"
```

MAXScript会显示Box和Box Gizmo两个类，如图1-11所示。

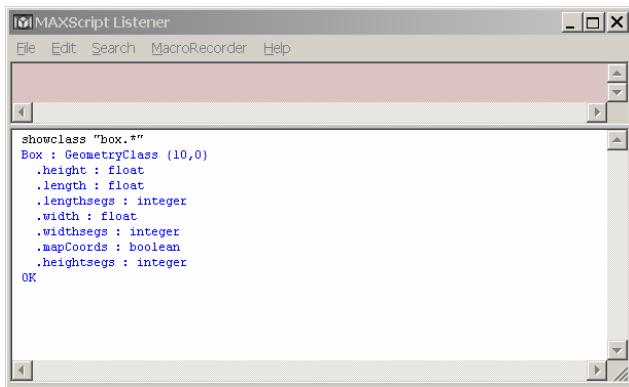


图1-10 showClass "Box.\*" 窗口

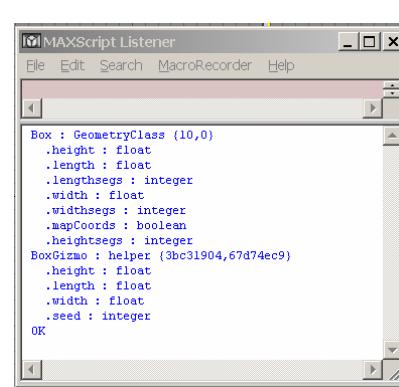


图1-11 showClass "Box\*.\*" 窗口

下面列出了该函数的一些其他用法，读者可以在Listener窗口里输入这些命令来查看其作用：

showClass "path*"	--显示所有类名以path开头的类
showClass "noise.*"	--显示noise类贴图所有可存取的属性
showClass "*:mod*"	--显示所有modifier类
showClass "*.*:rad*"	--显示拥有属性名中包含字符rad的所有类
showClass "*.*" to:f	--显示所有类、超类、属性，并输出到指定文件
showClass "*:*controller*"	--显示所有在其超类名中包含字符controller的类

## 2. showProperties()函数

showProperties()函数用来显示某一特定MAXWrapper类对象的属性。如果在场景里有一个具体的对象，可以用本函数来取代showClass()函数。和函数showClass()不同，本函数会显示“动态属性”——这些属性存在于具体的某一对象中，如一个List\_Controller里的次级Controller，或一个FFD\_Modifier里被设置动画的控制点。

showProperties()函数的调用格式如下：

`showProperties <maxwrapper_object> [ <property_pattern> ] [ to:<stream> ]`

参数说明如下：

- ◆ <maxwrapper\_object> 要检查的 3ds max 对象。
- ◆ <property\_pattern> 可选参数，要检查的属性名通配符。
- ◆ to:<stream> 可选参数， stream 类值，指定要将显示结果输出到哪里。

如果没有指定属性名通配符参数<property\_pattern>，显示指定对象的所有属性。例如：

```
showProperties $foo.bend      --显示对象 foo 的 Bend_modifier 的属性
ffdmod = $baz. 'FFD_Box_4x4x4' --指向对象 baz 的 FFD_modifier
showProperties ffdmod "disp*" to:log
                                --显示 FFD_modifier 里以 disp 开头的属性
showProperties $foo.pos.controller
                                --显示 position 类 controller 里的次级 controller
```

**注意** 如果参数<maxwrapper\_object>指定的是一个场景对象，如\$Box01，本函数仅显示基对象的属性，它不会显示 Transform 类 Controller、Modifier 和赋给对象的 Material 等属性。如果要显示这些属性，必须采用如上面例子所示的形式。

请读者参考下面的例子：

```
b=Box()                      --创建一个 Box 类对象
ffd_mod=ffdBox()              --创建一个 ffdBox_modifier
addmodifier b ffd_mod         --将 ffdBox_modifier 应用到 Box 对象
showproperties b               --显示 Box 对象的所有属性
showproperties ffd_mod        --显示 ffdBox_modifier 的所有属性
```

输出：

```
$Box:Box07 @ [ 0.0000,0.0000,0.0000]      --第 1 行输出结果
FFD_Box__4x4x4:FFD(Box)4x4x4           --第 2 行输出结果
OK                                         --第 3 行输出结果
.height : float                          --第 4 行输出结果的开始
.length : float
.lengthsegs : integer
.width : float
.widthsegs : integer
.mapCoords : boolean
.heightsegs : integer
OK                                         --第 4 行输出结果的结束
.dispLattice(Lattice) : boolean          --第 5 行输出结果的开始
.dispSource(Source Volume) : boolean
.deformType(<unknown>) : integer
.falloff : float
.tension : float
.continuity : float
.inPoints(Inside Points) : boolean
.outPoints(Outside Points) : boolean
.Offset : float
.lattice_transform : transform
```

OK

-- 第5行输出结果的结束

### 改变对象的属性

我们可以通过给属性赋值的方式来改变对象的属性。

比如，创建对象时颜色是随机产生的，我们为了把它的颜色改为同.name 属性匹配，在 Listener 窗口输入：

```
myBox.wireColor =blue --可以看到视窗里 Box 对象的颜色变成蓝色了
```

blue 是 MAXScript 里预先定义好的颜色常量，其 RGB 值为 0、0、255。其他预先定义的颜色常量还有 red、green、white、black、orange、yellow 和 brown，除了使用颜色常量以外，用户也可以用 RGB 值指定对象的颜色，例如：

```
myBox.wireColor=(color 0 0 55)
```

Box 对象的位置属性为.pos，在 Listener 窗口中输入：

```
myBox.pos=[0,-75,0]
```

即可改变对象的位置。

用下面的命令可以让 Box 对象变大：

```
myBox.scale=[1.5,1.5,1.5]
```

.scale 属性需要 X、Y、Z 三个方向的缩放比例，分别对应宽度、长度和高度三个方向，三个方向的缩放比例可以不一致。当用户对上面的 Box 对象进行缩放之后，它的.width、.length、.height 属性仍然为 20。尽管下面的命令：

```
myBox.width=60  
myBox.height=40
```

与命令：

```
myBox.scale=[1,3,2]
```

从表面上看都把 Y 方向放大了 3 倍，Z 方向放大了 2 倍，但改变.height 属性会改变对象的创建参数，而改变.scale 属性不会引起创建参数的改变。

### 1.8.3 Box 对象的标准转换

Box 对象的标准转换包括：move（移动）、scale（缩放）、rotate（旋转）。

#### move（移动）

格式：move <name\_obj> [(x,y,z)]

其中参数<name\_obj>为对象名，参数(x,y,z)为移动的距离。注意这个函数并不是把对象移到坐标点(x,y,z)，而是把对象在 x、y、z 方向各移动变量(x,y,z)指定的偏移量，如果用户不止一次执行上面的命令，对象会不停地移动，这一点也同样适用于 scale 和 rotate 转换，如果我们输入：

```
move myBox [10,0,0]
```

```
move myBox [10,0,0]
move myBox [10,0,0]
```

将把 Box 对象沿 x 轴移动 30。

### scale (缩放)

格式: scale <name\_obj> [(x,y,z)]

其中 name\_obj 为对象名, (x,y,z) 为一个三维坐标, 分别代表 x、y、z 轴三个方向的缩放因子。同样 scale 变形与对象的 .scale 属性不同, 重复赋给 .scale 属性相同的值只会在第一次赋值时改变对象的 .scale 属性, 而重复对对象施加相同的 scale 转换却会导致对象不断地缩放。

### rotate (旋转)

旋转转换不像 move 和 scale 转换那样简单, 在 MAXScript 里, 有三种方法旋转一个对象:

- ◆ 欧拉角方法 (euler angel)
- ◆ 四分角方法 (quaternion)
- ◆ 角向轴方法 (angleaxis)

这里仅讨论欧拉角方法, 为了执行一个旋转变形, 必须先把旋转定义为一个旋转对象 (rotation object), 然后把这个旋转对象应用到要旋转的对象上。旋转对象的定义格式如下:

```
rot_obj=eulerangles x y z
```

其中 rot\_obj 是旋转对象名, x、y、z 分别为绕 x、y、z 轴旋转的角度, 以度为单位。例如:

```
rot_obj=eulerangles 0 30 0
```

创建一个名为 rot\_obj 的旋转对象, 它将绕 y 轴旋转 30°。我们可以把这个旋转对象应用到 Box 对象上:

```
rotate myBox rot_Box
```

可以在视窗中看到 Box 对象绕 y 轴旋转了 30°。如图 1-12 所示。

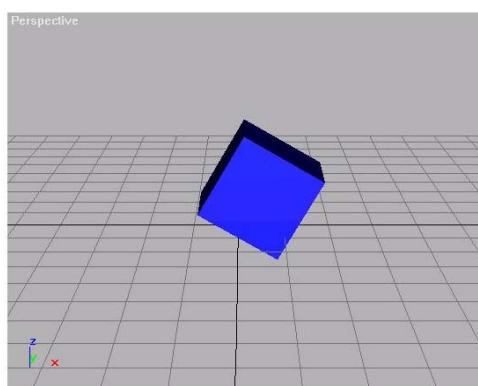


图 1-12 Box 对象绕 y 轴旋转了 30°

#### 1.8.4 Box 对象的更多转换

在 MAXScript 里创建一个 Modifier (修改器) 是非常简单的, 格式如下:

```
addmodifier obj_name (modifier_name<parameters>)
```

如对 Box 对象创建一个扭曲 30° 的 Modifier:

```
addmodifier myBox (Twist angle:30)
```

这时视窗里可以看到 Box 对象扭曲了 30°, 如图 1-13 所示。在用户界面里还可以看到修改器堆栈 (Modifier Stack) 里增加了一个 Twist 项, 如图 1-14 所示。

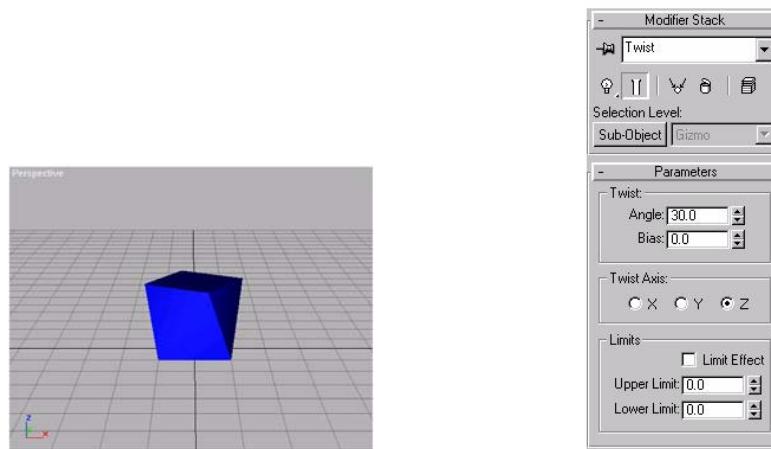


图 1-13 Box 对象扭曲了 30°

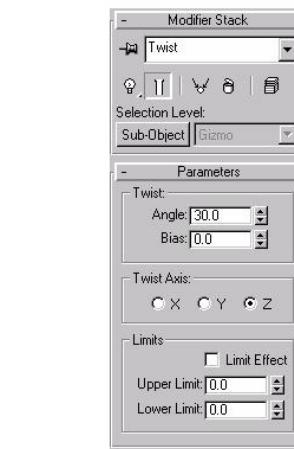


图 1-14 修改器堆栈里增加了 Twist 项

调整 Modifier 参数和调整创建参数类似。

例如:

```
myBox.twist.angle=60
```

将 Box 对象扭曲 60°。

用户还可以给对象加入别的修改器, 修改器的内容详见本书第 10 章。

#### 1.8.5 为 Box 对象创建动画

我们先来看一个简单的用 MAXScript 创建的动画:

```
animate on
(
at time 0(myBox.pos=[-100,0,0]; myBox.scale=[1,1,0.25])
at time 100(myBox.pos=[100,0,0]; myBox.scale=[1,1,3])
)
```

在用户界面里拖动时间滑杆, 可以看到 Box 对象会动起来, 关键帧设置在第 0 帧和第 100 帧。当我们在 MAXScript 里动画 Box 对象时, 在 3ds max 主界面里的 Animate 按钮并没有被打开, 时间滑杆也没有被移动, 这一切都发生在 MAXScript 内部。在上面例子中, 时间被指定为一个简单的数字, 如果没有指定单位, MAXScript 将它解释为帧的顺序号。

用户也可以用下面的格式指定时间：

```
2.5s    --2.5 秒  
20f    --20 帧  
4800t   --4800ticks=1 秒  
1m3s5f  --1 分 3 秒 5 帧  
1:15.5  --1 分 15 秒 5 帧
```

## 第2章 MAXScript语言基础

本章主要介绍关于MAXScript语言脚本源程序编写的一些基础知识。

### 2.1 脚本文件里命令的求值

如果手工向MAXScript Editor窗口里输入脚本命令，可以用下面的方法验证输入脚本的正确性：

如果Listener窗口没有打开，首先在MAXScript卷展栏里按Open Listener按钮，然后在MAXScript Editor窗口里，选择File | Evaluate all。

MAXScript会对脚本文件里的所有命令逐行求值，并把反馈信息输出在Listener窗口的输出区域中，同时MAXScript也把有效的命令输送到3ds max视窗，如果脚本命令中包含语法错误，MAXScript会在Listener窗口中用红色字符标出。

### 2.2 把脚本文件包含在另一个脚本文件中

MAXScript支持一种运行时的源文件包含机制，我们可以把一个已经写好的脚本文件包含到一个新的脚本文件中。命令如下：

```
include "filename_string"
```

文件名只能用字符串指定，不能用变量或表达式。例如：

```
utility foo "baz"
(
local a,b,c
include "foo_ui.ms"
Rollout bar "Bar"
(
include "bar_Rollout.ms"
)
include "foo_handlers.ms"
)
```

可以把包含文件语句放在任意嵌套层次，被包含的文件还可以包含别的文件。因为include是一个运行时的构造函数，被包含文件的变量作用域仅限于该文件，这一点与fileIn()方法不同。

我们可以用包含文件语句include <file>来作为表达式的一部分，例如：

```
include "opl.ms" + include "op2.ms"
if include "test2.ms" then print a else print b
```

## 2.3 向 MAXScript 里输入数据信息

用户可以向 MAXScript Listener 窗口输入的数据类型有：数字、字符、数组。

### 数字

MAXScript 有两种数字类型：整型（Integer）和浮点型（Float）。当 MAXScript 执行一个数字型的运算时，运算结果的数据类型通常和输入的操作数类型一致，如  $3+4$  得到 7，但  $3.0+4.0$  得到 7.0。但当 MAXScript 在两种不同类型的数据间进行运算时，得到的数据类型总是浮点型，如  $3+4.0$  得到 7.0。

### 字符

MAXScript 是一种基于表达式的语言，它的每一个表达式都会返回一个值，并在 Listener 窗口的输出区域里输出。在 MAXScript 里，字符必须放在一对“””符号里，比如：在宏记录区里输入 Hello 然后回车，在输出区域里将会得到 undefined，因为系统把 Hello 作为一个变量，而该变量没有事先定义。正确的做法应该是：在宏记录区里输入 Hello，回车后，在输出区域里将看到蓝色的字符串 Hello，这就是刚才输入 Hello 的运算结果。

### 数组

数组是一组数据的集合，在 MAXScript 里，数组的每个元素可以是任何不同的数据类型，而且可以被单独获取。定义数组有两种方法：

第一种方法是：#()

定义一个空的数组。它表明定义一个数组必须用一个“#”符号和一对括号。

另一种方法是：#(<expr>, <expr>)

为一个数组定义几个初始的元素。每个<expr>都可以为数字、表达式、字符，元素间不必是同一种数据类型，且对元素的数目没有限制。

## 2.4 使用“？”号

MAXScript 每次对一个命令或命令集的求值，求值结果都被输出到 Listener 窗口的输出区域，并被存储在一个名为“？”的内部变量里。我们可以像普通的变量一样使用“？”来获取该求值结果。例如：

```
x=?
```

## 2.5 脚本运行过程的错误信息

如果 MAXScript 在脚本文件中检测到运行错误，将会在 Listener 窗口的输出区域显示一个错误信息和错误诊断信息。如果是控制器脚本或卷展栏脚本发生错误，错误信息将显示在一个警告窗口里。

错误诊断信息以回调跟踪（call stack trace-back）形式给出，这样有助于让用户知道错误的原因。回调跟踪将显示错误发生点的 MAXScript 函数嵌套，最深的嵌套层次在最开始，最外层的嵌套层次在最后。回调跟踪还会显示局部变量和函数的参数值。

如果运行代码来自一个源代码文件，作为一个进一步的错误定位辅助手段，MAXScript 会在 MAXScript Editor 窗口里打开这个文件，同时显示错误信息或提示框。此时提示框或 Listener 窗口将处在桌面的最前面，包含源代码的 MAXScript Editor 紧随其后。

所有编译和运行错误信息以注释符“--”开始，图 2-1 是一个运行出错的示例。

错误发生在 MAXScript Editor 窗口中亮显的一行。分析 MAXScript Editor 窗口中的错误信息，可知错误发生在当值循环变量 i 等于 6 时。进一步的分析会发现数组 b 仅有 5 个元素，这样 b[i] 的值实际上为 undefined，由此导致与之相连的 .position 属性找不到的错误。

图 2-2 是另一个运行时出错的例子，系统显示一个错误对话框和脚本控制器对话框。因为系统在运行脚本控制器中的脚本时发现了一个错误，这个错误是由于删除了对象 Box02，而它的 .position 属性正被脚本引用。

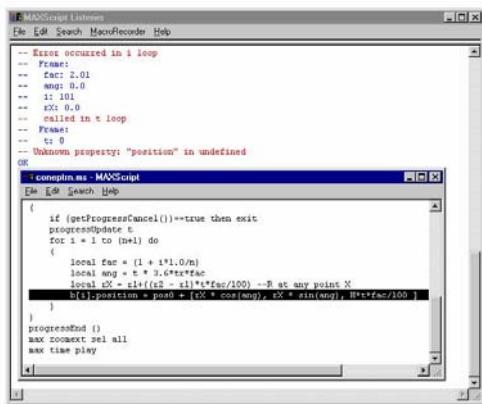


图 2-1 错误信息弹出窗口（一）

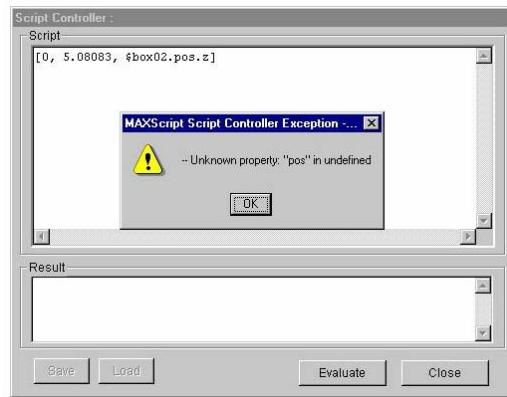


图 2-2 错误信息弹出窗口（二）

## 2.6 用 Esc 键中断程序运行

按 Esc 键可中止 MAXScript 脚本的运行。此时 MAXScript 也许需要一段时间来响应应该中止的操作，所以要一直按住 Esc 键直到系统作出响应。Esc 键会中止任何当前正在执行的代码，如果代码为脚本控制器，系统会弹出一个脚本控制器对话框，显示控制器使用的脚本，当用户单击对话框的 Close 按钮后，脚本控制器会重新开始执行。

MAXScript 全局变量 EscapeEnable 可用来设置是否允许用 Esc 键来中止程序的运行。

## 2.7 在 MAXScript 中使用 3ds max 命令

除了控制对象的造型和动画以外，MAXScript 脚本还可以调用 3ds max 菜单和工具栏，此时，必须以关键字 max 开始一条 MAXScript 命令，例如：

```
max file open
```

关键字后必须跟一个或几个描述命令的单词，我们可以用“？”号来查询可用的 3ds max 命令，例如：

```
max time ?      --显示所有与 time 有关的命令
max sel ?      --显示所有以子串 sel 开始的命令
max ?          --显示所有命令
```

下面是全部的 3ds max 命令列表：

3ds max 命令名	命令说明
max ?	在 Listener 窗口显示所有 max 命令
max accel pan	激活视窗 Pan 模式
max acthomegrid max activate home grid	将 Home Grid 设为活动网格
max activate grid object	将当前选择网格设为活动网格
max adaptive persp grid max adaptive perspective grid toggle	在非正交视窗里改变网格的外观
max align	单击 Align 按钮
max align normals max alignnormals	单击 Align Normals 按钮
max align camera	单击 Align Camera 按钮
max angle snap toggle	单击 Angle Snap toggle 按钮
max apply ik	单击 Hierarchy   IK 面板下 Apply IK 按钮
max array	显示 Array 对话框
max backface max backface cull toggle	为选择对象集打开 Backface Cull 开关
max background max background display toggle	显示 Viewport Background 对话框
max bind space warp mode max bindwsm	单击 Bind to Space Warp 按钮
max Box mode max Box mode selected	为选择对象集打开 Display as Box 开关

(续表)

3ds max 命令名	命令说明
max configure paths	显示 Configure Paths 对话框
max create mode	激活 Create 命令面板
max customize UI	显示 Customize User Interface 对话框
max cycle select	在三种区域选择方式之间循环：Rectangular、Circular、Fence。第一次调用本命令，区域选择方式为 Rectangular；第二次调用本命令，区域选择方式为 Circular；第三次调用本命令，区域选择方式为 Fence；第四次调用本命令，区域选择方式又变为 Rectangular，如此循环
max cycle sublevel max cycle subobject level	在四种子对象级别之间循环，调用本命令时必须处在 Modify 命令面板下，且可以选择子对象
max default lighting toggle max def lgt toggle	为视窗渲染指定默认视窗灯光还是场景灯光
max delete	删除选择对象或子对象
max display floater	显示 Display Floater 对话框
max display mode	打开 Display 命令面板
max dolly max dolly mode	单击 Zoom   Dolly 按钮
max drawingaids max drawing aids	打开 Grid and Snap Setting 命令面板
max fetch	单击菜单命令 Edit   Fetch，然后显示 About to Fetch 对话框
max file archive	单击菜单命令 File   Archive，执行一次场景文件存档操作
max file export selected	单击菜单命令 File   Export，然后显示 Select File to Export 对话框
max file import	单击菜单命令 File   Import，然后显示 Select File to Import 对话框
max file insert tracks	单击菜单命令 File   Merge，然后显示 Merge Animation 对话框
max file merge	单击菜单命令 File   Merge，然后显示 Merge File 对话框
max file new	单击菜单命令 File   New
max file open	单击菜单命令 File   Open，然后显示 Open File 对话框
max file preferences	单击菜单命令 Customize   Preferences，然后显示 Preference Settings 对话框
max file replace	单击菜单命令 File   Replace，然后显示 Replace File 对话框
max file save	单击菜单命令 File   Save
max file saveas	单击菜单命令 File   Save As，然后显示 Save File As 对话框
max file xref object	单击菜单命令 File   XRef Objects，然后显示 XRef Objects 对话框
max file xref scene	单击菜单命令 File   XRef Scenes，然后显示 XRef Scenes 对话框
max fov	单击 Field-of-View 按钮
max freeze inv	冻结未被选择的对象，就像单击 Display 面板下的 Freeze Unselected 按钮一样
max freeze Selection	冻结被选择的对象，就像单击 Display 面板下的 Freeze Selected 按钮一样
max fullinteract	无操作

(续表)

3ds max 命令名	命令说明
max grid nudge down	将活动 Grid 向下推动
max grid nudge up	将活动 Grid 向上推动
max grid toggle	在活动视窗里触发活动 Grid 的显示
max grids align	将活动 Grid 与活动视窗对齐
max group attach	进入 Attach Object to Group 模式，用户需要在场景里单击一个 Group，来指定选定对象要附属到哪一个组（Group）
max group close	关闭活动组（Group）
max group detach	将选定对象与组（Group）分离
max group group	组选对象，并显示 Group 对话框
max group open	打开选择的组（Group）
max group ungroup	将选择的组（Group）的分组拆开
max help about	单击菜单命令 Help   About，然后显示 About 3ds max 对话框
max hide camera toggle	触发 Display 面板 Hide by Category Rollout 里的 Cameras 复选框控件
max hide command panel toggle	触发命令面板的显示：如当前 3ds max 界面正显示命令面板，调用本命令会隐藏命令面板，再次调用本命令，命令面板会重新显示在 3ds max 界面上
max hide floating toolbars toggle	触发浮动工具栏的显示
max hide helper toggle	触发 Display 面板 Hide by Category Rollout 里的 Helpers 复选框控件
max hide inv	隐藏未被选择的对象，与 Display 面板的 Hide Unselected 按钮功能相同
max hide light toggle	触发 Display 面板 Hide by Category Rollout 里的 Lights 复选框控件
max hide main toolbar toggle	触发主工具栏的显示
max hide object toggle	触发 Display 面板 Hide by Category Rollout 里的 Geometry 复选框控件
max hide tab panel toggle	触发 TAB 面板的显示
max hide Selection	隐藏被选择的对象，与 Display 面板的 Hide selected 按钮功能相同
max hide shape toggle	触发 Display 面板的 Hide by Category Rollout 里的 Shapes 复选框控件
max hide system toggle	触发 Display 面板的 Hide by Category Rollout 里的 Particle Systems 复选框控件
max hide wsm toggle	触发 Display 面板的 Hide by Category Rollout 里的 Space Warps 复选框控件
max hierarchy mode	激活 Hierarchy 面板
max hold	单击菜单命令 Edit   Hold
max ik terminator	触发 Hierarchy 面板的 IK 选项卡 Object Parameters 卷展栏里的 Terminator 复选框控件
max ipan	进入交互 Pan 方式：将活动视窗的中心置于当前鼠标位置
max izoom in	进入交互 Zoom In 方式：将活动视窗在当前鼠标位置进行放大
max izoom out	进入交互 Zoom Out 方式：将活动视窗在当前鼠标位置进行缩小

(续表)

3ds max 命令名	命令说明
max key mode	触发 Key Mode Toggle
max link	打开 Link 模式
max load custom UI	单击菜单命令 Customize   Load Custom UI，并打开 Load UI File 对话框
max lock UI layout	触发 UI 布局的锁定
max material browser	显示 Material   Map Browser 对话框
max mirror	单击 Mirror 工具按钮，并显示 Mirror 对话框
max modify mode	打开 Modify 命令面板
max motion mode	打开 Motion 命令面板
max move	单击 Select and Move 工具按钮
max mtledit	单击 Material Editor 工具按钮，并显示 Material Editor 对话框
max next mod	在修改器堆栈里向上移动一级
max override	触发 Degradation Override，本命令仅对活动窗口有效，所以如果在 Listener 窗口里执行本命令，什么也不会发生
max pancamera	系统进入 Arc Rotate   Orbit 模式
max panview	系统进入 Pan   Truck 模式
max persp	系统进入 Zoom All   Perspective   Hotspot 模式
max prev mod	在修改器堆栈里向下移动一级
max preview	执行菜单命令 Rendering   Make Preview，并打开 Make Preview 对话框
max properties	显示 Object Properties 对话框
max quick render	单击 Quick Render 工具栏按钮
max redo	执行菜单命令 Edit   Redo
max renamereview	执行菜单命令 Rendering   Rename Preview，并显示 Save Preview As 对话框
max render last	单击 Render Last 工具栏按钮
max render scene	单击 Render Scene 工具栏按钮并显示 Render Scene 对话框
max reset file	执行菜单命令 File   Reset
max revert custom UI	执行菜单命令 Customize   Revert to Startup UI，并显示确认对话框
max rms	执行菜单命令 Edit   Edit Named Selections，并显示 Edit Named Selection 对话框
max roll	进入 Roll 模式，活动视窗必须为 Camera 或 Light 视窗
max rotate	单击 Select and Rotate 工具栏按钮
max rotateview	进入 Arc Rotate   Orbit 模式
max save custom UI as	执行菜单命令 Customize   Save Custom UI As，并显示 Save UI File as 对话框
max safeframe toggle	触发视窗的 Show Safe Frames
max saveplus	执行菜单命令 File   Save As

(续表)

3ds max 命令名	命令说明
max scale	单击 Select and Scale 工具栏按钮
max scale cycle	在三种缩放模式之间切换: Scale、Non-Uniform Scale、Squash
max select	单击 Select Object 工具栏按钮
max select all	执行菜单命令 Edit   Select All, 选择全部对象
max select by color	执行菜单命令 Edit   Select by Color, 然后用户选择一个对象, 所有具有与被选对象相同颜色的对象被选择
max select child	选择当前被选择对象的第一级子对象
max select invert	执行菜单命令 Edit   Select Invert
max select none	执行菜单命令 Edit   Select None
max select parent	选择当前被选择对象的父对象
max Selection floater	显示 Selection Floater 对话框
max set key keys	单击 Set Key 按钮, 使用 Character 下拉列表控件 (在 Filters...按钮的上面) 里的设置
max set key mode	单击 Set Key 按钮
max set key position filter	触发 Set Key filter 对话框里的 Rotation 复选框控件
max set key rotation filter	触发 Set Key filter 对话框里的 position 复选框控件
max set key scale filter	触发 Set Key filter 对话框里的 Scale 复选框控件
max set keys on selected	单击 Set Key 按钮, 但总是针对被选择的 Node 对象, 而忽略 Character 下拉列表控件的设置
max shade selected	执行菜单命令 Views   Shade Selected
max show last img	显示上一幅渲染图像
max showaxisicon	触发菜单命令开关 Views   Show Transform Gizmo
max showhomegrid	触发 HomeGrid 的显示
max snap toggle	触发 Snap Toggle 工具栏按钮
max spacebar	触发 Lock Selection Set 按钮。执行本命令仅对当前活动窗口有效, 所以如果在 Listener 窗口里执行本命令, 系统什么也不执行
max spacing tool	显示 Spacing Tool 对话框
max spinsnap toggle	触发 Spinner Snap Toggle 工具栏按钮
max subobject sel	触发 Modify 面板下的子对象按钮, 具体触发哪一按钮取决于系统变量 SubObjectLevel 的值
max texture correct	触发 Viewport Texture Correction 模式
max time back	将时间滑块移动到前一帧或关键帧, 相当于单击 Previous Frame   Previous Key 按钮
max time config	显示 Time Configuration 对话框
max time end	将时间滑块移动到最后一帧, 相当于单击 Go to End 按钮
max time forward	将时间滑块移动到后一帧或关键帧, 相当于单击 Next Frame   Next Key 按钮

(续表)

3ds max 命令名	命令说明
max time play	播放动画，相当于单击 Play 按钮
max time start	将时间滑块移动到第一帧，相当于单击 Go to Start 按钮
max toggle keyboard shortcuts	触发 Plug-in Keyboard Shortcut Toggle
max toggle ik	触发 IK Toggle
max toggle sound	触发 Sound Options 对话框里的 Audio Group
max tool animmode	触发 Animate 按钮
max tool center	在三种转换中心之间循环：Pivot Point（支点）、Selection（选择集中心）、Transform（转换矩阵）
max tool dualplanes	触发 Preference Settings 对话框 Viewports 选项卡下 Use Dual Planes 复选框控件
max tool hlist	单击 Select by Name 工具栏按钮，并显示 Select Objects 对话框
max tool maximize	将活动视窗最大/最小化
max tool x	单击 Restrict to X 工具栏按钮
max tool xy	单击 Restrict to XY 工具栏按钮
max tool y	单击 Restrict to Y 工具栏按钮
max tool z	单击 Restrict to Z 工具栏按钮
max tool zoom	单击 Zoom 按钮
max tool zoomall	单击 Zoom All 按钮
max tool zoomextents	单击 Zoom Extents 按钮
max tool zoomextents all	单击 Zoom Extents All 按钮
max tool zoomregion	单击 Zoom Region   FOV   FallOff 按钮
max trajectories	触发被选对象的轨迹显示
max treeview	执行菜单命令 Track View   Open Track View，并显示 Track View 对话框
max truck	进入 Pan   Truck 模式
max tti	执行菜单命令 Tools   Transform Type-In，并显示 Transform Type-In 对话框
max undo	执行菜单命令 Edit   Undo
max unfreeze all	解冻所有对象，相当于 Display 面板下的 Unfreeze All 按钮
max unfreeze by hit	单击 Display 面板下的 Unfreeze by Hit 按钮
max unfreeze by name	显示 Unfreeze Objects 对话框，相当于 Display 面板下的 Unfreeze by Name 按钮
max unhide all	单击 Display 面板下的 Unhide All 按钮
max unhide by name	单击 Display 面板下的 Unhide by Name 按钮，并显示 Unhide Objects 对话框
max unitsetup	执行菜单命令 Customize   Units Setup，并显示 Units Setup 对话框
max unlink	将被选对象与其父对象解除链接

(续表)

3ds max 命令名	命令说明
max utility mode	打开 Utility 命令面板
max videopost	执行菜单命令 Rendering   Video Post，并显示 Video Post 对话框
max view file	执行菜单命令 File   View File，并显示 View File 对话框
max view redo	执行一次视窗 Redo 操作
max viewpreview	执行菜单命令 Rendering   View Preview，并显示上一个预览图像
max views redraw	重绘所有视窗
max views undo	执行一次视窗 Undo 操作
max vpt back	将活动视窗设为 Back 视窗
max vpt bottom	将 Bottom 视窗设为活动视窗
max vpt camera	将活动视窗设为 Camera 视窗。如果场景里仅有一个 Camera 对象，或仅有多个 Camera 对象被选择，活动视窗使用该 Camera 对象；如果场景里没有一个 Camera 对象被选择，系统会显示 Select Camera 对话框；如果场景里没有 Camera 对象，系统会给出一个 No Camera in Scene 的警告信息
max vpt disable	触发活动视窗的 Disabled 状态，如果活动视窗处于 Disabled 状态，在视窗标题的后面会加上标识 “/ Disabled”
max vpt front	将活动视窗设为 Front 视窗
max vpt grid	将活动视窗设为 Grid 视窗
max vpt iso user	将活动视窗设为 User 视窗
max vpt left	将活动视窗设为 Left 视窗
max vpt persp user	将活动视窗设为 Perspective 视窗
max vpt right	将活动视窗设为 Right 视窗
max vpt shape	将活动视窗设为 Shape 视窗
max vpt spot	将活动视窗设为 Spotlight 视窗。如果场景里仅有一个 spotlight   directional 灯光，或仅有多个 spotlight   directional 灯光被选择，活动视窗使用该灯光；如果场景里有多个 spotlight   directional 灯光，且没有一个灯光被选择，系统会显示 Select Light 对话框。如果场景里没有 spotlight   directional 灯光，系统会给出一个 No Light in Scene 的警告信息
max vpt tab	在四个约束轴 (X、Y、Z 和 XY) 之间切换
max vpt top	将活动视窗设为 Top 视窗
max vpt track	将活动视窗设为 Track View 视窗
max vptconfig	执行菜单命令 Customize   Viewport Configuration，并显示 Viewport Configuration 对话框
max wire facet	将活动视窗的阴影模式设为 Facets + Highlights
max wire smooth	将活动视窗的阴影模式设为 Wireframe
max zoom in 2x	将活动视窗放大 2 倍
max zoom in 2x all	将所有视窗放大 2 倍

(续表)

3ds max 命令名	命令说明
max zoom out 2x	将活动视窗缩小 2 倍
max zoom out 2x all	将所有视窗缩小 2 倍
max zoomext sel	对被选对象执行 Zoom Extends 操作
max zoomext sel all	对被选对象执行 Zoom Extends All 操作

## 2.8 语法定义的格式说明

本书中记号和子语句的书写使用一套简写规则，该规则遵从标准 EBNF 规范（Extended Backus\_Naur Form）。规则中特殊字符的意义如下表所示。

书写规则	规则说明
[...]	括号中的内容为可选
(...   ...   ...)	从多项中选择一项
{...}	可以指定零次或多次括号 {} 里的内容
{...} +	可以指定一次或多次括号 {} 里的内容
:: =	为语法规则命名
(rule)	命名规则的内容
bold_characters	规则中直接使用的符号或字符

例如：

```
[-]{<digit>}[.{<digit>}][(e | E)[+ | -]{<digit>}]+]
```

上例为 MAXScript 中的十进制数的语法，其中包含了一些含有特殊意义的字符。其数字语法规格可解释为：

- ◆ [-]{<digit>} 可选的符号“-”，后跟零个或多个数字（整数部分）。
- ◆ [.{<digit>}] 一个可选的小数部分序列，由两部分组成：一个小数点，后跟零个或多个数字。
- ◆ [(e | E)[+ | -]{<digit>}]+] 一个可选的序列，由以下部分组成：e 或者 E 字符，后跟一个可选的“+”或“-”号，再后跟一个或多个数字。

我们可以把上面的语法规则赋给一个名字：

```
number ::= [-]{<digit>}[.{<digit>}](e | E)[+ | -]{<digit>}+
```

这样可以在另外的命令规则里引用它，如下：

```
mod <number1> <number2>
```

## 2.9 MAXScript 里的数学运算

和其他高级语言一样，在 MAXScript 里也可以进行数学运算，这些运算可以分成以下几类。

### 基本的数学运算

MAXScript 有一个内嵌的计算器功能，可以进行加、减、乘、除等基本的数学运算。它还能识别几个数学常量，如 Pi，比如为了计算一个半径为 2.5 的球体的体积，可以输入：

```
V=4/3*Pi*2.5^3
```

### 字符串运算

MAXScript 可以对字符串进行简单的运算，比如：

```
A = "MAXScript"  
B = "is fun"  
A+B
```

会在输出区域返回运算结果：

```
MAXScript is fun
```

### 其他运算

MAXScript 可以执行许多运算，包括三角函数（sin、cos）、超越函数（exp、log、sqrt）等。下面介绍最有用的两个函数：随机数函数和递增函数。

#### 1. 随机数函数

随机数是 MAXScript 里最有用的函数之一，格式如下：

```
random number1 number2
```

它生成一个大小处于 number1 和 number2 之间的随机数，生成随机数的数据类型和第一个变量相同。

#### 2. 递增函数

这是一种赋值的缩写形式，其语法为：

```
<destination> += <expr>  
<destination> -= <expr>  
<destination> *= <expr>  
<destination> /= <expr>
```

它们与下面赋值语句功能相同：

```
<destination>=<destination>+<expr>  
<destination>=<destination>-<expr>  
<destination>=<destination>*<expr>  
<destination>=<destination>/<expr>
```

## 2.10 源代码布局规则与注释

MAXScript 源代码布局规则与大多数编程语言不同，它拥有那些自由格式的语言如 C、C++的优点，但却不需要大量的标点符号，如分号、括号等。

MAXScript 允许用户在语句和表达式的任何需要的地方断开，另起一行继续输入，MAXScript 会自动判别一个表达式是否已经结束，如下面的算术运算语句：

```
A+b*c/d-e+f*g/h
```

用户可以在任意运算符后将语句分成多行，MAXScript 会在一行结束后继续读入下一行，直至它判定语句已经结束，例如：

```
A+b*c/d-e+
f*g/h
```

但如果用户按下面方式输入：

```
A+b*c/d-e
+f*g/h
```

就得不到正确的结果，因为系统认为第一行已经是一个完整的表达式，这时系统会在输出区对第二行给出一个语法错误的提示。

如果用户确实想在一个子表达式的结尾断开一个表达式，可以使用反斜杠 “\” 作为续行的标志，如上面的表达式正确的写法应该是：

```
A+b*c/d-e\
+f*g/h
```

一旦 MAXScript 遇到一个反斜杠 “\” 作为一行语句的结尾（注释语句除外），系统将继续读入下一行，这样就可以把一些长表达式或复杂的语句分成一行或几行来书写，以便于源程序的阅读。

MAXScript 同样允许把几个表达式或语句合并在一行里书写，这些表达式或语句之间用 “;” 隔开，比如：

```
A+b*c/d-e+f*g/h; A+b*c/d-e
```

有时为了增加脚本程序的可读性，我们希望在源程序中加入一些注释语句，在 MAXScript 里，一个以双连字符 “--” 开始的语句就是注释语句，一旦系统发现一个以 “--” 开始的语句，就停止解释执行后面的内容，例如：

```
A+b*c/d-e --MAX 的表达式
```

## 2.11 赋值

在 MAXScript 里，变量赋值的格式与其他编程语言一样，格式如下：

```
variable_name = variable_value
```

变量名 variable\_name 以字母或下划线（“\_”）开始，后面可以有任意数量的字符；变量值（variable\_value）可以是字符、数字或表达式，例如：

```
mystring = "This is my string."
```

MAXScript 不区分变量名里的大小写，如 Mystring 与 mySTRING 将被 MAXScript 认为是同一个变量。

## 2.12 属性、方法、操作符、字面常量

MAXScript 是一种内部面向对象的语言，在 MAXScript 中用到的数据都有一个类型，如 Integer（整型）、Matrix（三维矩阵）等。数据类型根据面向对象语言的规则，又叫做类（Class），而这些数据叫做对象。在本书中数据和对象是同一个意思，而所有对象是某一个类的特定例子。比如在 MAXScript 中 Box 是一个类，它不是一个在 3ds max 界面中显示的具体的盒子，而是定义了一个盒子具有的特征，这些特征包含了：一个 Box 对象的创建方法和 Box 对象的属性（如.height、.width、.length 等）以及可以对这个 Box 对象执行的操作（如移动、复制、删除等）。

面向对象编程的原则之一是：一个类内部如何运行，用户是不知道的。用户所有与类之间的交互都通过类的外部界面进行，而这些界面被分成下面几个范畴：

- ◆ 属性 可获取的类的参数。如 Box 类的.height、.width、.length 和 Sphere 类的.radius 都是属于属性的范畴。
- ◆ 方法 可以调用由某一类构造的对象的函数。移动或旋转一个 3ds max 对象、向一个 3ds max 对象添加一个修改器、获取一个 3ds max 对象的某一顶点的位置等都属于方法的范畴，在本书中“方法”和“函数”是意思相近的两个词，但又有所区别：方法特指参数为 3ds max 对象的函数，而通常所说的“函数”指那些参数为数字、字符等基本数据类型的函数，如三角函数、超越函数等。
- ◆ 操作符 为某一个类的数值定义的数学运算或别的操作符号，如+、-、\*、/ 等。
- ◆ 构造函数 创建各种对象的不同方法，比如：

```
Point3 0 0 0
<color> as Point3
```

是 Point3 类的两个构造函数，都可用来创建新的 Point3 类对象。

- ◆ 字面常量 所有类的对象的字面形式。如 [0, 0, 0] 就是 Point3 类的字面常量，Hello World 是 String 类的字面常量。

## 2.13 标识符

在 MAXScript 里，变量、函数、参数、属性等都有自己的名称，我们称之为标识符。标识符以字母、字符或下划线“\_”开始，可以包含任意数量的字母、数字、字符或下划线

“\_”。如下面几个标识符是合法的：

```
foo
bar123
this_is_a_very_long_identifier
_heresoneWithStudlyCaps
```

而下面几个是不合法的标识符：

3object	--第一个字符不是字母字符
pressed?	--“?”不是合法的字母数字字符
a big number	--空格不能出现在标识符中
seven(7)	--“(7)”不是合法的字母数字字符

与大多数编程语言不同，MAXScript 对标识符中字母的大小写不敏感，如下面标识符在 MAXScript 中是一样的：

```
BitmapTexture
bitmaptexture
BITMAPtexture
```

在语法定义中，标识符的名称为`<var_name>`，例如：

```
for <var_name> ( in | =) <sequence> ( do | collect) <expr>
```

在程序中标识符经常是作为变量和参数的名称，并代表变量或参数的当前值。MAXScript 也可以像字符和数字一样直接将标识符作为值使用，当使用标识符作为值时，应在前面加一个“#”，如`#<var_name>`。在语法定义中，标识符值的名称为`<name>`。

标识符值变量主要用于函数调用中的符号参数，许多内嵌函数都使用事先定义好的符号参数，如下面的函数：

```
setBeforeORT <controller> <ORT_type_name>
```

其中参数`<ORT_type_name>`必须是标识符类值：`#constant`、`#cycle`、`#loop`、`#pingpong`、`#linear`、`#relativeRepeat`。

调用该函数的例子如下：

```
setBeforeORT $Box01.position.controller #pingPong
```

有两点需要注意：

1. 将标识符置于一对单引号“‘’”里

允许在标识符里包含一些不合法的标识符字符，如空格、括号等。这样做主要用于那些包含特殊字符的对象类型和属性，格式如：

'<any_char_except_quote>'	用于标识符或属性名
#'<any_char_except_quote>'	用于标识符字面
'<any_char_except_quote>'	用于关键字参数

例如：

```
snow 'starttime(start)':5
b.'lifetime(life)'
$Box01.modifiers[ #'FFD 4x4x4' ]
```

## 2. 标识符中的空格

在许多的 3ds max 标识符中都包含空格，下划线“\_”可用来代替一个空格字符。下面语句与上面例子的最后一行是一样的：

```
$Box01.modifiers[#ffd_4x4x4]
```

## 2.14 保留关键字、标点、符号

### 2.14.1 保留关键字

在 MAXScript 里，用来进行开始语法定义的词，称为保留关键字，这些保留关键字不能用作变量名或标识符，否则系统会给出一个错误信息。

保留关键字	说明	保留关键字	说明
about	about 语句	not	逻辑表达式
And	and 逻辑表达式	of	case 表达式
animate	animate 语句	Off	预定义全局变量
as	数据类型转换操作符和方法	On	Rollout 子句
at	设置时间（关键帧）	or	逻辑表达式
by	for 循环	parameters	脚本插件子句
case	case 表达式	persistent	脚本插件全局变量
catch	try 表达式	plugin	脚本插件
collect	for 循环	rcmenu	鼠标右键脚本菜单
continue	跳过循环	return	Return 表达式
coordsys	coordsys 语句	Rollout	Utility 子句
do	while 和 do 循环	set	Sticky 关联表达式
else	if 表达式	struct	结构定义
exit	循环退出	then	if 表达式
fn	创建自定义函数	throw	try 表达式
for	for 循环	to	for 循环
from	from 作为保留字，但在目前版本中并未使用	tool	脚本鼠标工具
function	创建函数	try	try 表达式
global	指定变量作用域	undo	undo 子句
if	if 表达式	utility	脚本工具
in	for 循环	when	when 子句
local	指定变量作用域	where	for 循环
macroscript	定义脚本宏	while	while 和 do 循环
mapped	创建函数	with	循环退出
max	3ds max 命令		

## 2.14.2 标点、符号

标点、符号用来分隔或组合子语句，或是特殊的数学运算符。

标点、符号	说明
0	块表达式
+ - * / ^	数学表达式
+= -= *= /= =	数学赋值
--	注释语句
;	将同一行里几个语句分开
<eol>	文件末尾标记
,	用于分隔 Array 类数据的各元素
[ ]	用于 2D 和 3D 点字面常量
:	函数调用中可选参数指定
'	Name 类值
.	小数点
{ }	用于 BitArray 类值
#	用于 Array 类值
== !=	逻辑表达式
< <= > >=	逻辑表达式
?	上一次运算的结果
\$ ...	用于 PathName 类数据字面常量
..	用于 BitArray 类值
\	源代码续行

# 第3章 MAXScript 数据类型

脚本语言处理的对象是数据，在 MAXScript 里，数据被赋予一个类名：Value。数据也是所有 MAXScript 类的基础。每种高级语言都规定了它可以使用的数据类型，MAXScript 语言也有三种不同类型的数据类型：基本数据类型、特殊数据类型、高级数据类型。

## 3.1 数据的操作符和方法

在面向对象语言（OOP）里，每一类都定义了自己的外部界面，这些界面被划分成几大范畴：字面常量、构造函数、属性、操作符和方法。在下文关于操作符、方法的语法描述中，变量类型使用<type\_name>规则，指定了变量或属性允许的变量类型；用户也可以用表达式，只要表达式的求值结果为正确的数值类型。

### 3.1.1 操作符

<value> == <value>	比较是否相等
<value> != <value>	比较是否不相等
<value> as <class>	将数据转换成指定类型

### 3.1.2 方法

#### 1. print <value> [ to:<stream> ] [#noMap]

其中<stream> = (<filestream> | <stringstream> | <windowstream>)

功能：将指定的单个<value>打印到 Listener 窗口或指定的<filestream>、<stringstream>、<windowstream>中，并自动在后面加跟一个回车符。如果<value>是一个数据集合，而又没有指定#nomap，那么数据集合的所有成员会被打印在一一行里。除 BigArray 类数据以外，其他类型数据的打印结果可以直接由函数 readValue() 和 readExpr() 读取，这样如果将数据输出到一个文件里，可以很方便地阅读。如果用户想读入 3ds max R3 以前的数据，可以将系统变量 options.old\_PrintStyles 设为 True。

#### 2. format <format\_string> { <value> } [ to:<stream> ]

功能：将指定的一个或多个<value>打印到 Listener 窗口或指定的<filestream>、<stringstream>、<windowstream>中，并使用指定的格式<format\_string>作为输出模板。

MAXScript 用“%”号来代表一个要打印的数据值，例如：

```
format "name:% pos:%\n" obj.name obj.pos
```

会产生如下打印结果：

```
name:Box01 pos:[0,150.0,0.5]
```

format 方法不会自动在输出字符末尾加上回车符，这样用户可以用几个 format 语句来建立一个输出行。

### 3. classOf <value>

功能：返回<value>所属的数据类型，每一种数据都有它自己的类型。

**注意** 为了避免与 MAXScript 保留关键字 Rollout (作为 Rollout 子句定义的引导词) 相冲突，我们用 RolloutClass 作为 Rollout 类的类名，例如：

```
if classOf foo==RolloutClass
```

### 4. superClassOf <value>

功能：返回<value>的超级类，也即<value>所属类的父类。

### 5. isKindOf <value> <class>

功能：如果<value>继承自类<class>，则返回 True，否则返回 False。

#### 关于 ClassOf()、SuperClassOf() 和 isKindOf 的说明

下表表明了变量 b=Box()的类和超级类的层级结构。

	ClassOf (类)	SuperClassOf (超级类)
b	Box	GeometryClass
Box	GeometryClass	Node
GeometryClass	Node	MAXWrapper
Node	MAXWrapper	Value
MAXWrapper	Value	Value
Value	Value	Value

函数 isKindOf()和 ClassOf()有助于我们从一个对象集合进行过滤。例如下例中将仅把 Box 类型的场景对象收集到数组 allBoxes 中：

```
allBoxes=for obj in $* where(isKindOf obj Box)do collect obj
allBoxes=#()
for obj in $* do(if classOf obj == Box then append allBoxes obj)
```

### 6. isStructDef <value>

功能：如果<value>是一个结构定义，则返回 True。

### 7. isStruct <value>

功能：如果<value>是一个结构类的数据，则返回 True。

### 8. isController <value>

功能：如果<value>是一个控制器，则返回 True。

### 9. getHashCode <value> <Integer oldHashCode>

功能：如果指定值<value>为合法的数据类型，则返回一个整型 Hash 值，该值为指定

<value>和<Integer oldHashValue>的因子；如果<value>为非法的数据类型，则返回 undefined。合法的数据类型有：Float、Integer、String、BitArray、Point3、Ray、Quat、AngAxis、EulerAngles、Matrix3、Point2、Color 及元素为上述数据类型的数组。

## 3.2 基本数据类型

MAXScript 里基本数据类型有下面几种：

- ◆ Number（数字类）
- ◆ String（字符串类）
- ◆ Name（名称类）
- ◆ BooleanClass（布尔类）
- ◆ Color（颜色类）
- ◆ Point3（三维点类）
- ◆ Point2（二维点类）
- ◆ Ray（射线类）
- ◆ Quat（四元数类）
- ◆ AngleAxis（轴向角类）
- ◆ EulerAngles（欧拉角类）
- ◆ Matrix3（三维矩阵类）
- ◆ BigMatrix（大矩阵类）
- ◆ Box2（二维矩阵区域类）
- ◆ BitArray（位数组类）
- ◆ ArrayParameter（数组参数类）
- ◆ Time（时间类）
- ◆ Interval（间歇类）
- ◆ Bitmap（位图类）
- ◆ Stream（数据流类）

### 3.2.1 Number

MAXScript 有两种 Number 类数据类型：单精度浮点数（Float）和 32 位带符号整数（Integer），在 MAXScript 里，Integer 数的范围为：-2 147 483 648 到 2 147 483 647。如果计算结果超出该范围，会导致一个溢出错误，但 MAXScript 检测不到；Float 数绝对值范围为：1.18E-38 到 3.40E38，精度为 1.0E-7。如果计算结果的绝对值超出该范围，结果会是一个特殊的代表无限大的值：1.#INF。一个数与 1.#INF 进行加、减、乘运算结果仍是 1.#INF；一个数被 1.#INF 除，结果为 0。0.0 除以 0.0 或 1.#INF 除以 1.#INF、1.#INF 乘以 0 会得到一个特殊的代表不确定的值：-1.#IND，一个数与-1.#IND 进行+、-、\*、/运算，其结果都是-1.#IND。

## 字面常量

`[-]{<digit>}[.{<digit>}[(e | E)[+ | -]{<digit>}]+]  
0x{<hex_digit>}+`

十进制数  
十六进制数

例如：

```
123
123.45
-0.00345
1.0e-6
0x0E
0xFFE0
.1
```

## 操作符

`+ - * / ^ (乘方) - (负号)`

`== != > < >= <=`

`<number> as <class>` <class>可以是 Float、Integer、String、Time

## 方法

1. `copy <number>`

为指定<number>创建一份新的副本，本方法主要用来复制数组。

2. `abs <number>`

返回指定<number>的绝对值，返回结果的数据类型不变。

3. `mod <number1> <number2>`

求模计算，<number1>被<number2>除时的余数，返回结果总是 Float 型。

4. `ceil <number>`

返回与<number>最接近的、大于或等于<number>的整数，返回结果总是 Float 型。

5. `floor <number>`

返回与<number>最接近的、小于或等于<number>的整数，返回结果总是 Float 型。

6. `acos <number>`

7. `asin <number>`

8. `atan <number>`

9. `atan2 <number> <number>`

10. `cos <number>`

11. `cosh <number>`

12. `sin <number>`

13. `sinh <number>`

14. `tan <number>`

15. `tanh <number>`

标准三角函数，角度单位为度，返回结果总是 Float 型。

16. `exp <number>`

17. log <number>

18. log10 <number>

19. pow <number> <number>

20. sqrt <number>

标准超越函数，返回结果总是 Float 型。

21. random <number> <number>

返回两个变量之间的一个随机数，返回值的数据类型与第一个变量相同。

22. seed <number>

将指定数值<number>作为随机数生成函数 Random()的新根值（seed）。

23. degToRad <number>

将指定变量的单位从度转换为弧度，返回结果总是 Float 型。

24. radToDeg <number>

将指定变量的单位从弧度转换为度，返回结果总是 Float 型。

25. bit.and <integer1> <integer2>

返回整数<integer1>和<integer2>进行逐位和（AND）的运算结果，返回值为整数。

26. bit.or <integer1> <integer2>

返回整数<integer1>和<integer2>进行逐位或（OR）的运算结果，返回值为整数。

27. bit.xor <integer1> <integer2>

返回整数<integer1>和<integer2>进行逐位异或（XOR）的运算结果，返回值为整数。

28. bit.not <integer1>

返回整数<integer1>进行逐位非（NOT）的运算结果，返回值为整数。

29. bit.shift <integer1> <integer2>

返回整数<integer1>移位<integer2>的运算结果，返回值为整数。如果<integer2>为正数，移位为向左移动，如果<integer2>为负数，移位为向右移动。

30. bit.set <integer1> <integer2> <boolean>

将指定整数<integer1>的指定位<integer2>设置为指定布尔值<boolean>，<integer2>必须大于 1。

31. bit.flip <integer1> <integer2>

将指定整数<integer1>的指定位<integer2>的值反转，<integer2>必须大于 1。

32. bit.get <integer1> <integer2>

返回指定整数<integer1>的指定位<integer2>的状态，<integer2>必须大于 1。

33. bit.intAsChar <integer>

返回 ASCII 码为指定整数<integer>的字符。

34. bit.charAsInt <string>

返回指定字符串的第一个字符的 ASCII 码。

35. bit.intAsHex <integer>

返回 ASCII 码为指定十六进制整数<integer>的字符。

### 3.2.2 String

在 MAXScript 里，字符串的长度没有限制。

#### 字面常量

"<characters>"

String 类型为一个或多个包含在一对双引号 (" ") 里的字符。例如"3ds max", "October 5th, 1998", String 可包含除双引号之外的任何字符，对一些特殊的字符，可用下面方法输入：

\"	双引号"
\n	开始一新行
\r	回车键
\t	Tab 键
\*	一个星号
\?	一个问号
\\	一个反斜杠
\%	%
\x{d}	一个十六进制字符

例如语句：

```
Print "foo should be quoted like this : \"foo\" and a new line :\n"
```

输出如下：

```
"foo should be quoted like this: "foo" and a new line"
```

**注意** 如果后面没有跟上面指定的字符的反斜杠，“\”也被当作单个反斜杠 “\”。用户还可以把字符分成几行，回车键将保留在字符串中，例如：

```
"You can break a string literal  
into several lines and line break  
will stay in the string"
```

对 String 类数据，特别说明以下两点：

第一：用户可以用十六进制的 ASCII 码来在字符串中指定非打印字符，格式为：

\x{<hex.digit>}+

其中<hex.digit>是一个十六进制数，为 0~9、A~F。

例如：

```
str = "Copy Right \XA9 1997, Frab Works, INC"
```

其中 A9 为一个商标码的十六进制。

用下面的脚本可以在 Listener 窗口里输出与每个十六进制码对应的字符：

```
char = "0123456789abcdef"
```

```

for i=1 to char.count do
    for j=1 to char.count do
        s=execute("\\"+char[i]+char[j]+ "\\")
        format "%% %\n" char[i] char[j] s
    )
)

```

第二：在通常的文件路径名中，用“\”来将父路径与子路径分开，但在 MAXScript 中，因为“\”作为一个转码的标志，所以用“\\”来作为路径之间的分隔符，例如：

```
scene.path = "e:\\3ds max25\\scenes"
```

有时我们也可用“\”来代替“\\”，例如上例也可写成：

```
scene.path = "e:/3ds max25/scenes"
```

### 构造函数

<value> as string

可以将任何类的形式转化为字符串。

### 属性

<string>.count

返回字符串里的字符个数，只读整数。

### 操作符

1. <string> + <string>

返回连接两个字符串的新串。

<string> == <string>

<string> != <string>

<string> > <string>

<string> < <string>

<string> >= <string>

<string> <= <string>

比较语句，区分大小写。如果要进行一个不区分大小写的比较，先将 String 类值转换成 Name 类。

2. <string>[<index\_number>]

返回序号的单个字符，序号从 1 开始。

3. <string>[<index\_number>] = <single\_character\_string>

给序号字符赋新值。

4. <string> as <class>

类型转换<class>可以为 Name、Number、String、Stream 等，例如：

```
"Foo" as name
"123.4" as float
"$Box01" as String stream
```

## 方法

### 1. copy <string>

创建指定字符串的一份新副本，新副本包含的内容与源字符串相同，且与源字符串相互独立，例如：

```
newstr = copy oldstr
```

### 2. execute <string>

将指定字符串作为一个 MAXScript 表达式，对它进行编译，然后对它求值，并返回求值结果，例如：

```
execute "2 + 2"                                --返回 4
str = "$fooffd_2x2x2.control_point_" + n as string + " = [1,1,1] "
execute str
```

函数 execute()调用的返回值为参数<string>里最后一个表达式的求值结果。

在参数<string>里包含的变量的作用域为 Global，而不仅限于 execute()函数的执行范围。这样，如果在一个函数、脚本 Utility 或任何作用域不是 Global 的场合调用 execute()函数，用户必须显式地声明在 execute()函数里用到的变量的作用域。在下面的例子中，变量 posObj 的作用域为仅限于 Utility myUtil，所以需要显式地声明它的变量的作用域为 Local：

```
Utility myUtil "My Utility"
(
local posObj
fn test obj =
(
local i,j,thisCont
thisCont = execute("myUtil.posObj.' "+j+" 'pos.controller")
)
)
```

### 3. GetTextExtent <string>

返回一个 Point2 类值，表示如果指定字符串显示在命令面板里，占用屏幕的大小，单位为像素。

### 4. findString <string> <search\_string>

在字符串<string>里查找字符串<search\_string>，如果有找到，则返回第一个找到字符的序号，否则返回值 undefined。例如：

```
findString "Thanks for all the fish! " "all"      --返回 12
```

### 5. filterString <string> <token\_string>

将字符串<string>用<token\_string>指定的字符分解成一个字符串数组，每一个数组元素都为字符串<string>的一个子字符串。参数<token\_string>为一个分解字符列表。系统对字符串<string>进行扫描，每发现一个<token\_string>里的任何一个字符，都将作为一个子字符串的开始。例如：

```
filterString "MAX Script, is-dead-funky" " -"
```

将返回:

```
#( "MAX", "Script", "is", "dead", "funky")
```

#### 6. replace <string> <from\_integer> <length\_integer> <new\_string>

将字符串<string>里从序号<from\_integer>开始，长度为<length\_integer>的子字符串用新字符串<new\_string>替换，并返回新的字符串。<from\_integer>与<length\_integer>的和必须小于字符串的总长度。例如：

```
s = "1234567890"
s1=replace s 5 3 "inserted string" --返回"1234inserted string890"
```

#### 7. substring <string> <from\_integer> <length\_integer>

返回字符串<string>里从序号<from\_integer>开始，长度为<length\_integer>的子字符串。<from\_integer>与<length\_integer>的和如果大于字符串的总长度，子字符串为指定位置到源字符串的结尾。如果<length\_integer>为负数，返回子字符串从指定位置到源字符串的结尾。例如：

```
s = "Balerofon"
ss = substring s 5 3           --返回"rof"
ss = substring s 5 -1          --返回"rofon"
ss = substring s 5 100         --返回"rofon"
```

#### 8. matchPattern <string> pattern:<pattern\_string> [ ignoreCase:<boolean> ]

如果字符串<string>与通配符 pattern:匹配，返回 True，否则返回 False。如果没有指定参数 ignoreCase:False，比较是大小写不敏感的。例如：

```
s = "text1"
matchPattern s pattern : "text?"           --返回 True
matchPattern s pattern : "T*"               --返回 True
matchPattern s pattern : "T*" ignoreCase:False --返回 False
matchPattern s pattern : "s*"               --返回 False
```

下例中包含了 String 类的各种语法。

```
fn uppercase instring =           --函数定义体开始
( local upper, lower, outstring --局部变量声明
upper = "ABCDEFGHIJKLMNPQRSTUVWXYZ"
lower = "abcdefghijklmnopqrstuvwxyz"
outstring = copy instring           --创建输入字符串的副本
for i=1 to outstring.count do    --对字符串里所有字符进行循环，循环增量为 1
( j=findString lower outstring[i]
if(j != undefined)do outstring[i]=upper[j]
) --检查 outstring 变量里序号为 i 的单个字符是否为小写，如果为小写，变量 j 为该字符
在变量 lower 里的序号，如果不是，变量 j 等于 undefined
Outstring           --如果字符在变量 lower 里找到了，将它用变量 upper 里相应的字符
进行替换。返回被替换后的字符串
)           --函数 uppercase 的结束
s1 = "AbCdEfGh"      --给变量赋字符串
s2 = uppercase s1     --调用函数 uppercase，将变量 s1 作为参数传递
```

```

if s1 == s2 do print "strings s2 and s3 are the same"
if s1 != s2 do print "strings s1 and s2 are different" --比较字符串
theObject = "sphere" --给变量赋字符串
theRadius = (random 10.100.)as string --将数字转换为字符串
myObject = execute(theObject + " radius:" + theRadius) --连接字符串
串并执行字符串

```

输出为：

Uppercase()	--函数定义结果
"AbCdEfGh"	--第 12 行结果
"ABCDEFGHI"	--第 13 行结果
undefined	--第 14 行结果：字符串不相等
"strings s1 and s2 are different"	--第 18 行输出
"strings s1 and s2 are different"	--第 18 行结果
"sphere"	--第 19 行结果
"75.4091"	--第 20 行结果
\$Sphere:Sphere01 @ [0.0,0.0,0.0]	--第 21 行结果

### 3.2.3 Name

Name 类主要用于函数参数传递中从一系列选择中指定一种。

**字面常量**

#<var\_name>

**构造函数**

<string> as name

**操作符**

<name> == <name>  
<name> != <name>  
<name> < <name>  
<name> > <name>  
<name> <= <name>  
<name> >= <name>

**注意** 比较不区分大小写。

下面例子包含了 Name 类的各种语法：

```

name1 = #Hello --给变量赋 Name 值
name2 = "HELLO" as name --将 String 值转换为 Name 值
if name1 == name2 do print "names are equal" --比较 Name 值
Box_props=getpropnames Box --获取 Box 类的属性
sort Box_props --将属性名排序

```

输出为：

```

#Hello                                --第 1 行结果
#Hello                                --第 2 行结果
"names are equal"                      --第 3 行输出
"names are equal"                      --第 3 行结果
-- 下面为第 4 行和第 5 行的结果
#(#height,#length,#lengthsegs,#width,#widthsegs,#mapCoords,
#heightsegs)
#(#height,#heightsegs,#length,#lengthsegs,#mapCoords,#width,
#widthsegs)

```

### 3.2.4 BooleanClass

Boolean 值用于只能有两种状态中的一种的数据。

#### 字面常量

True、False 或 On、Off。

#### 操作符

not <boolean>	逻辑非
<boolean> and <boolean>	逻辑和
<boolean> or <boolean>	逻辑与

下面例子包含了 Boolean 类的各种语法:

```

bool1=True                           --给变量赋 Boolean 值
bool2=On
if bool1 and bool2 do print "booleans are equal" --比较 Boolean 值
fn xor b1 b2 =(not(b1 and b2))and(b1 or b2)
--定义函数 xor, 只有在输入的两个参数中只有一个为 True 时返回 True
xor False False                     --调用 xor 函数
xor False True
xor True False
xor True True

```

输出为:

```

True                                --第 1 行结果
True                                --第 2 行结果
"booleans are equal"                --第 3 行输出
"booleans are equal"                --第 3 行结果
Xor()                               --函数定义
False                               --第 6 行结果
True                                --第 7 行结果
True                                --第 8 行结果
False                               --第 9 行结果

```

### 3.2.5 Color

#### 字面常量

预定义的 MAXScript 系统变量 red、green、blue、white、black、orange、yellow、brown、gray 都是 Color 类值。

#### 构造函数

1. Color <r> <g> <b> [<a>]

r、g、b 和可选的 alpha 通道值都为 Float 数，取值范围为：0~255。

2. <Point3> as Color

<Point3>的 x 作为 r 分量，y 作为 g 分量，z 作为 b 分量。

#### 属性

<color>.red 或<color>.r

<color>.green 或<color>.g

<color>.blue 或<color>.b

<color>.alpha 或<color>.a

以上均为 Color 类值的成员属性，数据类型为 Float 类。

<color>.hue 或<color>.h

<color>.saturation 或<color>.s

<color>.value 或<color>.v

以上均为 Color 类值的派生属性，数据类型为 Float 类。

#### 操作符

<color> == <color>

<color> != <color>

<color> + <color>

<color> - <color>

<color> \* <color>

<color> / <color>

上面运算都是通道对通道的数学运算，各通道的值应该在 0~255 范围，如果超出该范围，3ds max 会将该值调整到 0~255。

-<color>

#### 方法

1. copy <color>

创建指定 Color 值的副本，例如：

```
newClr = copy oldClr
```

新副本包含的内容与源 Color 值相同，且与源 Color 值相互独立。

## 2. random &lt;from\_color&gt;&lt;to\_color&gt;

返回两个 Color 值之间的随机数，返回值的 alpha 分量总是为 255。

## 3. composite &lt;color1&gt; &lt;color2&gt;

将 Color 值<color1>和<color2>合成，本函数相当于下面的运算：

```
color1 + color2*((255. - color1.alpha)/255.)
```

## 4. 三维噪波函数

noise3 <color>

noise4 <color> <phase\_float>

turbulence <color> <frequency\_float>

fractalNoise <color> <H\_float> <lacunarity\_float> <octaves\_float>

参见 Point3 类值里的噪波函数描述。

下例包含了 Color 类的各种语法。

```
--用构造函数创建一个 Color 值
magenta=color 255 255 0 255
aqua = [0, 255, 255] as color
aqua.v /= 2.
aqua
aqua.alpha=128
aqua
lightGray=white/4
composite aqua lightGray
random black white
```

--创建白光，将 White 除以 4  
--合成 lightgray 和 aqua  
--生成一个随机 Color 值

输出为：

(color 255 255 0)	--第 2 行结果。
(color 0 255 255)	--第 3 行结果
127.5	--第 4 行结果
(color 0 127.5 127.5)	--第 5 行结果
128	--第 6 行结果
(color 0 127.5 127.5,128)	--第 7 行结果
(color 63.75 63.75 63.75)	--第 8 行结果
(color 31.75 159.25 159.25 159.75)	--第 9 行结果
(color 170.944 109.543 74.1957)	--第 10 行结果

## 3.2.6 Point3

Point3 类定义了三维空间的点，也称为三维矢量，包含三个 Float 类数。

## 字面常量

[<expr>, <expr>, <expr>]

其中<expr>是求值结果为整数或浮点数的表达式。例如：

```
[320, 240]      --2 维点
[10, 20, 30]    --3 维点
```

## 构造函数

```
Point3 <x> <y> <z>
<color> as Point3
```

## 属性

<Point3>.x	x 坐标
<Point3>.y	y 坐标
<Point3>.z	z 坐标

## 操作符

```
<Point3> == <Point3>
<Point3> != <Point3>
<Point3> + <Point3_or_number>
<Point3> - <Point3_or_number>
<Point3> * <Point3_or_number>
<Point3> / <Point3_or_number>
```

标准矢量运算，如果第二个操作数数据类型为 Number，操作对每一元素进行。

```
<Point3> * <matrix3>    坐标转换
<Point3> * <quat>        旋转<Point3>
<Point3> - <Point3>
```

## 方法

1. `copy <Point3>`

创建指定 Point3 值的副本，例如：

```
newPos = copy oldPos
```

新副本包含的内容与源 Point3 值相同，且与源 Point3 值相互独立。

2. `length <Point3>`

返回指定矢量的长度。

3. `dot <Point3> <Point3>`

返回两矢量的点矢积。

4. `cross <Point3> <Point3>`

返回两矢量的十字矢积。

5. `normalize <Point3>`

返回<Point3>对应的标准矢量，其矢量长度为 1。

6. `distance <Point3> <Point3>`

返回指定两点之间的距离。

7. `random <Point3> <Point3>`

返回指定两点之间的一个随机点。

## 8. arbAxis &lt;Point3&gt;

返回一个 Matrix3 值，代表使用指定点作为“向上”方向的任意坐标系。

## 9. matrixFromNormal &lt;Point3&gt;

返回一个 Matrix3 值，代表使用指定点作为 Z 轴方向的坐标系。返回 Matrix3 值的 Translation 分量为 [0,0,0]。例如：

```
MatrixFromNormal [0,0,1]
MatrixFromNormal [0,1,1]
```

返回值为：

```
(matrix3 [1,0,0], [0,1,0], [0,0,1], [0,0,0])
(matrix3 [0,-0.707107,0.707107] [1.41421,0,0] [0,1,1] [0,0,0])
```

以下为三维噪波函数。

## 10. noise3 &lt;Point3&gt;

一个三维空间浮点噪波函数，执行一个三次准随机曲线。返回值范围为 [-1.0 1.0]。如果用同样的<Point3>参数多次调用本函数，一般都会返回相同的值。

## 11. noise4 &lt;Point3&gt; &lt;phase\_float&gt;

与 noise3() 函数功能相同，但可以通过第二个参数<phase\_float>来指定相位，返回值范围为 [-1.0 1.0]。

## 12. turbulence &lt;Point3&gt; &lt;frequency\_float&gt;

本函数是一个建立在 noise3() 函数之上的简单分形循环紊乱噪波。第二个参数 <frequency\_float> 用来控制噪波紊乱的频率，返回值范围为 [-1.0 1.0]。

## 13. fractalNoise &lt;Point3&gt; &lt;H\_float&gt; &lt;lacunarity\_float&gt; &lt;octaves\_float&gt;

本函数模拟三维空间布朗运动的分形噪波函数，产生均匀、各向同性的噪波，返回一个 Float 类值。各参数说明如下：

- ◆ Point3 计算噪波的空间点。
- ◆ H\_float 分形增量，取值范围 [0 1]。当 H\_float 等于 1 时，函数相对平滑，当为 0 时，函数波动很激烈。
- ◆ lacunarity\_float 连续两个频率之间的间隙，最好设为 2.0。
- ◆ octaves\_float 函数中频率的数目。

下面例题包含了噪波函数的各种语法。

```
b_width=320           --指定位图尺寸
b_height=320
size=10.                --每一行、列覆盖的距离
z=0.                    --二维切片的 z 坐标
phase=0.5              --noise4 函数参数
frequency=10.            --turbulence 函数参数
fract_interval=.5       --fractalNoise 函数参数
lacunarity=2.
octaves=5
--设置噪波类型：1 = noise3; 2 = noise4; 3 = turbulence; 4 = fractalNoise
```

```

whichfunc=1
b=bitmap b_width b_height          --创建位图
for h=0 to(b_height-1)do           --对位图的每一行循环
(   h_norm=(h as float/(b_height-1))*size    --计算 y 坐标
row = for w=0 to(b_width-1)collect      --收集行像素颜色
(   w_norm=(w as float/(b_width-1))*size  --计算 x 坐标
noise_val = case whichfunc of         --根据噪波类型计算噪波
(   1 : noise3 [w_norm, h_norm, z]
2 : noise4 [w_norm, h_norm, z] phase
3 : turbulence [w_norm, h_norm, z] frequency
4 : fractalNoise [w_norm, h_norm, z] \
fract_interval lacunarity octaves
)
--将输出结果调整到[0.0 1.0]范围，并将它乘以颜色 white
noise_val = 0.5*(1.+noise_val)
white*noise_val
)
setpixels b [0,h] row             --将像素行存储到位图里
)
display b                         --显示位图

```

### 3.2.7 Point2

Point2 类定义了二维空间的点，本类是 Point3 类的二维特例，主要用来在 Rollout 中作为定位参数和存取 Bitmap 类值。

#### 字面常量

[<expr>, <expr>]

#### 构造函数

Point2 <x> <y>

Point3 as Point2 从<Point3>中取出 x、y 分量

#### 属性

<point2>.x

<point2>.y

#### 操作符

同 Point3 类。

#### 方法

1. copy <point2>

创建指定 Point2 值的副本，例如：

newPos = copy oldPos

新副本包含的内容与源 Point2 值相同，且与源 Point2 值相互独立。

2. random <point2><point2>

产生一个位于指定两点之间的随机点。

3. length <point2>

返回指定矢量的长度。

4. distance <point2> <point2>

返回指定两点之间的距离。

5. normalize <point2>

返回<point2>对应的标准矢量，其矢量长度为 1。

### 3.2.8 Ray

Ray 类定义了三维空间里的光线，一个 Ray 类值有两个 Point3 成员，第一个为光线起点，第二个为光线的矢量方向。

#### 构造函数

`ray <pos_Point3> <dir_Point3>`

`<node> as ray`

`<node>`为一个有目标对象的场景对象，返回一个 Ray 值，起点为`<node>`的`.position`属性所在的点，方向指向目标对象；如果对象没有目标对象，返回 undefined。

#### 属性

`<ray>.pos 或<ray>.position`

`<ray>.dir` 标准化的方向矢量

#### 方法

`copy <ray>`

创建指定 Ray 值的副本，例如：

`newRay = copy oldRay`

新副本包含的内容与源 Ray 值相同，且与源 Ray 值相互独立。

### 3.2.9 Quat

Quat 类提供了三维空间里方向的一种压缩形式，并提供了操作四元代数的方法。在 3ds max 中对象的旋转就是用四元数来存储的，一个四元数由四个子项组成，一个实标量指定了旋转的角度，一个虚矢量定义了旋转轴。

在两个关键帧之间用四元数进行旋转插值计算能得到平滑、自然的运动。

MAXScript 里角度的正方向遵从右手规则。

#### 构造函数

`quat <x_float> <y_float> <z_float> <w_float>`

x、y、z 值组成矢量部分，w 为围绕矢量的转角。

```
quat <degrees_float> <axis_Point3>
<angleaxis> as quat
<EulerAngle> as quat
<matrix3> as quat
```

### 属性

```
<quat>.w : Float
<quat>.x : Float
<quat>.y : Float
<quat>.z : Float
<quat>.angle : Float
<quat>.axis : Point3
```

### 操作符

```
<quat> + <quat>
<quat> - <quat>
<quat> * <quat_or_number>
<quat> / <number>
- <quat>
<quat> * <matrix3>
<quat> == <quat>
<quat> != <quat>
<quat> as <class>      <class>可以为 Matrix3、Angleaxis、EulerAngle
```

### 方法

1. copy <quat>

创建指定 Quat 值的副本，例如：

```
newQuat = copy oldQuat
```

新副本包含的内容与源 Quat 值相同，且与源 Quat 值相互独立。

2. isIdentity <quat>

如果指定 Quat 值为一个单位四元数 (x=y=z=0.0; w=1.0)，返回 True，否则返回 False。

3. normalize <quat>

返回与指定<quat>对应的标准矢量，其各分量的平方和之根为 1。

4. inverse <quat>

返回  $1/<\text{quat}>$  的值。

5. conjugate <quat>

返回指定<quat>对应的共轭数。

6. logN <quat>

返回指定`<quat>`的自然对数。

7. `exp <quat>`

返回指定`<quat>`的指数四元数（其`q.w=0`）。

8. `slerp <start_quat> <end_quat> <number>`

返回指定两 Quat 值的球面线性插值。当参数`<number>`从 0 变化到 1 时，返回值会逐渐由`start_quat`变到`end_quat`。

9. `LnDif <p_quat> <q_quat>`

返回两 Quat 值的对数差，其计算公式为：

`logN((inverse p_quat)*q_quat)`

10. `qCompA <prev_quat> <now_quat> <next_quat>`

计算 Boehm 式插值，其计算公式为：

```
a = now_quat * exp(-(1/4)*(logN((inverse now_quat)*next_quat)+ \
logN((inverse now_quat)*prev_quat)))
```

11. `squad <p_quat> <a_quat> <b_quat> <q_quat> <number>`

进行下面的计算，并返回计算结果：

`slerp(slerp p_quat q_quat number)(slerp a_quat b_quat number) (2*(1-number)*number)`

12. `qorthog <quat> <axis_Point3>`

将指定`<quat>`绕指定轴`<axis_Point3>`旋转 180 度。

13. `transform <quat> <matrix3>`

将指定`<quat>`绕指定矩阵`<matrix3>`进行转换。

14. `squadrev <angle_number> <axis_Point3> <start_quat> \`

`<start_tangent_quat> <end_tangent_quat> <end_quat> <number>`

进行四元数插值。参数`<angle_number>`指定旋转角度，参数`<axis_Point3>`指定旋转轴。

当参数`<number>`从 0 变化到 1 时，返回值会逐渐由`start_quat`变到`end_quat`。

15. `random <quat> <quat>`

返回两 Quat 值之间的随机数。

### 相关方法

1. `quatToEuler <quat> order:<eulertype_integer>`

`eulerToQuat <EulerAngle> order:<eulertype_integer>`

在两个 Quat 和 Euler 值之间进行转换，可选参数`order:<eulertype_integer>`指定了角度的顺序，如果没有指定，使用 XYZ 顺序，如果有指定，其有效值代表的意义如下：

1: XYZ

2: XZY

3: YZX

4: YXZ

5: ZXY

6: ZYX

7: XYX

```

8: YZY
9: ZXZ
2. getEulerQuatAngleRatio <quat1> <quat2> <EulerAngles1> \
                           <EulerAngles2> [angle:<eulertype_integer>]

```

如果用户进行一系列的 Quat 到 EulerAngle 的转换, EulerAngle 值有可能会发生符号反转。发生这种情况的原因是一个 Quat 值可以用不同的 EulerAngle 值进行表示。这种反转可以通过 EulerAngle/Quat 比率来进行检测。EulerAngle/Quat 比率是 EulerAngle 空间里两个 EulerAngle 值之间的角度差与 Quat 空间里两个 Quat 值之间的角度差的比值。

参数说明如下:

- ◆ quat1 和 quat2 为前一个和当前转角值;
- ◆ EulerAngles1 和 EulerAngles2 为前一个和当前被转换的转角值;
- ◆ eulertype\_integer 可选参数, 指定角度的顺序, 如果没有指定, 使用 XYZ 顺序, 如果有指定, 其有效值代表的意义同上。

### 3.2.10 AngleAxis

AngleAxis 类提供了三维空间里方向的一种表示方法, 由一个以度为单位的角度和一个旋转轴组成, 本类与 Quat 类有点类似, 角度也遵循右手规则。

#### 构造函数

```

angleaxis <degree_float> <axis_Point3>
<quat> as angleaxis
<EulerAngle> as angleaxis
<matrix3> as angleaxis

```

#### 操作符

```

<angleaxis> == <angleaxis>
<angleaxis> != <angleaxis>
<angleaxis> as <class>           <class>可以为 Matrix3、Quat、EulerAngle

```

#### 属性

```

<angleaxis>.angle : Float
<angleaxis>.axis : Point3
<angleaxis>.numrevs : Integer

```

#### 方法

```

1. copy <angleaxis>
创建指定 angleaxis 值的副本, 例如:

```

```
newangleAxis = copy oldangleAxis
```

新副本包含的内容与源 angleaxis 值相同, 且与源 angleaxis 值相互独立。

## 2. random &lt;angleaxis&gt; &lt;angleaxis&gt;

返回两个 Angleaxis 值之间的随机数，单位为度。

## 3.2.11 EulerAngles

EulerAngles 提供了三维空间里方向的另一种表示方法，由三个分别绕 X、Y、Z 轴的转角组成，以度为单位，遵从右手规则。

## 构造函数

```
angleaxis <degrees_float> <axis_Point3>
<quat> as angleaxis
<EulerAngle> as angleaxis
<matrix3> as angleaxis
```

## 操作符

```
<angleaxis> == <angleaxis>
<angleaxis> != <angleaxis>
<angleaxis> as <class>           <class>可以为 Matrix3、Quat、EulerAngle
```

## 属性

```
<angleaxis>.angle : Float
<angleaxis>.axis : Point3
<angleaxis>.numrevs : Integer
```

## 方法

## 1. copy &lt;EulerAngles&gt;

创建指定 EulerAngle 值的副本，例如：

```
newEulerAngle = copy oldEulerAngle
```

新副本包含的内容与源 EulerAngle 值相同，且与源 EulerAngle 值相互独立。

## 2. random &lt;EulerAngles&gt; &lt;EulerAngles&gt;

返回两个 EulerAngle 值之间的随机数。

## 3. quatToEuler &lt;quat&gt; order:&lt;eulertype\_integer&gt;

```
eulerToQuat <EulerAngle> order:<eulertype_integer>
```

说明参见本书 3.2.9 节。

## 4. getEulerQuatAngleRatio &lt;quat1&gt; &lt;quat2&gt; \

```
<EulerAngles1> <EulerAngles2> \
```

```
[angle:<eulertype_integer>]
```

说明参见本书 3.2.9 节。

## 5. getEulerMatAngleRatio &lt;matrix3\_1&gt; &lt;matrix3\_2&gt; \

```
<EulerAngles1> <EulerAngles2> [angle:<eulertype_integer>]
```

本函数与 getEulerQuatAngleRatio()类似，惟一的不同是传递的参数为 Matrix3 值，而不是 Quat 值。

### 3.2.12 Matrix3

Matrix3 定义了一个 4×3 转换矩阵。

#### 构造函数

```
matrix3 <row1_Point3> <row2_Point3> <row3_Point3> <row4_Point3>
matrix3 0      零矩阵
matrix3 1      单位矩阵
<quat> as matrix3
<angleaxis> as matrix3
<EulerAngles> as matrix3
rotateXMatrix <number>    所有角度单位为度
rotateYMatrix <number>
rotateZMatrix <number>
transMatrix <Point3>
scaleMatrix <Point3>
rotateYPRMatrix <yaw_number> <pitch_number> <roll_number>
matrixFromNormal <Point3>
```

#### 操作符

```
<matrix3> + <matrix3>
<matrix3> - <matrix3>
<matrix3> * <matrix3>
<matrix3> as <class>           <class>可以是 Quat、Angle、Axis、EulerAngles
```

#### 属性

<matrix3>.row1 : Point3	
<matrix3>.row2 : Point3	
<matrix3>.row3 : Point3	
<matrix3>.row4 : Point3	
<matrix3>.translation : Point3	
<matrix3>.rotationpart : Quat	只读
<matrix3>.translationpart : Point3	只读
<matrix3>.scalerotationpart : Quat	只读
<matrix3>.scalepart : Point3	只读
<matrix3>.determinantsign : Integer	只读

## 方法

### 1. copy <matrix3>

创建指定 Matrix3 值的副本，例如：

```
newMatrix3 = copy oldMatrix3
```

新副本包含的内容与源 Matrix3 值相同，且与源 Matrix3 值相互独立。

### 2. isIdentity <matrix3>

如果指定 Matrix3 矩阵为单位矩阵，返回 True，否则返回 False。

### 3. inverse <matrix3>

返回指定 Matrix3 矩阵的逆矩阵。

### 4. xformmat <transform\_matrix3> <space\_matrix3>

对指定矩阵<transform\_matrix3>进行空间转换。比如，假设用户想在另一个空间对对象施加一个旋转，通常的做法是先将坐标系转换到该空间，然后旋转对象，再把坐标系转换回来。本函数可以简化这些操作，它可以将矩阵<transform\_matrix3>转换到矩阵<space\_matrix3>所在的空间里去。本函数返回值为下面的运算结果：

```
<space_matrix3> * <transform_matrix3> * (inverse <space_matrix3>)
```

### 5. identity <matrix3> mapped

将指定单个或多个矩阵转化为单位矩阵。如果参数<matrix3>为单个矩阵，本函数返回一个单位矩阵；如果参数<matrix3>为一个矩阵数组，本函数返回 OK。在两种情况下，参数<matrix3>都会被单位矩阵替换。

### 6. zero <matrix3> mapped

将指定单个或多个矩阵转化为零矩阵。如果参数<matrix3>为单个矩阵，本函数返回一个零矩阵；如果参数<matrix3>为一个矩阵数组，本函数返回 OK。在两种情况下，参数<matrix3>都会被零矩阵替换。

### 7. orthogonalize <matrix3> mapped

将指定单个或多个矩阵转化为正交化矩阵。正交化矩阵的每一轴之间的夹角都是 90 度。如果参数<matrix3>为单个矩阵，本函数返回一个正交化矩阵；如果参数<matrix3>为一个矩阵数组，本函数返回 OK。在两种情况下，参数<matrix3>都会被正交化矩阵替换。

### 8. translate <matrix3> <Point3> mapped

按右手规则将指定单个或多个矩阵施加一个平移转换。如果参数<matrix3>为单个矩阵，本函数返回转换后的矩阵；如果参数<matrix3>为一个矩阵数组，本函数返回 OK。在两种情况下，参数<matrix3>都会被转换后的矩阵替换。

### 9. rotateX <matrix3> <number> mapped, 所有角度单位为度

rotateY <matrix3> <number>

rotateZ <matrix3> <number>

rotate <matrix3> <quat>

按右手规则将指定单个或多个矩阵施加一个旋转转换。如果参数<matrix3>为单个矩阵，本函数返回转换后的矩阵；如果参数<matrix3>为一个矩阵数组，本函数返回 OK。在

两种情况下，参数<matrix3>都会被转换后的矩阵替换。

10. scale <matrix3> <Point3> [ <boolean> ] mapped

按右手规则将指定单个或多个矩阵施加一个缩放转换。如果参数<boolean>为 True，平移分量被同时缩放，如果参数<boolean>为 False，缩放对平移分量没有影响，默认值为 False。

11. pretranslate <matrix3> <Point3> mapped

按左手规则将指定单个或多个矩阵施加一个平移转换。

12. prerotateX <matrix3> <number> mapped, 所有角度单位为度

prerotateY <matrix3> <number>

prerotateZ <matrix3> <number>

prerotate <matrix3> <quat>

按左手规则将指定单个或多个矩阵施加一个旋转转换。

13. preScale <matrix3> <Point3> [ <boolean> ] mapped

按左手规则将指定单个或多个矩阵施加一个缩放转换。如果参数<boolean>为 True，平移分量被同时缩放，如果参数<boolean>为 False，缩放对平移分量没有影响，默认值为 False。

### 相关方法

```
getEulerMatAngleRatio <matrix3_1> <matrix3_2> <EulerAngles1> \
<EulerAngles2> [angle:<eulertype_integer>]
```

说明参见 3.2.9 节。

## 3.2.13 BigMatrix

BigMatrix 类主要用在  $4 \times 3$  的 matrix3 类不够用的情形。BigMatrixRowArray 是一个包含 N 个元素的矢量，BigMatrix 类值的每一行就是一个 BigMatrixRowArray 类值。BigMatrixRowArray 的元素数据类型只能为 Float。

### 构造函数

BigMatrix <rows> <columns>

### 属性

<BigMatrix>.rows 只读, Integer

<BigMatrix>.columns 只读, Integer

### 操作符

<BigMatrix> + <BigMatrix> 两个<BigMatrix>必须维数相同

<BigMatrix> [<int\_row>] 返回指定行

<BigMatrix> <RowArray> [<int\_column>] 返回指定列 float 值

<BigMatrix> <RowArray> [<int\_column>] = <float\_value> 给指定元素赋值

## 方法

### 1. invert <BigMatrix>

计算指定矩阵的逆矩阵，并将计算结果赋给该参数，返回值也是该逆矩阵。注意，参数<BigMatrix>必须为一个方阵，也即 m=n。

### 2. transpose <BigMatrix>

计算指定矩阵的转置矩阵，也即进行运算 BigMatrix[i][j] = BigMatrix[j][i]。

### 3. clear <BigMatrix>

将指定矩阵的大小设为 0x0。

### 4. setSize <BigMatrix> <rows> <columns>

删除矩阵现有元素，并将它的大小重新设置为指定 rows x columns。

### 5. identity <BigMatrix> mapped

如果矩阵行、列数相等，将指定单个或多个矩阵转化为单位矩阵（对角元素=1，其他元素=0）。如果矩阵行、列数不相等，什么也不执行。如果参数<matrix3>为一个矩阵数组，本函数返回 OK。在两种情况下，参数<matrix3>都会被单位矩阵替换。

### 注意

BigMatrix 类值是一个复合对象，与元素为数组的数组类似。所有的 BigMatrix 方法直接操作指定 BigMatrix 值的元素。正如“引用赋值”一节中所述，如果一个 BigMatrix 值被多个变量引用，可能会导致预料不到的结果。如果用户给 BigMatrix 值的一个元素赋给新值，或使用上述的 BigMatrix 方法，同一对象仍然被多个变量引用。读者可参考下面的例子：

```

bm1=bigmatrix 2 2
for i=1 to 2 do for j=1 to 2 do bm1[i][j]=random 0 10
bm2=bm1
invert bm1
bm2
输出为:
#BigMatrix(          --第 1 行输出结果, 变量 bm1 的内容
[0.00,0.00],
[0.00,0.00]
)
OK                  --第 2 行输出结果
#BigMatrix(          --第 3 行输出结果, 变量 bm2 的内容
[4.00,6.00],
[4.00,9.00]
)
#BigMatrix(          --第 4 行输出结果, 变量 bm1 的内容
[0.75,-0.50],
[-0.33,0.33]
)

```

```
#BigMatrix(          -- 第 5 行输出结果, 变量 bm2 的内容
    [0.75,-0.50],
    [-0.33,0.33]
)
```

### 3.2.14 Box2

Box2 类用整数坐标定义了一个二维矩阵区域，主要用于视窗图形方法。

#### 构造函数

`Box2 <x_integer> <y_integer> <w_integer> <h_integer>`

构造一个 Box2 值，左上角坐标为[x,y]，宽度为 w，高度为 h。

`Box2 <upperleft_point2> <lowerright_point2>`

#### 属性

<code>&lt;Box2&gt;.x : integer</code>	
<code>&lt;Box2&gt;.y : integer</code>	
<code>&lt;Box2&gt;.w : integer</code>	
<code>&lt;Box2&gt;.n : integer</code>	
<code>&lt;Box2&gt;.left : integer</code>	x 的别名
<code>&lt;Box2&gt;.right : integer</code>	
<code>&lt;Box2&gt;.top : integer</code>	y 的别名
<code>&lt;Box2&gt;.bottom : integer</code>	
<code>&lt;Box2&gt;.center : Point2</code>	只读，二维区域的几何中心

#### 操作符

`<Box2> == <Box2>`

`<Box2> != <Box2>`

#### 方法

1. `scale <Box2> <float>`

以`<Box2>`的中心对`<Box2>`进行指定倍数缩放。

2. `translate <Box2> <point2>`

将`<Box2>`平移指定距离。

3. `contains <Box2> <point2>`

如果指定点在`<Box2>`的范围内或在`<Box2>`的边缘上，返回 True，否则返回 False。

4. `rectify <Box2>`

调整`<Box2>`的坐标顺序，使`.top < .bottom` 且 `.left < .right`。

5. `empty <Box2>`

将`<Box2>`设置为一个特殊的“空”值。

### 6. isEmpty <Box2>

如果<Box2>为一个特殊的“空”值，返回 True，否则返回 False。

#### 3.2.15 BitArray

在 MAXScript 里 BitArray 提供了对 3ds max 的 SDK BitArray 类的直接存取。SDK BitArray 在 3ds max 里用来把一系列选择简洁地编译成一个位码字符串。

##### 字面常量

#{<Selection>}

其中<Selection>用逗号隔开，可以是下面几种形式：

<integer> 表示单个选择，<integer>必须大于 0

<integer> .. <integer> 表示一个起止范围

例如：#{1, 5, 10, 200, 302}表示代码 1、5、10、200、302 被“打开”。当用在与选择集有关（如 VertexSelection 或 FaceSelection）的类时，“打开”某一位码表示该序号的元素或子对象被选择。

##### 属性

<bitArray>.count : Integer 只读

在调用本属性时 BitArray 可容纳的最大的索引号。

##### 操作符

<BitArray>.[<integer>] 如指定位当前被选择，返回 True，否则返回 False

<BitArray>.[<integer>]=<boolean> 设置或清除索引位

<BitArray>+<BitArray> 对两个 BitArray 数组执行逻辑或（OR）操作

<BitArray>-<BitArray> 对两个 BitArray 数组求差

-<BitArray> 求逆值

例如：

```
$foo.verts[#{20..1023}] --选择序号从 20 到 1023 的顶点
for i in #{2..200} do print i --打印从 2 到 200 之间的数字
a=#{1..5}
b=#{3..10}
c=-(a-b) --返回值为：#{3..5}
```

##### 方法

###### 1. append <bitarray> <integer>

将指定序号添加到 BitArray 数组，并增加数组的长度。

###### 2. join <bitarray> <bitarray>

将两个 BitArray 数组联合（执行 OR 操作）。

###### 3. findItem <bitarray> <integer>

再指定 BitArray 数组里查找指定序号，如果有找到，返回该序号，否则返回 0。

4. `deleteItem <bitarray> <integer>`  
将指定序号从 BitArray 数组里清除。

5. `copy <bitarray>`  
为 BitArray 数组创建一个独立的副本。

### 3.2.16 Time

Time 类在 MAXScript 里用来计算动画时间。在 3ds max 中，最小的时间单位为 tick，其代表的时间长度为 1/4800 秒。

#### 字面常量

<code>[-]{&lt;decimal_number&gt;[m   s   f   t]}+</code>	分、秒、帧、tick
<code>[-]{&lt;digit&gt;}:{&lt;digit&gt;}[,{&lt;digit&gt;}]</code>	SMPTE 时间：分、秒、帧
<code>[-]&lt;decimal_number&gt;</code>	标准化时间

下面是几种有效的 Time 类数据：

<code>2.5s</code>	--2.5 秒
<code>1m15s</code>	--1 分 15 秒
<code>2m30s5f2t</code>	--2 分 30 秒 5 帧 2 tick
<code>125f</code>	--125 帧
<code>18.25f</code>	--18.25 帧
<code>1f20t</code>	--1 帧 20 tick
<code>2:10.0</code>	--2 分 10 秒 0 帧(SMPTE)
<code>0:0.29</code>	--29 帧(SMPTE)
<code>0.45n</code>	

#### 构造函数

`normTime <float>`

#### 属性

`<time>.ticks`  
`<time>.frame`  
`<time>.normalized`

#### 操作符

`<time> + <time>`  
`<time> - <time>`  
`-<time>`  
`<time> * <time>`  
`<time> / <time>`  
`<time> * <number>`  
`<time> / <number>`  
`<time> == <time>`

```
<time> != <time>
<time> > <time>
<time> >= <time>
<time> < <time>
<time> <= <time>
<time> as Number
```

### 方法

- random <time> <time>

返回两个时间之间的一个随机数。

- abs <time>

返回时间的绝对值。

## 3.2.17 Interval

Interval 类值代表一个时间段，它有两个 Time 类的成员：起点 (.start) 和终点 (.end)。成员还可以是 Number 类值，此时 Number 值被解释为帧数。

### 构造函数

```
Interval <start.time> <end.time>
```

### 属性

```
<Interval>.start : Time
```

```
<Interval>.end : Time
```

例如：

```
t = animationRange.start + 5
orig_anim_range=animationRange           --保存初始动画时间范围
animationRange=interval 0 100000          --设置实际动画时间范围
--创建一个 Script 类 Controller，其动画时间范围为当前动画时间范围
ControlObj.scale.controller=scale_script()
--将一个脚本赋给 Controller
ControlObj.scale.controller.script = controlscript
--将动画时间范围重置为初始动画时间范围
animationRange=orig_anim_range
```

## 3.2.18 Bitmap

Bitmap 类提供了对 3ds max 位图的存储和低级存取，位图可以是仅驻留在内存的临时对象，也可以是磁盘上的一个文件。那些存储在文件里的位图有一个.fileName 属性，且只能为只读或只写，而不能同时进行读写。用户可以用 openBitmap() 和 SelectBitmap() 函数打开一个位图文件，可以用函数 Bitmap() 或 render()、copy()、get- ChannelAsMask() 创建一个临时的或只写的位图。

## 构造函数

```
1. bitmap <width <height> [ fileName:<fileName_string> ] \
   [ numframes:<integer> ] [ color:<color> ] \
   [ gamma:<float> ] [ pixelAspect:<float> ]
```

功能：创建一个空位图，返回值为一个可存储的位图。

函数各参数说明如下：

- ◆ **Filename** 指定使用 save 方法存储位图时使用的文件名，如果文件名已存在，系统并不会把原位图文件内容装入。
- ◆ **Numframes** 允许用户创建一个多帧位图。
- ◆ **Color** 允许用户为位图所有像素设置初始化的颜色。
- ◆ **Gamma** 允许用户为新建位图设置 gamma 值。
- ◆ **pixelAspect** 允许用户为新建位图设置 pixelAspect 值。

**注意** 一旦位图被创建，其参数 gamma 和 pixelAspect 就不能被改变了。

2. openBitmap <filename\_string>

打开指定位图文件，并返回包含指定位图文件内容的 Bitmap 值，如果指定文件不存在，会产生一个运行错误。打开的位图文件是只读的。

3. selectBitmap [caption :<open\_dialog\_title\_string>]

显示 Image Input Selection 浏览窗口让用户选择一个位图文件，并打开该位图文件，如果用户按 Cancel 退出浏览窗口，返回值为 undefined。打开的位图文件是只读的。

4. copy <bitmap>

复制已有的位图，创建一个独立的位图，如果源位图为只读的，返回的 Bitmap 值为一个临时 Bitmap 值，文件名为 null，且仅包含一帧；如果源文件为多帧文件，仅复制当前帧，作为目标位图的 0 帧。

## 属性

1. <bitmap>.fileName : string

存取位图的文件名字符串。

2. <bitmap>.palette : Array

返回一个 256 色的数组。在由 getIndexa Pixels() 函数用颜色索引存取该数组时应小心，因为本数组定义 0 为第一个成员，而 MAXScript 数组序号从 1 开始，如果位图没有调色板，返回 undefined。目前在 MAXScript 里还不能创建一个带调色板的位图，但可以对带调色板的位图进行读写操作。

3. <bitmap>.frame : Integer

返回一个多帧文件当前的帧数，仅在只读的位图里才能设置该属性。对那些本来就是多帧文件（如.avi 和.mov 类文件）而言，帧数从 0 开始。

4. <bitmap>.numframes : Integer

多帧文件的总帧数，单帧位图文件返回 1。

5. <bitmap>.height : Integer 只读

以像素为单位的位图高度。

6. <bitmap>.width : Integer 只读

以像素为单位的位图宽度。

7. <bitmap>.gamma : Float 只读

位图的 gamma 值。

8. <bitmap>.aspect : Float 只读

返回位图的 pixelAspect 值，注意是像素的 aspect 值，而不是图像的 aspect 值。

## 方法

1. display <bitmap>

在 VFB（虚拟帧缓存）里显示指定图像。此后对图像的修改不会自动对 VFB 显示进行刷新。如果要进行刷新，必须重新调用本函数。但是改变图像的.frame 属性会导致 VFB 刷新。注意每一图像都有自己的 VFB，也即如果显示两个不同图像，会有两个 VFB 被显示。

2. unDisplay <bitmap>

如果指定图像正被显示，关闭其对应的 VFB。

3. save <bitmap> [frame:<integer>]

保存位图文件。在调用本函数之前，必须确定有设置.fileName 属性。如果保存一个多帧图像文件，关键词参数 frame: 的作用将根据输出文件类型的不同而不同：对一些本来就是多帧文件（如.avi 和.mov 类文件）的图像文件，本函数会忽略参数 frame:，而总是将指定图像保存到下一帧里，用户不能随机将内容写到或覆盖某一帧；而如果对图像序列（如.bmp 和.jpg、.tga）必须指定参数 frame:，系统会将图像内容插入到指定帧位置。在用户调用函数 close() 之前，输出文件都会保持打开的状态。

Save() 函数仅能调用那些可写的图像文件。如果用户尝试对一个用函数 openBitmap() 或 selectBitmap() 打开的图像文件进行存盘操作，会导致一个运行错误。

4. close <bitmap>

关闭与指定位图对应的位图文件（如果它正被打开）。如果有一个 VFB 与打开的位图对应，VFB 也会被关闭。本函数同时会释放由位图占用的内存，包括内部像素帧缓存。如果用户正在生成大量位图，比如在一个渲染循环语句下，可以调用函数 close() 来释放这些大量的内存，否则这些内存只有等到下一次内存收集时才会被释放。

5. copy <source\_bitmap> <dest\_bitmap>

将源位图 <source\_bitmap> 复制给目标位图 <dest\_bitmap>。如果两位图的大小不一样，系统会将图像缩放到目标位图的大小。

6. gotoFrame <bitmap> <frame>

在多帧文件（avi、flc 等文件）里定位到指定帧 <frame>。仅在只读的位图里才能设置该属性。对那些本来就是多帧文件（如.avi 和.mov 类文件）而言，帧数从 0 开始。如果指定位图有 VFB 被打开，VFB 会同时刷新。

7. getPixels <bitmap> <coord\_point2> <num\_pixels>

返回一个元素数据类型为 Color 的一维数组，表示位图从指定位置开始指定长度的像

素序列。参数<coord\_point2>指定开始像素的位置，位图左上角的坐标值为[0,0]，参数的.x 分量为列坐标，.y 分量为行坐标。获取的像素必须处在同一行，而不能一次获取跨行的像素。参数<coord\_point2>和<num\_pixels>如果指定的像素超出位图范围，返回一个空数组。

8. `setPixels <bitmap> <coord_point2> <color_value_array>`

将指定范围（从指定位置<coord\_point2>开始，长度为数组<color\_value\_array>长度）的像素设置成<color\_value\_array>指定的颜色。参数<coord\_point2>的.x 分量为列坐标，.y 分量为行坐标。和函数 `getPixels()`一样，用户不能一次给跨行的像素设置颜色。

9. `getIndexedPixels <bitmap> <coord_point2> <num_pixels>`

返回一个元素数据类型为 `Integer` 的一维数组，表示位图指定位置开始指定长度的像素颜色对应的调色板序号。参数<coord\_point2>指定开始像素的位置，位图左上角的坐标值为 [0,0]，参数的.x 分量为列坐标，.y 分量为行坐标。注意：调色板序号从 0 开始。用户不能一次获取跨行的像素调色板序号。

10. `setIndexedPixels <bitmap> <coord_point2> <number_array>`

将指定范围（从指定位置<coord\_point2>开始，长度为数组<number\_array>长度）的像素设置成<number\_array>指定的调色板序号。用户不能一次给跨行的像素调色板序号赋值。

11. `getChannel <bitmap> <coord_point2> <chan_name>`

返回指定位置像素、指定通道的 G 缓冲通道。参数<chan\_name>为通道标识符，有效值有：

```
#zDepth
#matID
#objectID
#UVCoords
#normal
#unClamped
#coverage
#node
#shaderColor
#shaderTransparency
#velocity
#weight
```

如果位图包含指定通道，返回值为一个数组，每一个数组元素对应指定像素每一层的通道值，顺序为从前到后。如果位图没有包含指定通道，返回值为 `undefined`。一般来说，.rla 类文件和用 MAXScript 函数 `Render()`、带参数 `channels`：渲染出来的位图都包含通道。

12. `getChannelAsMask <bitmap> <chan_name> [to:<bitmap>] \`  
`[fileName: "filename_string"] [pixelAspect:<float>] \`  
`[gamma:<float>] [layer:<integer>] [invert:<boolean>] \`  
`[node:<node> | objectID:<integer> | matID:<integer>] \`  
`[velocity:#angle | #magnitude] | [angle:<float>]`

建立并返回一个独立的 8 位灰度级位图，可用来作为材质、贴图、效果等的蒙版。该位图大致与显示在 VFB 里的图像对应。

参数说明如下：

- ◆ <bitmap> 包含 G 缓冲通道数据的源位图。
- ◆ <chan\_name> 指定要生成蒙版的通道，有效值见函数 getChannel() 的说明。
- ◆ to : <bitmap> 指定要将蒙版写入现有的位图，其宽度和高度必须与源位图匹配。
- ◆ fileName : "filename\_string" 指定与要生成位图相连的文件名。
- ◆ gamma : <float> 指定要生成位图的 gamma 值。
- ◆ pixelAspect : <float> 指定要生成位图的 pixelAspect 值。
- ◆ layer : <integer> 指定从源位图的哪一通道层中摘取数据，默认值为 1。
- ◆ invert : <boolean> 如果为 True，反转生成位图，默认值为 False。
- ◆ node : <node> | objectID : <integer> | matID : <integer> 指定一个子蒙版。如果指定了上述参数中的一个，生成的蒙版会被裁减，只有在指定 Node、objectID 或 materialID 可见的地方才会包含数据。如果要使用子蒙版，用户必须确定#node、#objectID 或#matID 通道在源位图中存在。
- ◆ velocity : #angle | #magnitude | angle : <float> 这些参数只有在选定通道为#velocity 时才可以使用。其默认值为#magnitude。这些参数对生成位图的影响说明如下：
  - velocity:#angle 生成位图的像素值与像素运动方向的角度成正比，角度 0（水平向右）对应亮度值 0，180° 对应亮度值 127，360° 对应亮度值 255。
  - velocity:#magnitude 生成位图的像素值与像素运动速度大小成正比，速度越大，像素颜色越深。
  - angle:<float> 生成位图的像素值与像素运动方向的角度与指定角度之间的偏差成正比，如果像素运动方向的角度刚好为指定角度，则亮度值为 255，如果两者之间的夹角为 10°，则亮度值为 0。

### 相关方法

#### 1. setSilentMode <boolean>

开关 Silent 模式。如果设为 Off，在位图输入、输出过程中发生的任何错误都会导致一个错误信息对话框。如果设为 On，在位图输入、输出过程中发生的任何错误都不会显示。返回一个布尔值，表示调用本函数前的 Silent 模式。

Silent 模式是 3ds max 内部的一个状态，其他 3ds max 插件也可以开关该模式。建议用户如果使用本函数开关 Silent 模式，将返回值保存，在执行完位图输入、输出操作后，将它的状态恢复到原来的值。

#### 2. silentMode()

返回一个布尔值，表示当前的 Silent 模式。

#### 3. selectSaveBitmap [ caption:<open\_dialog\_title\_string> ]

显示一个 3ds max 位图文件存储对话框，并返回一个字符串，表示完整的文件路径名。

如果用户按 Cancel 退出对话框，返回值为 undefined。

## 4. freeSceneBitmaps()

释放所有由位图缓存占用的内存。如果内存被许多位图分割成许多碎片，可以调用本函数先释放内存，再将要用的位图重载。

## 5. getBitmapOpenFileName caption:&lt;title&gt; filename:&lt;seed\_filename\_string&gt;

```
getBitmapSaveFileName caption:<title> filename:<seed_filename_string>
```

显示 3ds max 位图文件选择对话框。两个函数都返回完整的文件路径名或 undefined。

如果用 getBitmapSaveFileName() 函数时，选择了一个已有的文件名，系统会显示一个 Exists | Confirm Overwrite 对话框。

**注意** getBitmapOpenFileName() 函数并不需要选择一个已有的文件，用户也可以直接在对话框里键入一个文件名。如有必要，用户应该检测返回文件是否已经存在。

当用户打开一个位图文件时，位图的所有帧都会载入内存。如果用户完成对位图的处理后，最好对位图调用函数 close() 或给保存位图值的变量赋给值 undefined，然后执行一次内存收集 (gc())，这样内存会被释放。如果用户重复使用函数 render() 或 getChannelAsMask()，为了节约内存，可以使用参数 to:<bitmap>，将渲染结果或蒙版位图输出到同一个现有的位图。

下面的脚本演示了如何正确创建、保存一个多帧图像文件：

```
theTeapot=teapot() --创建对象
animate On at time 10 \
    rotate theTeapot 180 z_axis --设置 animate 和 time 关联语句
cam=targetcamera pos:[200,0,100] \
    target:theTeapot --旋转 teapot 对象
    --设置 camera 对象
renderFrames=#{1,3,5..12} --指定要渲染的帧
b=bitmap 160 120 filename:"c:\\t.avi" --创建新位图
for i = 1 to renderFrames.count do --循环
( if renderFrames[i] then --如果为要渲染的帧
( at time i --设置 time 关联语句
    render 160 120 camera:cam to:b --渲染到位图帧
    save b --存储每一帧
) --如果仅在循环结束后存盘,
) --只有最后一帧被存盘
)
close b --关闭输出文件，释放内存
```

输出结果为：

```
$Teapot:Teapot01 @ [0,0,0] --第 1 行结果
OK --第 2 和 3 行结果
$Target_Camera:Camera01 @ [200,0,100] --第 4 和 5 行结果
#{1, 3, 5..12} --第 6 行结果
Bitmap:c:\\t.avi --第 7 行结果
OK --第 8 到 15 行循环结果
OK --第 16 行结果
```

下面的脚本读入一个位图文件，输出一个脚本到 Listener 窗口，这个脚本定义了一个

函数，其功能为重建刚刚读入的位图。用户可以把这个函数复制到自己的脚本。这个函数可用来在 MAXScript 里创建按钮图像，这样编程者在发行自己编写的脚本时，就不要将图像捆绑发行了。

```

(
b=selectbitmap()
--打开 image file browser 窗口
bname = "bitmap_" + (getfilenamefile b.filename)
--用变量 filename 建立文件名
w=b.width
--获取位图属性
h=b.height
format "-----\nfn load_% =(\n" bname
--开始定义函数
format "local %=bitmap % %\n" bname w h
--在函数里创建位图
format "fn unpack val = for p in val collect(r=p/256^2; g=p/256-r*256;
b=mod p 256; color r g b)\n"
--定义一个函数，将一个整数展开成一个像素颜色值
for r=0 to h-1 do
--对位图里每一行循环
( format "setpixels % [0,%](unpack #(" bname r
--调用 unpack 函数 unpack 将像素列写入位图
pixels=getpixels b [0,r] w           --读入像素列
        for c=1 to w do             --对每一像素循环
( p=pixels[c]                      --获取像素
    --将像素压缩成一个整数并输出该整数
    format "%((p.r as integer)*256+(p.g as integer))*256+(p.b
as integer))
        if c != w then           --如果数据没完
            format ", "
        else
            format ")\n"          --否则结束本行
)
)
format "return %\n" bname           --函数返回值为位图
format ")\n-----\n"                --结束函数定义
)

```

输出：

```

-----
fn load_bitmap_convert =
local bitmap_convert=bitmap 4 4
fn unpack val = for p in val collect(r=p/256^2; g=p/256-r*256;
b=mod p 256; color r g b)
setpixels bitmap_convert [0,0](unpack #(12632256,      12632256,
12632256,12632256))
setpixels bitmap_convert [0,1](unpack #(255, 255, 255, 255))
setpixels bitmap_convert [0,2](unpack #(255, 255, 255, 255))

```

```

setpixels bitmap_convert [0,3](unpack #(255, 12632256, 12632256,
    12632256))
return bitmap_convert
)

```

### 3.2.19 Stream

Stream 值用来格式化和保存字符类数据。用户可以用 Stream 值进行字符读写。Stream 值有三种类型：FileStream、StringStream、WindowStream。下面分别进行讨论：

#### FileStream（文件数据流）

FileStream 类值用来在 MAXScript 里对文本文件进行 I/O 操作，一个 FileStream 值在 MAXScript 里代表一个打开的文本文件。

##### 构造函数

###### 1. CreateFile <filename\_string>

创建一个新文件并返回一个 FileStream 值，如果不能创建指定文件，返回 undefined。

###### 2. OpenFile <filename\_string> [mode:<mode\_string>]

打开一个已存的文件并返回一个 FileStream 值，如果指定文件不存在，返回 undefined。

可选的 mode: 参数有下面两种模式。

- ◆ "a" 在文件末尾加入内容

- ◆ "r" 打开文件为只读

###### 3. openEncryptedFile <filename> <key>

用指定的整数密码打开一个加密的文件，并返回一个 FileStream 值，用户只能对它进行读操作。

##### 方法

###### 1. readLine <filestream>

读入下一行，并作为字符串返回。

###### 2. readChar <filestream>

读入下一个字符，并作为字符串返回。

###### 3. readChars <filestream> <char\_count>

读入指定数量字符，并将它们作为一个字符串返回。

###### 4. readDelimitedString <filestream> <string>

在指定文件里查找指定分界符（一个字符串），一旦找到分界符（或到达文件结尾），则返回指定分界符号。

###### 5. skipToString <filestream> <string>

在文件里向后扫描，查找指定字符串，如果找到，则将文件指针定位在该字符串后面。否则，函数返回 undefined。

###### 6. skipToNextLine <filestream>

将文件指针定位在下一行的起始位置。

7. filePos <filestream>

返回文件指针的当前偏移位置。

8. seek <filestream> <integer>

将文件指针定位到指定偏移位置，以便后面的 I/O 操作从该位置开始。

9. eof <filestream>

如果文件指针处在文件末尾，返回 True，否则返回 False。

10. flush <filestream>

将缓存块里的数据写入文件，以确保输出到文件的数据都写入磁盘。

11. close <filestream>

将缓存块里的数据写入文件，并关闭文件。

12. readValue <filestream>

读入文件里下一个 MAXScript 操作数，并对它求值。

13. readExpr <filestream>

读入文件里下一个 MAXScript 表达式，并对它求值。

14. execute <filestream>

读入文件里剩下的所有表达式，并对它们求值。

### 相关方法

print <value> to : <filestream>

format <fmt\_string> { <value> } to : <filestream>

上面方法的说明，参见 3.1 节。

## StringStream (字符串数据流)

StringStream 类允许用户用所有文本文件 I/O 函数来构造、分解字符串。因为很容易把 String 类值和 FileStream 值互相转化，我们可以用 FileStream 值来操作那些很长的复杂 String 值。

### 构造函数

StringStream <initial\_string>

<String> as FileStream

### 操作符

<FileStream> as string

### 方法

1. copy <FileStream>

为<FileStream>创建一个独立的副本。

下面的这些方法和 FileStream 类值相应的方法功能一致，详细说明请读者参见 FileStream 类说明。

```

readValue <stringstream>
readExpr <stringstream>
readLine <stringstream>
readChar <stringstream>
readChars <stringstream> <number>
readDelimitedString <stringstream> <string>
skipToString <stringstream> <string>
skipToNextLine <stringstream>
execute <stringstream>
filePos <stringstream>
seek <stringstream> <pos>
eof <stringstream>
close <stringstream>
flush <stringstream>

```

函数 close() 和 flush() 是为与文件 I/O 操作保持一致而保留的函数，对 `StringStream` 类值调用这两个函数，系统不会执行任何操作。

上面这些方法之间的区别可以从下面的例子中看出。在下面例子中，创建了一个 `StringStream` 类值，包含两个表达式。

```

s=stringstream "random 0. 1.;random red blue"
readvalue s      --读入第 1 行并求值
readvalue s      --读入第 2 行并求值
seek s 0         --定位到 stringStream 值开始
readexpr s       --读入第 1 个表达式并求值
readexpr s       --读入第 2 个表达式并求值
seek s 0         --定位到 stringStream 值开始
execute s        --对所有表达式求值

```

输出：

```

StringStream: "random 0. 1.;random red blue"  --第 1 行结果
random()          --第 2 行结果
0.0              --第 3 行结果
OK               --第 4 行结果
0.448042         --第 5 行结果
(color 86.476 0 163.24)  --第 6 行结果
OK               --第 7 行结果
(color 30.2417 0 143.636) --第 7 行返回值

```

### WindowStream (窗口数据流)

`WindowStream` 类允许用户向一个 `Script Editor` 窗口输出字符。

#### 构造函数

`newScript()`

打开一个新的 Script Editor 窗口并返回一个 WindowStream 值。

#### 相关方法

```
print <value> to :<windowstream>
format <fmt_string> { <value> } to :<windowstream>
```

例如：

```
debug = newScript()
...
print $foo to:debug
...
format "name is %\n" obj.name to:debug
```

## 3.3 特殊数据类型

MAXScript 里的特殊数据类型有：undefined、OK、unsupplied、DontCollect，下面分别进行解释。

### 3.3.1 undefined 类

undefined 类仅有一个实例，就是保留系统变量 undefined，所有未初始化的变量和数组成员的值都为 undefined。

#### 构造函数

undefined

例子：

```
--检查灯光对象 MyLight 是否有 target 对象
if MyLight.target != undefined then
--如果有 target 对象，获取 light 对象与 target 对象之间的距离
dist = distance MyLight.pos MyLight.target.pos
else
--否则，获取 light 对象与坐标原点[0,0,0]之间的距离
dist=length MyLight.pos
--将 CH_Hand1 设为值 undefined，用户可以用来判别它是否被删除
deleteChangeHandler CH_Hand1; CH_Hand1=undefined
```

### 3.3.2 OK

OK 类仅有一个实例，就是保留系统变量 OK，用于一些没有确定意义返回值的函数和表达式的返回值。

### 3.3.3 unsupplied

unsupplied 类仅有一个实例：就是保留系统变量 unsupplied。在函数调用时，如果未给那些关键词参数（Keyword Parameter）指定调用值，系统会自动将这些参数的初始值赋给

值 `unsupplied`。下面是一个例子：

```
fn walk_tree obj parent :=
( if parent == unsupplied then
  ...
else
...
)
```

### 3.3.4 DontCollect

`DontCollect` 类用于 For 循环中标明哪些值不被收集到数组中，`DontCollect` 类仅有一个同名实例：`DontCollect`。

例如：

```
--收集所有半径小于 5 的 Sphere 类对象
little_spheres =
for i in $* collect
  if classOf i == sphere and i.radius < 5 then i else DontCollect
```

## 3.4 高级数据类型

在 MAXScript 里，除了前面所述的基本数据类型之外，还提供了几种高级数据类型，其中一些高级数据类型如 `Structure`（结构）、`Array`（数组）在其他高级语言中也有，但更多的是专门针对 3ds max 的特点而设计的，如 `MaxKey`、`PathName`、`ObjectSet` 等。下面小节将分别进行详细说明。

### 3.4.1 Structure（结构）

在 MAXScript 里可以定义自己的“新”数据类型：一种特殊的复合数据类型，`Structure`（结构）。结构定义可以让用户定义值的布局，并在随后的脚本里使用它们。

构造函数

```
struct <struct_name>(<member> {,<member>})
```

其中每一个成员`<member>`可以有下面几种形式：

`<name> [=<expr>]`: 结构变量和初始值

`<function_def>`: 函数定义

例如：

```
struct person(name,height,age,sex)
bill=person name: "Bill" height:72 age:34 sex:#male
```

第一行定义了一个名为 `person` 的“新”类，第二行创建了一个 `person` 类的实例，并存在变量 `bill` 里，各成员分别赋给它一个默认值 `undefined`。

存取结构成员的值的方法和存取标准属性一样，例如：

```
bill.age    --34
joe.age     --undefined
joe.age = bill.age-4
```

结构定义在处理一些固定的，通常小数量的互相关联的数据成员时比数组更好用。我们可以像属性一样引用成员值，而不用像数组一样使用索引值。

当用函数 classOf() 来查询结构的类时，返回结构定义中的<struct\_name>，例如：

```
classof bill      --返回 person
```

struct 值构造函数和函数调用一样，接受位置变元和关键字变元的初始化，新 struct 值的成员必须按先为位置变元后为关键字变元的顺序排列。例如对下面的结构定义：

```
struct foo(name,age,height,hair="brown")
```

可以用下面几种方法来创建一个 foo 结构的实例：

```
f1=foo "bill" 23 72          --name、age、height
f2=foo age : 45 name : "sam"  --随机排列
f3=foo "mary" hair = "red"   --name、hair
```

### 方法

```
copy <struct>
```

例如：

```
struct test(pos1, pos2)           --定义一个结构
testval = test [1,2,3] [4,5,6]    --创建一个结构实例
testval_copy=copy testval        --创建结构副本
testval_copy.pos1.x=10           --改变备份结构成员值
testval                         --查看源结构的值
```

输出：

```
#Struct:test(                  --第 1 行结果
    pos1:<data>,
    pos2:<data>)
#test(pos1:[1,2,3], pos2:[4,5,6]) --第 2 行结果
#test(pos1:[1,2,3], pos2:[4,5,6]) --第 3 行结果
10                           --第 4 行结果
#test(pos1:[10,2,3], pos2:[4,5,6]) --第 6 行结果
```

结构是从 Value 类发展而来的，它继承了 Value 类数值的所有方法：print()、format()、classOf()、SuperClassOf()、==、!= 等，请读者参见 3.1 节。

### 在结构中定义局部函数

如构造函数中所示，用户可以在结构中定义局部函数，这样可以方便地为结构创建一些与之相关的函数，并避免与全局函数（可能在几个不同的脚本文件中定义的）发生命名冲突，例如：

```
struct foo
(a, b, c,
```

```

fn baz n = sqrt(n+1)
fn bar x y = print(x^z+y^z)
)

```

上例中结构 foo 中包含了三个成员和两个局部函数，用户可以像属性一样直接通过结构定义值本身或结构的实例来存取这些局部函数，例如：

```

p = foo.baz q          --作为结构定义值的属性存取
foo.bar x y
f = foo 2 3 4
f.bar x y              --作为结构实例的属性存取

```

在结构定义的局部函数里可以引用结构里的数据成员，例如下例所示：

```

struct RndVals
( RndVals=#(), ptr, seedVal,
  fn generateRV num fromVal toVal =
( if seedVal != undefined do seed seedVal
    RndVals=for i=1 to num collect random fromVal toVal
    ptr=1
    RndVals
),
  fn deleteRv =
( RndVals=#()
  ptr=undefined
  undefined
),
  fn getNextRV =
( if Nvals == 0 do return undefined
  val=RndVals[ptr]
  ptr += 1
  if ptr > RndVals.count do ptr=1
  val
),
  fn sortRV =(sort RndVals),
  fn setSeed val =(SeedVal = val),
  fn setPointer val =
( if val > 0 and val <= RndVals.count then
    (ptr=val;True)
  else
    False
)
)
MyRandomVals = RndVals()
MyRandomVals.setSeed 12345
MyRandomVals.generateRV 10 0 100
MyRandomVals.getNextRV()
MyRandomVals.sortRV()
MyRandomVals.setPointer 1
MyRandomVals.getNextRV()
MyRandomVals.deleteRv()

```

输出：

```
#Struct:RandVals(                                     --行 1 到 29 结构定义结果
    generateRV:<fn>,
    sortRV:<fn>,
    RndVals:<data>,
    deleteRv:<fn>,
    getNextRV:<fn>,
    seedVal:<data>,
    setSeed:<fn>,
    setPointer:<fn>,
    ptr:<data>)
--第 30 行结果
#RandVals(RndVals:#(), ptr:undefined, seedVal:undefined)          --第 31 行结果
12345                                         --第 32 行结果
#(23, 59, 79, 68, 17, 72, 84, 16, 89, 72)  --第 33 行结果
23                                         --第 34 行结果
#(16, 17, 23, 59, 68, 72, 72, 79, 84, 89) --第 35 行结果
True                                         --第 36 行结果
16                                         --第 37 行结果
undefined
```

上面脚本中，结构 RandVals 有三个数据成员和六个局部函数成员。

### 3.4.2 MaxKey 类

在 MAXScript 里，一个 MaxKey 值代表某个控制器（Controller）的一个关键帧。

#### 构造函数

##### 1. addNewKey <controller> <time> [#select] [#interpolate]

功能：在指定时间给指定控制器添加一个新的关键帧（key）。如果可选参数#select 被同时指定，新添加的关键帧在 Track View 视窗里被选中。

如果可选参数#interpolate 被指定，新关键帧的值为在指定时间处控制器的插入值，否则新关键帧的值等于前一个关键帧的值。

##### 2. getKey <controller> <index\_integer>

功能：返回一个 MaxKey 值，表示指定序号的关键帧。

##### 3. <key\_array> [<index\_integer>]

功能：存取关键帧数组<key\_array>里指定序号的关键帧。

其中<key\_array>为<controller>.keys 或<node>.<animatable\_property>.keys。

#### 属性

##### 1. <key>.time : Time

返回关键帧所处的时间。

##### 2. <key>.selected : Boolean

指定关键帧是否被选择。

其中.time 属性对某些控制器为只读的，而有些为可存取的，都会在第 12 章的关于各控制器的描述中指出。对那些.time 属性为可存取的控制器，还有下面的属性。

### 3. <key>.value

指定关键帧的当前值，数据类型可变，取决于控制器类型。

#### 方法

`copy <key>`

为指定<key>创建一份副本。但副本并不是独立于源<key>，因为一个<key>值总是指向控制器里的一个关键帧。

Beizer、TCB 和 Barycentric Morph 类控制器还有它们自己的属性和方法，具体参见第 12 章的内容。

#### 在工作中使用 MaxKey 值

改变一个关键帧的.time 属性可能会打乱控制器原来的时间顺序，所以在全部完成了此类操作后，用户必须调用 SortKeys() 函数来对关键帧重新排序，这样动画才不会出错。一个关键帧的.value 属性所属的类取决于它所在的控制器，例如：Linear\_float.value 的类为 Float，而 TCB\_rotation.value 的类为 Quat。所有关键帧的属性都支持数学赋值操作和嵌套属性存取，例如：

```
$foo.pos.keys[2].time += 20f      --改变单帧的.time 属性
$foo.pos.keys.time += 20f        --改变所有帧的.time 属性
$Box1.uvw_map.center.keys[2].value.x += 20
for k in $baz.bend.angle.keys do k.value *= 1.1
```

在某些情况下，存储在关键帧的.value 属性里的值与在命令面板或 Track View 视窗里显示的值并不一样，它们之间存在一个比例关系。例如：在命令面板或 Track View 视窗里百分比一般显示在 0~100 的范围；转角用度作为单位显示；而颜色值的范围为 0~255。而实际存储在.value 属性里时，百分比数值范围为 0~1；转角用弧度作单位；而颜色值的范围为 [0~1, 0~1, 0~1]。取下面的几个 PArray 类对象属性为例：

属性名	数据类型	默认值	备注
<PArray>.X_Spin_Vector	Float	1.0	可动画
<PArray>.Spin_Time_Variation	Float	0.0	可动画, percentage
<PArray>.Spin_Phase	Float	0.0	可动画, angle
<PArray>.viewPercent	Float	10.0	percentage

上面例子中，属性.X\_Spin\_Vector 对与其对应控制器的.value 属性不作用缩放因子；属性.Spin\_Time\_Variation 对与其对应控制器的.value 属性会施加一个缩放因子（用百分比显示，用小数存储）；属性.Spin\_Phase 同样对其他对应控制器的.value 属性会施加一个缩放因子（用度显示，用弧度存储）；属性.viewPercent 因为不能设置动画，用百分比存储。

一般情况下，我们在用 MAXScript 编程时可不考虑这种缩放。但有两种情况编程者必须加以考虑：一是在脚本控制器里；二是当一个控制器被用作两个或两个以上属性的控制

器时，例如下面的例子：

```
obj = PArray()
cont = bezier_float()
obj.X_Spin_Vector.controller=cont
obj.Spin_Time_Variation.controller = cont
obj.Spin_Phase.controller = cont
obj.X_Spin_Vector = 1
obj.X_Spin_Vector      --无缩放
obj.Spin_Time_Variation   --百分数
obj.Spin_Phase      --角度
```

输出：

```
$PArray:PArray03 @ [0,0,0] --第 1 行结果
Controller:Bezier_Float      --第 2 行结果
Controller:Bezier_Float      --第 3 行结果
Controller:Bezier_Float      --第 4 行结果
Controller:Bezier_Float      --第 5 行结果
1                           --第 6 行结果
1.0                         --第 7 行结果，无缩放
100.0                       --第 8 行结果，百分比缩放，1 = 100%
57.2958                     --第 9 行结果，角度缩放，1 弧度 = 57.2958 度
```

### 3.4.3 NoteTrack

3ds max 系统里 NoteTrack 类数据用来在脚本插件里给 MAXWrapper 对象（如场景对象、Material、Controller、Modifier）添加一些用户定制的信息，这些 NoteTrack 数据在 Track View 视窗里可以看到。所有 MAXWrapper 对象都可以有一个或多个 NoteTrack 数据，这些信息会固定地和对象一起存储在场景文件里。

NoteTrack 的结构和动画控制器相似，它也有一个名为.keys 的属性，该属性返回一个数据类型为 MaxNoteKeyArray 的值。

#### 构造函数

`notetrack <name_string>`

用指定名称创建一个新的空 NoteTrack 数据。该名称仅在 MAXScript 里用来区分不同的 NoteTrack，在 3ds max 界面里是不可见的。

#### 属性

<code>&lt;NoteTrack&gt;.keys</code>	MAXNoteKeyArray
<code>&lt;NoteTrack&gt;.name</code>	String

#### 方法

1. `getNoteKeyTime <NoteTrack> <index>`  
返回指定序号 NoteKey 的时间。
2. `getNoteKeyIndex < NoteTrack > <time>`

返回指定时间 NoteKey 的序号。

#### 相关方法

1. addNoteTrack <MAXWrapper\_object> <NoteTrack>

为指定对象添加一个新 NoteTrack。

2. deleteNoteTrack <MAXWrapper\_object> <NoteTrack>

将指定 NoteTrack 从指定对象删除。

3. hasNoteTracks <MAXWrapper\_object>

如果指定对象包含 NoteTrack，返回 True，否则返回 False。

4. numNoteTracks <MAXWrapper\_object>

返回一个整数，表示指定对象包含 NoteTrack 的数目。

5. getNoteTrack <MAXWrapper\_object> <index\_integer>

返回指定对象里指定序号的 NoteTrack，序号从 1 开始。

#### 存取 NoteTrack 值

用户可以把一个或多个 NoteTrack 赋给一个 MAXWrapper 对象，这些 NoteTrack 可以在 Track View 视窗里作为 MAXWrapper 对象的子对象被看到。NoteTrack 为插件提供了一个添加数据到 3ds max 场景对象的方法，并且和对象一起以 NoteKey 形式存储在场景文件里。一个 NoteTrack 可包含任意数量的 NoteKeys。

### 3.4.4 Collection (集合类数据)

在 MAXScript 里用到的许多数值都是数据集合，如数组、使用匹配符的路径名和一些预定义的选择集。和大多数操作系统的壳命令语言一样，MAXScript 会自动对集合里的每一元素执行适当的操作：为数据集合里的每一元素调用一次函数，这种机制在计算机科学里称为“映射”（Mapping）。例如：

```
hide $lights/key*          --隐藏场景里所有以 key 开头的 light 类对象
$Box*.position = [0,0,0]    --为所有 Box 对象设置中心
delete dead_objects        --删除数组 dead_objects 里所有对象
--将对象集 obj 里所有对象的 Position 控制器的.intangent 类型设为#fast
obj.pos.keys.intangent = #fast
--将所有 light 对象的值减小 20%
lights.value *= 0.8
```

MAXScript 通过查看函数的第一个变元参数或属性赋值的对象来判断一个序列集合是否要执行映射操作。在函数调用时，MAXScript 还会查看函数类型是否合适，并对其他变量表达式仅求一次值，然后用序列集合的每一元素和求值后的其他变量调用函数。在属性赋值中，MAXScript 对“=”号右边的表达式仅求一次值，然后将它赋给序列集合的每一元素。

在运行中如发现序列集合中的某一元素为非法的函数变元或参数，MAXScript 会报告一个错误并中止程序运行。

除了几个通用属性外，在 MAXScript 里如果是对属性赋值，映射操作仅对序列集合的

当前级别对象进行。例如下面语句，仅对 objects 选择集执行映射操作。

```
objects.pos = [0,0,0] --将所有对象移至 World 坐标原点
```

但如果属性赋值语句指定的最后一级为下面的属性，映射操作将在前一级的属性上进行：.angle、.b、.blue、.axis、.controller、.g、.green、.isAnimated、.keys、.r、.red、.track、.x、.x\_rotation、.y、.y\_rotation、.z、.z\_rotation。例如：

```
objects.pos.z = 0.0 --将所有对象移至 World 坐标系的 XY 平面
```

在每一种集合类型的描述里，读者可以发现那些可被执行映射操作的类型被标记为 mappable，在函数和操作符的描述里，可以对集合进行映射操作的都加上了标签 mapped。一般而言，可执行映射操作的函数是那些会给它们的变量带来一些副作用的函数，如移动、删除和隐藏对象。另外一些测试函数如 isHidden() 不执行映射操作，因为这样做毫无意义。所有 MAXScript 定义的可执行映射操作的函数如 copy()、delete()、move() 等最后都有一个可选的变量#nomap，可以强制指定调用函数时不进行映射操作。

用户可以定义自己的映射函数，只要在函数定义前加上系统保留字 mapped，例如：

```
mapped function rand_color x =
  (x.wireColor = random black white)
```

**注意** 可选参数#nomap 变量在自定义的映射函数里无效。

MAXScript 里数据集合的类型很多，下面专门用一节的篇幅来讨论 MAXScript 里常见的集合类数据类型。

## 3.5 集合类数据类型

### 3.5.1 Array（数组）

在 MAXScript 里一个数组是一组排序的值集合，其长度可变。数组里每一个元素可以是任意类型的数值，可以进行索引操作，用户可以直接在程序中构造数组，或用 MAXScript 提供的数组函数来建立数据集合。

数组可进行映射操作。

#### 字面常量

```
#(<value>,<value>...)
#(<expr>{,<expr>})
#() 空数组
```

下面几个都是数组的例子：

```
#(1,2,3)
#()
#(1, "foo", #(1.2,-4,#fred),[1,2,3])
```

## 构造函数

`<collection> as Array`

将别的类型数据集合转化为数组。

## 属性

`<array>.count : Integer` 只读, Array 里的成员数目

## 操作符

`<array> [<integer>]` 返回指定位置成员, 序号从 1 开始

`<array> [<integer>]=<value>` 给指定成员赋值

`<array> + <collection>` 创建一个新 array, 包含两个操作数里的所有元素

## 方法

1. `append <array> <value>`

将指定值插入到数组的最后。

2. `copy <array> [#noMap] mapped`

为数组里的所有元素创建一份副本, 返回值 OK。如果有指定参数#noMap, 我们称之为“浅层副本”：仅创建数组值本身的副本, 而没有为存储在数组里的元素创建副本。

3. `deleteItem <array> <number>`

从数组里删除指定序号的元素。

4. `join <array> <collection>`

将变量<collection>里的所有元素添加到数组<array>的后面。

5. `sort <array>`

按升序对数组进行排序。所有数组的元素必须具有可比性。

6. `findItem <array> <value>`

在数组里查找指定目标对象, 返回第一次找到的元素序号, 如果没有找到, 返回 0。

Point3 类值和 String 类值如果内容相同, 系统也认为它们是匹配的。

7. `insertItem <value> <array> <integer>`

在数组的指定位置<integer>插入指定值<value>。

8. `qsort <array> <function> [start : <integer>] [end : <integer>] \`

[user-defined key args passed to function]

用指定函数（该函数对数组元素进行逐一比较）对数组排序。指定的函数必须带两个参数, 其返回值为: 如果第一个参数小于第二个参数, 返回一个<0 的 Number 数; 如果第一个参数等于第二个参数, 返回一个 0; 如果第一个参数大于第二个参数, 返回一个>0 的 Number 数。如果没有用参数 start: 和 end: 来指定排序的起止位置, 系统对整个数组进行排序。

下面的脚本产生 10 个随机坐标位置, 然后根据它们到坐标原点[0,0,0]的距离进行排序:

```
positions = for i=1 to 10 collect(random [0,0,0] [100,100,0])
qsort positions(fn distFrom0 v1 v2 =(length v1)-(length v2))
```

## 9. amin(&lt;array&gt; | {value})

返回数组或指定值序列里的最小值。如果数组长度为 0 或没有指定值序列，返回值 undefined。例如：

```
myMin1 = amin #(5,1,4,2,8)
myMin2 = amin 5 1 4 2 8
```

## 10. amax(&lt;array&gt; | {value})

返回数组或指定值序列里的最大值。如果数组长度为 0 或没有指定值序列，返回值 undefined。

用户可以将一个对象集和通配符路径名用 as 操作符转化为一个数组。这相当于给对象集或与路径名匹配的当前对象拍了一张“快照”，这样可以在随后的脚本里操作集合里的对象，而不用担心对象集改变。这种功能与 3ds max 用户界面下的 Named Selection Sets 按钮类似。如果用户删除了数组里的某一对象，而在随后的脚本里再对数组进行映射操作，系统会给出一个错误信息。

例子：

```
sel1 = Selection as array
Boxes_at_load = $Box* as array
snap_children = $torso...* as array
original_cameras = cameras as array
```

## 3.5.2 PathName (路径名)

## 字面常量

如果用户想用名称来标识一个场景对象，必须用 PathName (路径名) 类型的数据。

PathName 可以通过指定对象的层级结构来指定场景中某一对象，也可以在 PathName 中使用匹配符来指定一系列对象。完整的 PathName 语法定义为：

```
<Path_name>::=$<Path> | $
<Path>::=[<objectset>] [/][<levels>]<level-name>
<levels>::=<level>/<level>
<level>::=<level-name>
<level_name>::={<alphanumeric>|-|*|?|\}
‘{<alphanumeric>|-|*|?|\}’
```

其中 objectset 将查找目标设在指定的对象选择集范围，有关 objectset 内容详见 3.5.3 节。

所有 3ds max 对象都按层次结构排列，这种排列可以在 Track View 窗口中看到。这一点与操作系统里文件的路径类似，如\$dummy/head/neck 指定了一个名字为 neck 的场景对象，它的父对象为 head，而它的顶级父对象为 dummy。这样组织的 PathName 有助于区分 3ds max 里特定的对象，即使几个对象有同样的名字。如果对象名为 head 的对象在场景中是惟一的，MAXScript 允许用简写\$head 来引用它而不考虑它的层级位置。即使对象名不惟一，用户也可以用简写方式，但会得到哪一个对象将不可预料。

PathName 同时提供了一次引用几个对象的强大功能，比如某一父对象的所有子对象，或是对象中名称中包含指定字符串的全部对象，此时可以使用匹配符。正如我们在操作系统中一次指定一批文件时一样，如 \$dummy/\* 指定了当前 dummy 对象的所有子对象，\$foot\* 指定了层级结构任何名字以 foot 开头的对象，“\*”代表任意数目的字符。也可以用匹配字符“？”来代表一个任意的字符，如 \$Boxo? 表示那些以名字 boxo 开始，且在 o 之后仅有一个字符的对象。

匹配符也可用来在层级结构中指定父对象级，如 \$dummy/\*\* 指定了对象 \$dummy 的所有子对象的子对象。如果用户想在对象的任意嵌套层次的子对象中查找一个对象，可以用“...”，例如：

```
$dummy/.../Box*
```

指定了 \$dummy 任意层次的子对象中名字以 Box 开头的所有子对象。当使用“...”时，MAXScript 允许省略“/”，上面路径名也可以写成：

```
$dummy ... Box*
```

如果用户在对象的路径名里用了匹配符，MAXScript 将同时运算所有匹配的对象，如命令：

```
max hide $*foot*
```

会隐藏所有名字中包含字符 foot 的对象。

用户也可以自动执行映射操作，一次设置所有匹配对象的属性，例如：

```
$*Box*.position + =[10,10,0]
```

将把场景中所有名字中包含字符 Box 的对象的 position 属性加上 [10, 10, 0]。

用户还可以用一种简写形式来指定当前选定的对象（一个或多个）：“\$”。如果当前仅一个对象被选中，字符“\$”代表了被选中的对象，如果不止一个对象被选中，字符“\$”代表了被选择的对象集，而如果当前没有对象被选择，字符“\$”代表值 undefined。“\$”在 Listener 窗口下是一种有用的缩写形式，例如：

```
$.pos=[0,0,0]
max hide $
rotate $ 35 z.axis
```

但在脚本文件中，并不鼓励使用“\$”这种缩写形式，因为容易导致错误出现。

#### PathName 中的空格和其他特殊字符

3ds max 允许用户在对象名中使用任何字符，包括空格和其他 MAXScript 标点，可以把 PathName 中的这些字符放在一对单引号 ‘’’ 内，例如：

```
$' a silly name !!'
```

#### 注意

引号并不能屏蔽匹配符，所以如果你的场景对象名字中碰巧含有一个“\*”字符，应该在“\*”前面加一个转码符号“\”，例如：

```
$ 'what the \ * '
```

因为大量的 3ds max 名称中包含空格，用下划线“\_”可以代替名称中的空格，例如下面两行语句是一样的：

```
body part=$' Bip 01 L Upper Arm '
body part=$ Bip_01_ L_Upper_Arm
```

### 属性

1. <PathName>.center                  Point3, 只读  
返回<PathName>里所有对象绑定框的中心。
2. <PathName>.max                  Point3, 只读  
返回绑定框的右上角。
3. <PathName>.min                  Point3, 只读  
返回绑定框的左下角。
4. <PathName>.count                  Integer, 只读  
返回<PathName>里对象数目，本属性仅对使用匹配符的 PathName 值有效。

### 操作符

1. <PathName>[<integer>]  
存取指定位置的对象，排序从 1 开始。
2. <PathName> as array  
把 PathName 转化成 array。

下面是 PathName 值的几个实例：

```
$Box01                                  --名称为 Box01 的对象
$torso/left-up-arm/left-low-arm --层级路径名
$*Box*                                  --所有名称中包含 Box 的对象
$torso/*                                  --所有 torso 的直接子对象
$helpers/d*                                  --所有名字以 d 开头的 helper 类对象
```

例子：

```
a = $*feet*[i]                          --获取对象集 feet 里的第 i 个对象
--检测那些.scale 属性不是单位矢量的对象
for obj in $dummy...
where obj.scale != [1,1,1]
do format "% has non-identity scale : %\n" obj.name obj.scale
--将对象 foo 的.pivot 属性设为对象集 Box 的右上角
$foo.pivot = $Box*.max
in $dummy
(
sphere name: "ear1" pos:[10,10,10] --创建 dummy 的子对象
sphere name: "ear2" pos:[-10,10,10]
scale $foo/* [1,1,2]                  --查找 dummy 子对象里名称为 foo 的对象
)
```

### 3.5.3 ObjectSet (对象集)

ObjectSet 代表 3ds max 里场景对象的各个类型 (category)，下面的构造函数都是系统保留变量。

#### 构造函数

Objects	所有场景对象
Geometry	指定类型的 3ds max 对象
Lights	
Cameras	
Helpers	
Shapes	
Systems	
spacewarps	
Selection	当前选择对象集合

#### 属性

1. <ObjectSet>.center Point3, 只读  
返回<ObjectSet>里所有对象绑定框的中心。
2. <ObjectSet>.max Point3, 只读  
返回绑定框的右上角。
3. <ObjectSet>.min Point3, 只读  
返回绑定框的左下角。
4. <ObjectSet>.count Integer, 只读  
返回<ObjectSet>里对象数目。

#### 操作符

<ObjectSet>[<integer>] 存取集合里指定序号的元素，序号从 1 开始  
<ObjectSet> as array 将对象集合转化为一个数组

#### 相关方法

1. clearSelection()  
清除当前场景对象选择集。
2. deselect <PathName>  
将指定对象从当前选择集里去掉，例如：  
`deselect $Box* --将所有名字以 Box 开头的对象从当前选择集里去掉`
3. select <PathName>  
先解除所有当前选择集，然后选择指定的对象。
4. selectMore <PathName>

将指定对象集添加到当前选择集。

#### 5. getCurrentSelection()

返回一个数组，表示当前选择集。本函数相当于函数 Selection as array，但如果场景里有大量的对象，本函数的执行速度会大大快于后者。

需要特别注意的是：Lights 和 Cameras 两个 ObjectSet 值还包含了它们的目标对象（如果有的话），如果想改变所有光线或摄影机的属性，必须要逐个改变 ObjectSet 里的每一对对象来确定它的类型是为 Light 或 Camera，例如：

```
for obj in lights do
  if iskindof obj light do
    obj.multiplier *=1.3
```

### 3.5.4 SelectionSet (选择集)

一个 SelectionSet 类值代表 3ds max 工具栏里 Name Selection Set 下拉列表里的一个命名选择集。

#### 构造函数

1. SelectionSets[<set\_name>]  
<set\_name>可为 string 或 name 类。
2. SelectionSets[<index>]  
<index>为在下拉列表里的序号。

#### 操作符

<SelectionSet>[<integer>] 获取选择集指定序号的对象

#### 属性

1. <SelectionSet>.center Point3, 只读  
返回选择集里所有对象绑定框的中心。
2. <SelectionSet>.max Point3, 只读  
返回绑定框的右上角。
3. <SelectionSet>.min Point3, 只读  
返回绑定框的左下角。
4. <SelectionSet>.count Integer, 只读  
返回<Selectionset>里的对象数目。

### 3.5.5 SelectionSetArray (选择集数组)

SelectionSetArray 类仅有一个实例：SelectionSets，它包含 3ds max 用户界面的工具栏里 Named Selection Set 下拉列表里全部的命名选择集。

## 构造函数

SelectionSets

### 属性

<SelectionSetArray>.count Integer, 只读

返回 SelectionSets 包含的对象个数。

### 操作符

1. <SelectionSetArray> [<set\_name>]

用子集名存取集合成员，参数<set\_name>可为 String 或 Name 类值。

2. <SelectionSetArray> [set\_index\_integer]

用序号存取集合成员，序号从 1 开始。

3. <SelectionSetArray> [<set\_name>]=<node\_array>

创建或替换名为 set\_name 的子集，参数<set\_name>可为 String 或 Name 类值，node\_array 必须为一个节点数组。

### 方法

deleteItem <SelectionSetArray> <set\_name>

从指定选择集<SelectionSetArray>里删除指定子集<set\_name>，参数<set\_name>的类型可为 String 或 Name。

### 相关方法

1. getNumNamedSelSets <SelectionSetArray>

返回一个整数，表示系统里当前命名选择集的个数。

2. getNamedSelSetName <set\_index\_integer>

返回一个字符串，表示指定序号命名选择集的名称，序号从 1 开始。

3. getNamedSelSetItemCount <set\_index\_integer>

返回一个整数，表示指定序号命名选择集里的对象个数，序号从 1 开始。

4. getNamedSelSetItem <set\_index\_integer> <node\_index\_integer>

返回一个对象，表示指定序号命名选择集里的指定序号的对象。

例如：

```
set1 = SelectionSets ["my set 1"]
move set1 [10,0,0]
SelectionSets[#set2].wireColor = red
for i in 1 to SelectionSets.count do
    saveNodes SelectionSets[i] ("set" + i as string + ".max")
```

用户还可以对 SelectionSets 数组调用标准数组方法来修改、添加、删除新的命名选择集，例如：

```
SelectionSets["new set"] = Selection --创建新的命名选择集
```

```
SelectionSets["old spheres"] = $sphere* --创建新的命名选择集
SelectionSets[#foo] = #(obj1, obj2, obj3) --创建新的命名选择集
deleteItem SelectionSets "old set" --删除命名选择集
```

### 3.5.6 NodeChildrenArray (子对象数组)

**NodeChildrenArray** 类值代表一场景节点的直接子对象，组成一个虚拟的数组。这样可以对子对象进行循环操作，执行映射函数和存取如.count 等属性。

#### 构造函数

<node>.children

#### 属性

1. <nodechildrenarray>.center Point3, 只读  
返回所有子对象绑定框的中心。
2. <nodechildrenarray>.max Point3, 只读  
返回绑定框的右上角。
3. <nodechildrenarray>.min Point3, 只读  
返回绑定框的左下角。
4. <nodechildrenarray>.count Integer, 只读  
返回选择集里的子对象数目。

#### 操作符

<nodechildrenarray>[<integer>]

#### 方法

1. append <nodechildrenarray> <node>  
将指定对象设为<nodechildrenarray>选择集父对象的子对象。
2. deleteItem <nodechildrenarray> <node>  
将指定对象从<nodechildrenarray>选择集里移走。

### 3.5.7 VertexSelection (顶点选择集)

一个 **VertexSelection** 类值代表一个 **Mesh** 类对象的顶点集合，作为一个虚拟数组，**VertexSelection** 数组是动态的，它的元素会随着 **Mesh** 类对象的顶点或选中顶点的改变而改变。

#### 构造函数

- |                      |                   |
|----------------------|-------------------|
| <Mesh>.selectedVerts | Mesh 对象当前被选中的顶点数组 |
| <Mesh>.verts         | Mesh 对象的全部顶点，只读   |

## 属性

1. <VertexSelection>.count Integer, 只读

返回指定 VertexSelection 数组里的顶点数目。

2. <VertexSelection>.selSetName 数组名, 只读

返回与 VertexSelection 相关的 Mesh 对象的顶点级选择集数组名。

下面的属性只能用在单个顶点 (即形式为\$foo.verts[n]):

3. <VertexSelection>.index Integer, 只读

返回指定顶点在 Mesh 对象中的索引号, 例如:

```
$foo.selectedVerts[2].index -- 返回当前选择集中第 2 个顶点的序号
```

4. <VertexSelection>.pos Point3

返回或设置 Mesh 对象中指定顶点的位置, 例如:

```
$foo.verts[i].pos = $baz.verts[j].pos + [10,0,0]
sv = for i in $foo.selectedVerts collect i.index
```

## 操作符

1. <Mesh>.selectedVerts=(<array> | <bitarray>)

选择指定顶点。

2. <VertexSelection>[<integer>]

将指定序号的顶点作为一个单独的 VertexSelection 值 (也即长度为 1) 返回。序号从 1 开始。

3. <VertexSelection>[<integer>]=<Point3>

设置指定序号顶点的位置。

4. <VertexSelection>[(<integer\_array> | <bitarray>)]

返回一个由数组所指定序号的顶点组成的 VertexSelection 值。序号从 1 开始。

5. <VertexSelection>[(<#name> | <string>)]

返回顶点级命名选择集, 命名选择集的名字可用 Name 或 String 类来指定。

6. <VertexSelection>[(<#name> | <string>)] = (<vertexSelection> | \ <integer\_array> | <bitarray>)

将顶点级命名选择集设置成指定顶点集。

## 方法

1. move <VertexSelection> <Point3>

将<vertexSelection>选择集里的所有顶点移动指定距离。

2. select <VertexSelection>

选中<VertexSelection>选择集里的所有顶点。

3. deselect <VertexSelection>

解除对<VertexSelection>选择集里的所有顶点的选择。

4. delete <VertexSelection>

删除<VertexSelection>选择集里的所有顶点。

## 5. append &lt;VertexSelection&gt;(&lt;VertexSelection&gt; | &lt;integer&gt;)

将单个或多个顶点添加到<VertexSelection>选择集里。

## 6. findItem &lt;VertexSelection&gt;(&lt;VertexSelection[&lt;integer&gt;] | &lt;integer&gt;)

在<VertexSelection>选择集里查找指定序号顶点。

例如：

```
move $foo.verts[#mouth] [0,0,10]
--移动命名选择集 mouth 里的顶点
select $baz.verts["front verts"]
--选中命名选择集 front verts 里的顶点
$foo.verts[#baz] = #(1,3,4,5,10)
--给指定顶点设置命名选择集名称 baz
$baz.verts[#cursel] = $baz.selectedVerts
--给当前选择顶点设置命名选择集名称 cursel
$foo.verts.selSetName
--获取对象 foo 的所有顶点级命名选择集的名称
for n in $.verts.selSetName do print $.verts[n]
--打印出所有顶点级命名选择集
```

## 3.5.8 FaceSelection (面选择集)

一个 FaceSelection 类值代表一个 Mesh 对象的面 (Face) 集合，是一个虚拟数组。FaceSelection 数组是动态的，其内容随着 Mesh 对象的面或选择面的改变而改变。

## 构造函数

<Mesh>.selectedFaces	Mesh 对象当前被选中的面
<Mesh>.Faces	Mesh 对象的全部面，只读

## 属性

## 1. &lt;FaceSelection&gt;.count Integer, 只读

在 FaceSelection 数组里的面数目。

## 2. &lt;FaceSelection&gt;.selSetName Name 类数组，只读

下面的属性只能用在单个面 (即形式为：\$foo.faces[n])：

## 3. &lt;FaceSelection&gt;.index Integer, 只读

返回指定面在 Mesh 对象中的索引号，例如：

```
$foo.selectedFaces[2].index --返回当前选择集中第 2 个面的索引号
```

## 操作符

## 1. &lt;Mesh&gt;.selectedFaces=(&lt;array&gt; | &lt;bitarray&gt;)

选择指定面。

## 2. &lt;FaceSelection&gt;[&lt;integer&gt;]

将指定序号的面作为一个单独的 FaceSelection 值 (也即长度为 1) 返回。序号从 1 开始。

3. <FaceSelection>[<integer>]=<Point3>

设置指定序号面的位置。

4. <FaceSelection>[(<integer\_array> | <bitarray>) ]

返回一个由数组所指定序号的面组成的 FaceSelection 值。序号从 1 开始。

5. <FaceSelection>[(<#name> | <string>) ]

返回面级 (Face-Level) 命名选择集，命名选择集的名字可用 Name 或 String 类来指定。

6. <FaceSelection>[(<#name> | <string>)]=(<FaceSelection> | \<integer\_array> | <bitarray>)

将面级命名选择集设置成指定面集。

### 方法

1. move <FaceSelection> <Point3>

将<FaceSelection>选择集里的所有面移动指定距离。

2. select <FaceSelection>

选中<FaceSelection>选择集里的所有面。

3. deselect <FaceSelection>

解除对<FaceSelection>选择集里的所有面的选择。

4. delete <FaceSelection>

删除<FaceSelection>选择集里的所有面。

5. append <FaceSelection>(<FaceSelection> | <integer>)

将单个或多个面添加到<FaceSelection>选择集里。

6. findItem <FaceSelection>(<FaceSelection[<integer>] | <integer>)

在<FaceSelection>选择集里查找指定序号面。

例如：

```
move $foo.Faces[#mouth] [0,0,10]
--移动命名选择集 mouth 里的面
select $baz.Faces["front Faces"]
--选中命名选择集 front Faces 里的面
$foo.Faces[#baz] = #(1,3,4,5,10)
--给指定面设置命名选择集名称 baz
$baz.Faces[#cursel] = $baz.selectedFaces
--给当前选择面设置命名选择集名称 cursel
$foo.Faces.selSetNames
--获取对象 foo 的所有面级命名选择集的名称
for n in $.Faces.selSetNames do print $.Faces[n]
--打印出所有面级命名选择集
```

### 3.5.9 EdgeSelection (边选择集)

一个 EdgeSelection 类值代表一个 Mesh 对象的边 (Edge) 集合，作为一个虚拟数组，EdgeSelection 数组是动态的，其元素随着 Mesh 类对象的边或选中边的改变而改变。

### 构造函数

`<Mesh>.selectedEdges` Mesh 对象当前被选中的边数组  
`<Mesh>.Edges` Mesh 对象的全部边，只读

### 属性

1. `<EdgeSelection>.count` Integer, 只读

返回指定 EdgeSelection 数组里的边数目。

2. `<EdgeSelection>.selSetName` 数组名, 只读

返回与 EdgeSelection 相关的 Mesh 对象的边级 (edge-level) 选择集数组名。

下面的属性只能用在单个边 (即形式为: \$foo.edges[n]):

3. `<EdgeSelection>.index` Integer, 只读

返回指定边在 Mesh 对象中的索引号, 例如:

```
$foo.selectedEdges[2].index
```

返回当前选择集中第二条边的索引号。

### 操作符

1. `<Mesh>.selectedEdges=(<array> | <bitarray>)`

选择指定顶点。

2. `<EdgeSelection>[<integer>]`

返回指定序号边作为一个单独的 EdgeSelection, 序号从 1 开始。

3. `<EdgeSelection>[(<integer_array> | <bitarray>)]`

返回数组元素所指定序号的边, 序号从 1 开始。

4. `<EdgeSelection>[(<#name> | <string>)]`

返回指定边级 (edge-level) 命名选择集, 命名选择集的名字可用 Name 或 String 类数据。

5. `<EdgeSelection>[(<#name> | <string>)] = \`

`(<EdgeSelection> | <integer_array> | <bitarray>)`

将边级 (Edge-Level) 选择集设置成指定顶点集。

### 方法

1. `move <EdgeSelection> <Point3>`

将<EdgeSelection>选择集里的所有边移动指定距离。

2. `select <EdgeSelection>`

选中<EdgeSelection>选择集里的所有边。

3. `deselect <EdgeSelection>`

解除对<EdgeSelection>选择集里的所有边的选择。

4. `delete <EdgeSelection>`

删除<EdgeSelection>选择集里的所有边。

5. append <EdgeSelection>(<EdgeSelection> | <integer>)  
将单个或多个边添加到<EdgeSelection>选择集里。
6. findItem <EdgeSelection>(<EdgeSelection[<integer>] | <integer>)  
在<EdgeSelection>选择集里查找指定序号边。  
例如：

```
move $foo.Edges[#mouth] [0,0,10]
--移动命名选择集 mouth 里的边
select $baz.Edges["front Edges"]
--选中命名选择集 front Edges 里的边
$foo.Edges[#baz] = #(1,3,4,5,10)
--给指定边设置命名选择集名称 baz
$baz.Edges[#cursel] = $baz.selectedEdges
--给当前选择边设置命名选择集名称 cursel
$foo.Edges.selSetNames
--获取对象 foo 的所有边级命名选择集的名称
for n in $.Edges.selSetNames do print $.Edges[n]
--打印出所有边级命名选择集
```

### 3.5.10 MaxKeyArray (关键帧数组)

MaxKeyArray 类值代表 3ds max 的 Track View 视窗下 Controller track (控制器轨迹) 里的所有关键帧。

#### 构造函数

1. <node>.<animatable.property>.keys

一个包含所有关键帧的数组，例如：

```
$foo.position.keys
$foo.twist.angle.keys
```

如果 3ds max 对象的参数还没有设置动画（也即尚未有控制器被赋给该参数），.keys 属性返回 undefined。

2. <controller>.keys

例如：

```
$foo.position.controller.keys
$foo[3][1].keys
```

#### 属性

<key\_array>.count              Integer, 只读

返回<key\_array>里的关键帧数目。

#### 操作符

<key\_array>[<index.integer>]

返回一个 MaxKey 值，表示指定序号的关键帧，序号从 1 开始。

## 方法

1. append <key\_array> <key>

向<key\_array>的末尾添加指定关键帧<key>。

2. deleteItem <key\_array> <index\_integer>

删除指定序号的关键帧，序号从 1 开始。

3. addNewKey <key\_array> <time> [ #select ]

在指定时间向<key\_array>添加一个新关键帧，新关键帧的值为指定时间处的插入值。

如果有指定参数#select，新关键帧在插入后被选中。如果在指定时间已有关键帧存在，addnewKey()函数不会再增加一个关键帧，在这种情况下，函数的返回值为在该时间已有的关键帧。

4. deleteKeys <key\_array> [ #allKeys ] [ #Selection ]

根据指定参数从控制器里删除关键帧：

#allKeys (默认值): 删除控制器里所有关键帧；

#Selection: 删除控制器里当前被选中的关键帧。

5. deleteKey <key\_array> <index\_integer>

删除指定序号关键帧，序号从 1 开始。

6. moveKeys <key\_array> <time> [ #Selection ]

将所有关键帧移动指定时间。如果有指定参数#Selection，仅移动当前有选中的关键帧。

7. sortKeys <key\_array>

对关键帧根据它们的时间重新进行排序。因为有些 MaxKey 操作可能打乱关键帧的排序，所以必须调用本函数来更正关键帧的排序。

### 注意

与 PathName 和数组类值一样，MaxKeyArray 值也支持映射赋值，例如：

\$Box01.pos.keys.time += 10f --将所有关键帧的时间向前移动 10 帧

\$cyl23.bend.angle.keys.value \*= 1.2

--将属性.bend.angle (弯曲角度) 的所有关键帧的值放大 1.2 倍

## 3.5.11 MaxNoteKeyArray (Note 轨迹关键帧数组)

MaxNoteKeyArray 代表 3ds max 的 Note Track 视窗里的所有 NoteKey。

### 构造函数

<NoteTrack>.keys

例如：

```
nt=getNoteTrack $foo.position.controller 1
ntk=nt.keys
```

### 属性

<NoteKeyArray>.count                  Integer, 只读

返回集合里的对象个数。

### 操作符

`<NoteKeyArray>[<index_integer>]`

存取指定序号集合成员，序号从 1 开始。

### 3.5.12 ModifierArray (修改器数组)

ModifierArray 类代表一个场景对象的修改器 (Modifier) 数组，包含了该场景对象的修改器堆栈 (Modifier Stack) 下的所有修改器。

#### 构造函数

`<node>.modifiers`

#### 属性

`<modifierarray>.count` Integer, 只读

返回 ModifierArray 数组里修改器个数。

#### 操作符

1. `<modifierarray>[<integer>]`

返回指定序号的修改器，索引号从 1 开始，并从修改器堆栈的顶部开始计数。

2. `modifierArray[<name> | <string>]`

返回由 Name 类或 String 类值指定名字的修改器。

### 3.5.13 MaterialLibrary (材质库)

一个 MaterialLibrary 类值是包含一个材质表的数组。

3ds max 系统变量 currentMaterial\_Library 和 sceneMaterial 是 MaterialLibrary 类的两个实例。系统变量 meditMaterials 严格说不是一个 MaterialLibrary 类值，而是在包含 Material Editor 窗口样品槽 (Sample Slot) 材质的一个虚拟数组，在大多数情况下，meditMaterial 可以被看作一个 MaterialLibrary 值，但 meditMaterial 数组的元素不仅可以被赋予一种材质，还可以被赋予一种贴图 (textureMap)。

#### 构造函数

`currentMaterialLibrary` 系统变量，当前材质库里的材质

`SceneMaterials` 系统变量，场景里的材质

`meditMaterials` 在 Material Editor 窗口里的材质

`MaterialLibrary {<material>}`

创建一个临时材质库，用户可以用函数 `append()` 向它加入材质，但目前尚不能存取或作为当前材质库。

#### 属性

`<mat_lib>.count` Integer, 只读

返回材质库里的材质数目。

### 操作符

#### 1. <mat\_lib> [<integer> | <name> | <string>]

从材质库返回指定的一种材质，材质库可以像数组一样用整数进行索引，也可以用材质名（用 Name 或 String 类）来进行索引。

#### 2. <mat\_lib>[<integer> | <name> | <string>]=(<material> | <textureMap>)

将指定材质或贴图赋给材质库的指定元素，材质库的索引方法可以用整数或由 Name 或 String 类值指定的材质名，如果用整数进行索引，而索引值大于材质库的长度，会产生一个运行错误。如果材质库用材质名进行索引，而指定的材质名不在材质库里，指定的材质或 textureMap 被加入到材质库的最后。

#### 3. meditMaterials [<slot\_index\_integer>]

返回 Material Editor slots 里指定序号的材质或贴图，有效的<slot\_index\_integer>值为 1~24。

#### 4. meditMaterials [<slot\_index\_integer>]=(<material> | <textureMap>)

将指定材质或贴图赋给 Material Editor 窗口里指定序号的样品槽（Slot），有效的<slot\_index\_integer>值为 1~24。例如：

```
meditMaterials[3]=standard()
```

### 方法

#### 1. append <mat\_lib> <material>

将指定材质添加到指定材质库。本方法不适用于 meditMaterials 型材质库。

#### 2. deleteItem <mat\_lib>(<integer> | <name> | <string>)

从指定材质库删除指定材质，指定材质可以使用索引（Integer）或直接指定材质名（Name 或 String）。本方法不适用于 meditMaterials 型材质库。

#### 3. findItem <mat\_lib> <material>

如果在指定材质库找到指定材质，返回该材质的序号，否则返回 0。

### 相关方法

#### 1. getMeditMaterial <slot\_index\_integer>

返回 Material Editor 窗口里指定 Slot 的顶级材质或贴图。参数<slot\_index\_integer>为整数，取值范围为 1~24。例如：

```
foo=getMeditMaterial 3
```

#### 2. setMeditMaterial <slot\_index>(<material> | <textureMap>)

将指定 Material Editor Slot 里的材质或贴图替换为指定的材质或贴图。

#### 3. loadDefaultMatLib()

装载 3ds max 默认的材质库。

#### 4. loadMaterialLibrary <filename\_string>

装载 3ds max 指定的材质库文件，并将它设为当前材质库。如果材质文件没有指定路径，函数会在当前配置的 bitmap 路径里查找。如果装载成功，返回 True，否则返回 False。

#### 5. saveMaterialLibrary <filename\_string>

将当前材质库存储到指定文件里。如果存盘成功，返回 True，否则返回 False。

#### 6. fileOpenMatLib()

显示 File Open 对话框，让用户选择一个材质库来装载。注意：如果当调用本方法时，Material/Map Browser 窗口正被显示，该窗口在选定材质库后不会被刷新。

#### 7. fileSaveAsMatLib()

显示标准 Save File As 对话框，存储当前材质库。

#### 8. fileSaveMatLib()

如果当前材质库在前面被存盘，本方法将当前材质库存为同一文件；如果当前材质库尚未有过存盘，显示标准 Save File As 对话框，存储当前材质库。

#### 9. getMatLibFileName()

返回一个 String 值，表示当前材质库的文件名。

例如：

```
$foo.material = currentMaterialLibrary["Rough Gold"]
for m in sceneMaterials do print m.name
append currentMaterialLibrary $baz.material
```

### 3.5.14 ArrayParameter (数组参数类)

在 3ds max 里一些插件用 ArrayParameter 类值来存储参数，这样用户可以按序号来存取这些参数。

#### 构造函数

<MaxWrapper>.Parameter

其中 Parameter 为一个 ArrayParameter 类数据。

#### 属性

<ArrayParameter>.count

返回 ArrayParameter 里的数据个数。

#### 操作符

<ArrayParameter>.[<integer>]

返回指定序号元素值，序号从 1 开始。

<ArrayParameter>.[<integer>]=<value>

为指定位置元素赋值。

#### 注意

每一个 ArrayParameter 只能允许有一种数据类型或一个与数据类型匹配的控制器。

函数 showProperties() 能指出 ArrayParameter 类数据的数据类型，其输出格式为 <type> array，如 int array、texmap array。

有时也可以为 ArrayParameter 的某一个元素定义一个参数名，如 mapAmounts 是标准材质的一个属性，它是一个 ArrayParameter 类数据，其每个元素存储某一贴图通道的贴图数。为了方便在脚本里存取元素，mapAmounts[1]有一个别名：ambientMapAmount（这个别名同样适用于别的普通贴图）。下面两个语句是相同的：

```
$foo.material.ambientMapAmount.controller
$foo.material.mapAmounts[1].controller
```

下面的脚本用 ShowProperties() 函数来显示 ArrayParameter 数组元素的数据类型并存取 ArrayParameter 元素：

```
m=CompositeMaterial()
showproperties m
m.amount
m.amount[1] *= .5
m=standard()
showproperties m
m.ambientMapAmount = 13
m.mapAmounts[1]
```

-- 创建一个 Composite 合成材质  
-- 显示材质属性  
-- 显示参数个数  
-- 将第一各元素的值减小一半  
-- 创建一个 Standard 类材质  
-- 显示其属性  
-- 通过别名为数组的一个元素赋值  
-- 并显示该元素的改变

输出为：

```
compositematerial:Composite
  .materialList(Material : material array) -- 第 1 行的结果
  .mixType(Composite Type) : int array -- 第 2 行的输出
  .mapEnables(Map Enable) : bool array
  .amount : float array

OK
#(100, 100, 100, 100, 100, 100, 100, 100, 100) -- 第 3 行结果
50.0 -- 第 4 行结果
Standardmaterial:Standard
  .mapEnables(Map Enables) : bool array -- 第 5 行结果
  .maps : texmap array -- 第 6 行输出
  .mapAmounts(Map Amounts) : percent array
  .ambientMap(alias for maps[0])
  .ambientMapAmount(alias for mapAmounts[0])
  .ambientMapEnable(alias for mapEnables[0])
  .bumpMap(alias for maps[8])
  .bumpMapAmount(alias for mapAmounts[8])
  .bumpMapEnable(alias for mapEnables[8])

13 -- 第 7 行结果
13.0 -- 第 8 行结果
```

# 第4章 MAXScript语言 的变量和表达式

## 4.1 变量赋值

变量是各种值的一个载体。我们可以先给变量赋一个值，然后通过变量名又可以重新获取这个值。在本章里，将讨论与变量有关的各种属性。

通常变量赋值的语法为：

<destination> = <expr>

其中<destination>可以是变量名、属性、数组元素。下面将分别讨论每一种<destination>。

MAXScript有一个Swap()方法，可以交换两个<destination>的值，其语法为：

Swap <destination1> <destination2>

### 赋值给变量

在MAXScript里，变量有一个标识符<var\_name>，如果<destination>为变量，表明要给该变量赋一个值，例如：

```
x = 2+2*3
y = [1,2,3]+pos
z = #(1,2, "foo", "bar")
z = #oops
```

### 赋值给存取器

如果<destination>是属性或数组元素，我们称为赋值给存取器。存取器用来存取一个复合值如数组、三维点的某个成员。

在MAXScript里有两种存取器分别对应两种复合值：一种是分量值按可索引的顺序排列，如数组或字符串，另一种复合值由一组有固定名字的分组组成，如三维点。所有3ds max对象，如Box、Sphere、Bend、Modifier都是第二种意义的复合值，所有它们的参数如Height、Angle、贴图文件名都可通过命名分量进行存取。在MAXScript里，这些命名分量也叫做属性。存取器的两种形式为：

<operand>[<expr>] 数组  
<operand>.<var\_name> 属性

例如：

```
$Box01.pos = baseObj.pos +(baseObj.radius*[0,0,1])
z[2] = d.rotation as EulerAngles
```

在MAXScript中，数组的第一个元素序号为1，这一点在对数组进行索引操作时非常

重要。语法定义中允许用户在“.”或“[]”之前使用任何操作数<operand>, 用户可以用下面的形式:

```
my_things[i][j]          --二维数组
my_objects[i].target.position.x    --存取器将按从左至右的顺序求值
```

<operand>还可以是一个块表达式, 例如:

```
(instance myNode).pos=myNode.parent.pos
(find_table "foo")[n - 2] = "fred"
```

其中 find\_table 为自定义函数, 变量 foo 为调用参数。

## 4.2 变量的求值顺序

在赋值语句中, 系统“=”号右面的先对<expr>求值, 然后才对<destination>求值, 在下面例子中可以看到这一点, 在 Listener 窗口中输入:

```
a=#()           --创建一个空数组
a[<print "in <destination>";1>]=(print "in<expr>";10)
```

输出结果为:

```
#()           --第 1 行输出
"in<expr>"   --从<expr>一边输出结果
"in<destination>" --从<destination>一边输出
10           --第 2 行输出
```

在 MAXScript 中, 赋值语句本身也是表达式, 它返回刚刚赋给变量的值, 这样我们可以在别的表达式里使用赋值语句, 或重叠赋值, 例如:

```
X=Y=Z=0           --重叠赋值
#(1,2,a=[0,0,1],4)      --在别的表达式里使用赋值语句
```

## 4.3 变量的作用域

变量有一个被称为作用域的特性, 它决定了在 MAXScript 程序中可以存取一个变量的范围: 全局或局部。全局变量在所有运行的 MAXScript 代码中都可以进行存取并保持它们的值, 直至退出 3ds max。局部变量仅可以在它有效的范围内进行存取操作, 并仅在作用域内保持它们的值, 一旦程序运行超出了局部变量的作用域, 我们就不能存取它们的内容了。这一点与大多数现代编程语言中处理变量的方法是相似的, 下面来讨论在什么情况下, MAXScript 创建一个新的变量作用域。

MAXScript 可以显式地声明并初始化全局变量和局部变量, 其语法为:

```
<variable_decls>:=( local | global)<decl> { ,<decl> }
```

其中<decl>的定义为: <var\_name> [= <expr>], 变量名和可选的初始值, 例如:

```

global baz          --初始值为 undefined
global foo=10, initialized=False   --初始化全局变量
local x = 1,           --续行
    y = 2,
    z =sin(30)

```

如上面提到的，局部变量仅在它们的作用域里才保持它们的值。在某些情况下，作用域可能是一个非常短的周期，因为它会随着一个函数或循环、块语句的开始运行而开始，随着它们的退出而结束。在每次运行脚本程序进入这一作用域时，都会创建一系列新的局部变量，然后到退出作用域时，它们又都会消失。但是在某些情况下，比如脚本卷展栏、工具、插件、控制器和脚本宏等，其在最外层作用域里声明的局部变量将一直在系统里保持它们的值，直至用户重新定义它们。例如：用户第一次运行一个定义好的脚本宏后，最外层作用域里声明的局部变量将会创建，并在重复运行该脚本宏时保持它们的值，除非用户重新定义脚本宏。在某种意义上，我们可以把这些变量当作“私有”的全局变量，它们的值仅在定义它的代码重复运行时得到保持，而不会与其他同名的全局变量冲突。

在 3ds max R2 中，全局变量的初始化仅在创建一个新的变量时才会执行。例如，如果运行下面的程序段二次，第二行里变量 X 的值在第一次运行时为 10，而第二次运行时为 20，在 3ds max R3 以后的版本中，全局变量的初始化总是会被执行。

```

global X=10
Print X
X=20

```

如果在 3ds max R3 中有条件地初始化全局变量，可以用下面的方法：

```
global foo;if foo==undefined do foo=23
```

如果用户没有显式地声明一个变量，而变量名又在高一级的作用域找不到的话，MAXScript 会在用户第一次使用它的时候创建它，并赋给它一个特别的初始值：`undefined`。这样我们并不需要在使用一个变量之前显式地声明它，并对它初始化。未被显式声明的变量称为隐式声明变量。隐式声明变量的作用域为第一次被使用时的有效的 MAXScript 作用域；初始化的 MAXScript 作用域为全局范围，然后在下面情况下会打开一个新的局部作用域：

- ◆ 在脚本文件或 Listener 窗口中最外层的括号“()”函数体的开始。
- ◆ For 循环体的开始。
- ◆ 工具、卷展栏、鼠标右键菜单、脚本宏、鼠标工具定义的开始。
- ◆ 工具、卷展栏、鼠标右键菜单、脚本宏、鼠标工具事件处理程序的开始。
- ◆ When 表达式的开始。

在上面新的局部作用域内，隐式声明的变量和显式声明的变量的作用域为该新开始的作用域以及下一层的作用域。以下面脚本为例：

```

a=10          --全局作用域
(
    b=20        --新作用域：第一级
    for i=1 to 5 do  --新作用域：第二级
        (
            j=random i a
            k=random i b

```

```

print( j*a+k*b)
) --作用域结束: 第二级
a=a+b
) --作用域结束: 第一级
print a
print k

```

在上例中，变量 `a` 在全局作用域里第一次出现，它的作用域包括 `global`、第一级、第二级。变量 `b` 在第一级里首次出现，所以它是一个隐式声明的局部变量，它的作用域为第一级和第二级。

变量 `i`、`j` 在第二级里首次出现，它们的作用域仅为第二级。

变量 `k` 的作用域取决于脚本是否是一次运行。如果脚本第一次运行，`k` 首次在第 5 行出现，它的作用域仅在第二级。在第 11 行，变量 `k` 第二次出现，因为它已超出定义的作用域，系统将会创建一个新的全局变量。第二次运行脚本时，在第 5 行里，MAXScript 检测到变量 `k` 已经存在并使用它。这样第一次运行脚本时，第 11 行将输出 `Undefined`，而第二次运行时会输出第 5 行的运算结果。

### 注意

- ◆ for 循环的计数器（如上面脚本中的变量 `i`）是一个特例，它的作用域总是在 for 循环定义的作用域里，即使计数器变量已经在上一层作用域里隐式或显式地声明。
- ◆ 如果显式声明一个局部变量，即使在高层次的作用域里已经声明该变量，新声明的变量将屏蔽高层次的同名变量。在新作用域里对该变量的引用都使用后面新定义的变量。例如：

```

global foo=23, x=20      --显式声明变量 foo,x 范围为 global
y=10                      --隐式声明变量 y 范围为 global
format "context level global:foo=%\n" foo
if x>y then
(
local baz=foo+1          --最外层括号 "()"，开始了第一个新的作用域
local foo=y-1            --使用全局变量 foo
format "context level 1 : foo=%\n" foo
b=Box                     --隐式声明局部变量 b
b.pos.x=foo               --使用第一个局部变量 foo
if( foo>0)then
(
local a                   --开始第二个作用域
--嵌套的第二个局部变量 foo，屏蔽第一个局部变量 foo
local foo=y-x
format "context level 2 : foo=%\n" foo
a=sin foo                 --使用第二个局部变量 foo
format "a=%\n" a
)
--结束第二个作用域
b.pos.y=foo-1.5           --回到第一个局部变量 foo
Format "context level 1:foo=%\n" foo
)                         --结束第一个作用域，变量 b, baz, foo 不再有效
Format "context level global :foo=%\n" foo
--回到全局变量 foo

```

下面是运行该脚本的输出：

20 --第 1 行结果

```

10          --第 3 行结果
context level global:foo=23      --第 5 行输出
ok          --第 5 行结果
context level 1:foo=9           --第 11 行输出
context level 2:foo=-10         --第 18 行输出
a=- 0.173648                   --第 20 行输出
context level 1:foo=9           --第 23 行输出
ok          --从 6 行到 24 行 if 表达式结果
context level global:foo=23      --从 26 行输出
ok          --第 26 行结果

```

这样重复使用同一个变量名也许看起来有些奇怪，但在一些大型程序里，这些规则是很重要的。比如：如果想向一个 Utility 脚本加入一个新的用户界面控件和它的事件处理程序，将事件处理程序里用到的变量显式地声明为局部变量，可以保证这些变量不会与先前的程序代码里同名的变量发生冲突。当编写一个脚本时，显式地声明变量有助于减少错误和增加程序的可读性。同时，除非确实需要一个全局变量，应尽量将变量声明为局部变量。这样做有以下几点理由：

- ◆ 程序块和函数中的变量名集中在一起声明，可以更容易地找到想要的变量名，以避免使用不正确的变量名。
- ◆ 同时执行几个脚本的时候，如果在两个脚本中使用同名的全局变量，可能会引起互相干扰，从而导致一些不确定的错误的发生。
- ◆ 保证尽管在高层次的作用域里存在同名的变量，也不会受到影响。
- ◆ 显式声明的局部变量的值，当发生运行错误时，会在回调堆栈中显示出来。

在 3ds max R2.5 中，如果一个 3ds max 类、MAXScript 方法或 MAXScript 只读系统变量与隐式声明的局部变量同名，当赋值给该变量，会产生一个运行错误。在 3ds max R3 以后的版本中，MAXScript 在编译语句时，会隐式地声明一个局部变量，而不是将它仍然作为一个只读系统变量。如在 3ds max R2.5 下执行下面的语句，会生成一个 Box 类对象，因为 Box 是一个对象类名；而在 R3 以后的版本中，系统检测到用户想隐式地声明一个局部变量 Box 后，会创建一个名为 Box 的局部变量。

```
If False do Box=10
```

下面的脚本中，因为在第 6 行使用未定义的变量 k，会产生一个错误：

```

fn afunc=
(
    local b
    b=Box()
    for i in 1 to 10 do
        for j in 1 to 10 do
            instance b pos : [i*20,j*30,30*k]
)
.....
(
    global c=10
    local b
    b = "hello"

```

```

    afunc()
)

输出:

    afunc()
--Error occurred in j loop
--Frame :
--k : undefined
--j : 1
--called in i loop
--Frame :
--i : 1
--called in afunc()
--Frame :
--b : $Box:Box102 @ [0.000000,0.000000,0.000000]
--called in <block>()
--Frame:
--b : "hello"
--No ""***" function for undefined
OK

```

如果将第 2 行和 11 行去掉，运行脚本会有下面的输出，因为函数 afunc 和块表达式在不同的作用域，变量 b 每次都被隐式定义为局部变量并包含不同的值，因为它们隐式地声明，它们没有包含在堆栈回调中。

输出:

```

afunc()
--Error occurred in j loop
--Frame :
--k : undefined
--j : 1
--called in i loop
--Frame :
--i : 1
--called in afunc()
--Frame :
--No ""***" function for undefined
OK

```

## 4.4 局部变量和全局变量

如 4.3 节所述，在 MAXScript 里，我们可以创建两类变量，全局变量（Global）和局部变量（Local）。我们可以在程序的任何地方使用这些变量，除了在定义该变量之前，如下面程序会产生一个运行错误：

```

sphere radius : rad
global rad = 10

```

应该写成下面的形式：

```
global rad = 10
sphere radius : rad
```

全局变量一旦被定义，就可以在接下来的脚本程序的任何地方使用。和 AutoCAD 一样，在 MAXScript 里也有一些系统事先定义好的全局变量，称之为系统变量，系统变量由系统创建，并被赋予了默认值，它们中绝大多数可以被重新赋值。关于系统变量的更多信息，请参见 4.5.2 节。

而局部变量仅能在定义它的程序块中使用，所谓程序块是一组 MAXScript 代码，如一个循环或条件语句，下面是一个 for 循环的例子：

```
for i =1 to 3 do
(
local rad =10
s = sphere()
s.pos.x = i *10
s.radius = rad
)
```

局部变量仅能存在于一个程序块内，因此我们只能在一个程序块内部定义一个局部变量，否则系统会给出一个错误信息。

在上例中，变量 rad 仅在括号之间的范围有效，如果试图在别的地方使用变量 rad，也将会得到一个错误信息。尽管上例中变量 rad 在初始时被同时赋值，但这并不是必需的。当你用 local 或 global 标识符来定义变量，在初始化时可以不用赋值，系统仍然可以识别。然而系统将无法确定变量的数据类型，MAXScript 自动把一个特别的值 undefined 赋给它。用户首次赋值给变量时，同时也根据赋值确定了变量的数据类型。实际上，定义变量时赋一个初始值是一个好的办法。

在上例中，除变量 rad 外，变量 s 也是一个局部变量，并不需要为每一个变量定义加上 global 或 local 前缀，MAXScript 会根据变量被定义的位置来判别它的类型。例如，你在一个程序块内部创建一个变量，MAXScript 自动将它视为一个局部变量，而如果用户在程序的顶层（也即在所有程序块之外）创建一个变量，MAXScript 将它视为一个全局变量。请参见下面的例子：

```
for i =1 to 5
(
global rad = 10
cylinder()
c.radius = rad
c.height = cyl_height
c.pos.x = I*5
)
s = sphere radius :rad
```

上例中，共创建了四个变量：rad、cylinder()、c、s，因为 cylinder() 和 cyl\_height 是在一个块

内部被隐式地创建，它们都是局部变量，另外两个变量则是全局变量，其中 s 是因为在程序块的外部被创建，而 rad 则被显式地定义为全局变量。

## 4.5 保留全局变量

保留全局变量是 MAXScript 自动赋值的变量，分为三类：预定义全局变量（如 pi 和 red）、3ds max 系统变量（如 currentFrame）、MAXScript 系统变量。

和保留关键字不一样，用户可以将保留全局变量用作局部变量，将它们赋予指定的值，但此种做法应尽量避免，因为这样极易造成混淆。

### 4.5.1 预定义全局变量

预定义全局变量是一些在脚本中频繁出现的特定的值，这些变量是只读的。

全局变量	说明
True、False	用来作为比较或测试的结果，布尔值
On、Off	同 True、False
pi、e	数学常量 pi=3.1415926535； e=2.718281828
red、green、blue、white black、orange、yellow、 brown、gray	一系列预定义的颜色值
x.axis、y.axis、z.axis	定义三主轴的标准矢量点。分别为[1, 0, 0]、[0, 1, 0]和[0, 0, 1]
OK	一些函数或表达式的返回 void 值
undefined	所有变量和数组元素的初始值，一般只用于测试一个值是否等于 undefined。如果用户试图对一个值为 undefined 的变量执行任何操作，MAXScript 会产生一个错误信息
unsupplied	所有函数关键字类型变量的初始值。关键字类型变量对函数而言是可选的，如果函数调用时未提供关键字类型变量的值，而函数定义中又没有为它指定一个默认值，MAXScript 将把值 unsupplied 赋给它。用户可以用这个值来测试函数是否指定了某一特定关键字类型变量值。unsupplied 在 for 循环中还有别的意义，详见 5.4 节

### 4.5.2 3ds max 系统变量

3ds max 系统变量为用户提供了获取 3ds max 系统信息的途径，除了那些只读的系统变量以外，用户可以获取或赋值给系统变量。

3ds max 系统变量	说明
activegrid	包含当前活动网格。如果当前活动网格为 Home Grid, 返回值为 undefined。用户可以将任一网格节点对象赋给该变量, 以指定该对象为当前活动网格
ambientcolor	存储/设定渲染环境 ambient (环境光) 的颜色值。该变量也可以在菜单 Rendering   Environment 里设定
ambientcolorController	存储/设定渲染环境 ambient (环境光) 颜色值的控制器
animationrange	存储/设定当前动画时间范围, 一个 Interval 类型的值。该变量也可以在 Time Configuration 对话框中设定
animButtonEnabled	指定用户是否可以改变 Animate 按钮的状态, 布尔值。在脚本文件中, 可以用系统变量 animButtonState 来改变 Animate 按钮的值, 而与另一个系统变量 animButtonEnabled 无关 False: 表示不能改变; True: 表示可以改变
animButtonState	存储/设定 Animate 按钮的状态, 布尔值。 True 表示 Animate 按钮打开; False 表示 Animate 按钮关闭
autoBackup.enabled	存储/设定是否启用自动备份功能, 布尔值。
autoBackup.time	存储/设定两次自动备份之间的时间间隔, 浮点数。以分钟为单位。
backgroundColor	存储/设定渲染环境的背景颜色值。该变量也可以在 Rendering   Environment 菜单里设定
backgroundColorController	存储/设定渲染环境背景控制器
backgroundImageFileName	存储/设定视窗背景图像文件名, 字符串。该变量也可以在 Viewport Background 对话框里设定
cui.commandPanelOpen	存储/设定是否显示命令面板, 布尔值。 True 表示显示; False 表示不显示
currentMaterialLibrary	包含当前打开材质库里所有材质和根目录级贴图的虚拟数组, 只读。用户可以在 for 循环里通过数组索引获取材质库里所有的材质, 还可以直接用数组名、材质名来获取材质
displayGamma FileInGamma FileOutGamma	存储/设置定义 gamma 参考设置的浮点值。用户可以用这些全局变量来为用 MAXScript 创建的位图指定 gamma 值, 例如: <pre>b:bitmap 32 240 gamma:displayGamma render camera:c to:b</pre> 这样如果位图用来作为卷展栏位图或按钮位图, 将用当前 Gamma 值来显示。该变量也可以在 Customize   Preference   Gamma 选项卡里设置
DisplaySafeFrames	存储/设置是否为活动视窗打开 Show Safe Frame 开关, 布尔值。如为 True, 打开 Show Safe Frame
environmentMap	存储/设置用户定义的渲染环境纹理贴图。也可以在 Rendering   Environment 里设置
flyOffTime	存储/设置从鼠标单击到按钮引出弹出式命令之间的间歇时间, 以毫秒为单位, 整数。该变量也可以在 Customize   Preference   General 选项卡里设置
globalTracks	一个 MAXTVNote 值, 表示在 Track View 视窗里最高一级 Global Tracks 轨节点, 只读

(续表)

3ds max 系统变量	说明
hardwareLockID	只读整数, 3ds max 硬件锁 ID
hotspotAngleSeparation	聚光灯的聚光区 (Hotspot) 和衰减区 (fallOff) 的角距值, 约束聚光区角度, 使之不能与衰减区角度相等, 只读浮点数。该变量也可以在 Customize   Preference   Rendering 选项卡里设定
Keyboard.ShiftPressed Keyboard.ControlPressed Keyboard.altPressed	存取变量读取时, 键盘上 Shift、Control、Alt 键是否被按下, 只读, 布尔值
LightTintColor	存储/设置渲染环境里 Global Lighting Tint 的颜色值。该变量也可以在 Rendering   Environment 里设置
LightLevel	存储/设置渲染环境里的 Global Lighting Tint Level 值, 浮点数。也可以在 Rendering   Environment 里设置
LightLevelController	存储/设置渲染环境下 Global Lighting Tint Level 控制器。该变量也可以在 Rendering   Environment 里设置
Listener	代表 Listener 编辑窗口的<Window Stream>值, 只读
LocalTime	定义当地日期和时间的字符串, 只读, 形式如: “4114197 10:24:57AM”, 其格式由 Windows 系统控制面板的“区域设置”控制
Logsystem.quietmode	存储/设置当渲染器发现错误时, 是否显示对话框, 布尔值。 False: 渲染器检测到的错误信息会在一个对话框中显示; True: 不显示错误信息对话框。当渲染器检测到一个错误时, 会退出渲染
MacroRecorder	代表宏记录编辑窗口的<Window Stream>值, 只读
MaxFileName	当前打开场景的文件名, 字符串, 只读
MaxFilePath	当前打开场景文件的路径, 字符串, 只读
MeditMaterials	一个虚拟数组, 包含 Material Editor 窗口下的所有材质和根级贴图。用户可以使用数组索引的方式来存取材质和根级贴图, 还可以用 Name 类及 String 类直接指定材质和根级贴图的名称来进行索引。例如: <pre>\$foo.material = meditMaterials[1] meditMaterials["foo mat"].diffuse = red for m in meditMaterials do print m.diffuseMap meditMaterials[1]=standard() print meditMaterials.count</pre>
NumEffects	存储在 Rendering   Effects 对话框里定义的渲染效果数目, 只读整数
NumAtmospherics	存储在 Rendering   Environment 里设置的大气效果事件数目, 只读整数
numSubObjectLevels	存储当前修改器堆栈里被选择的修改器或对象支持的子对象级数, 只读整数。如果 Modify (修改) 面板没有打开或没有选择对象, 该变量值为 undefined
PlayActiveOnly	存储/设置是否仅支持活动视窗回放, 布尔值。也可以在 Time Configuration 对话框中设定
Preferences. ConstantReferenceSystem	存储/设置在 3ds max 工具栏中的 Move、Rotate、Scale 工具是否使用同一坐标系, 布尔值。该变量与 Customize   Preference   General 选项卡下 Ref.Coord.System 里 Constant 复选框对应
Preferences.flyOffTime	同 flyOffTime

(续表)

3ds max 系统变量	说明
Preference.MaximumGbuffer-Layers	存储/设置渲染中允许产生的 G-buffer 层的最大数目, 整型
Preferences.SpinnerWrap	存储/设置用鼠标拖动微调器改变对象的属性值时, 是否限制鼠标指针靠近该微调器, 布尔值。该变量与 Customize   Preference   General 选项卡下 Spinner 项里 Wrap Cursor Near Spinner 复选框对应
Preferences.UseLargeVertex-Dots	存储/设置当将顶点显示成圆点时, 是用小点还是用大点来显示, 布尔值。该变量仅在变量 UseVertexDots 设为 True 时有效, 也可以在 Customize   Preference   Viewport 选项卡里设置
Preferences.UseTransform-Gizmos	存储/设置是否使用转换 Gizmo, 布尔值, 该变量也可以在 Customize   Preference   Viewport 选项卡里设置
Preferences.UseVertexDots	存储/设置是否将顶点显示成圆点, 布尔值。 True: 面片的顶点显示为圆点 (dots); False: 面片的顶点显示为圆点 (ticks)。该变量也可以在 Customize   Preference   Viewport 选项卡里设置
realtimePlayback	存储/设置是否支持实时模式下的回放, 布尔值。该变量也可以在 Time Configuration 对话框里设置
renderer	指定渲染器采用哪种渲染模式: #draft 或 #production, 例如: <pre>If renderer = #draft then..... Renderer = #production render camera:c.....</pre>
renderDisplacements	存储/设置是否在渲染时执行贴图位移, 布尔值。该变量也可以在 Render Scene 对话框里设定
renderEffects	存储/设置在渲染完一个场景后是否执行渲染特效, 布尔值
renderHeight	存储/设置渲染输出图形的高度, 整型。该变量也可以在 Render Scene 对话框里设定
renderPixelAspect	存储/设置渲染输出图形的像素比例, 整型。该变量也可以在 Render Scene 对话框里设定
renderWidth	存储/设置渲染输出图形的宽度, 整型。该变量也可以在 Render Scene 对话框里设定
renderOutputFilename	存储/设置渲染输出文件名。该变量也可以在 Render Scene 对话框里设定。如果该变量为空字符串 “”, 表示 Render Scene 对话框里的 Save File 复选框未被选中
rootNode	目前场景的根节点, 只读。根节点在场景中并不真实存在, 仅表示它为场景里所有节点的父节点, 它们都可以用根节点的.children 属性获取。如果用户试图对一个根节点执行除.children 之外的节点操作, 将导致一个运行错误
SceneMaterials	包含当前场景中所有材质和根目录级贴图的虚拟数组。我们可以用数字、材质名、贴图名来索引任一种材质、贴图, 只读
ScriptsPath	当前系统 Scripts 路径完整路径名, 只读字符串
SelectionSets	一个虚拟数组, 包含所有 3ds max 工具栏 Named Selection Sets 下拉列表中全部命名选择集
ShowEndResult	存储/设置修改面板里 Show End Result 图标的状, 布尔值

(续表)

3ds max 系统变量	说明
SkipRenderedFrames	存储/设置渲染时是否跳过某些帧, 布尔值
SliderTime	存储/设置与 3ds max 用户界面里时间滑杆对应的时间值
SnapMode.active	存储/设置 Snap toggle 的状态, 布尔值, 本变量在 3ds max 里不可用
SnapMode.type	存储/设置捕捉模式, Name 类值, 有效值为: #2D (二维); #2.5D (2.5 维); #3D (三维)。本变量仅能在 MAXScript 脚本程序中使用, 在 3ds max 里不可用
SubObjectLevel	Modify (修改) 面板被打开时, 存储/设置其子对象层次, 整型, 可为 0, 最大值为系统变量 numSubObjectLevels; 如果 Modify 面板未被打开, 或当前 Modifier (修改器) 的子对象层次被禁止, 那么 SubObjectLevel 的值为 undefined。例如: <pre>b=Box() --创建一个 Box 对象 em=edit_Mesh() --创建一个 Edit Mesh 修改器 addModifier \$Box01 em --添加修改器 select \$Box01 --选择对象 Print SubObjectLevel --打印当前子对象层次 Edge SubObjectLevel=2 --将子对象层次设为</pre>
SysInfo.DesktopSize	返回一个以像素为单位的 Point2 值, 表示桌面的大小, 只读
SysInfo.DesktopBpp	表示桌面颜色深度, 单位为位/像素, 只读, 整型
SysInfo.MaxPriority	存储/设置 3ds max 的处理优先级别。有效的优先级别为: #high、#normal、#low
ticksPerFrame	设置系统时间精度, 只读变量, 整型
timeConfiguration.playActiveOnly	同 PlayActiveOnly
TimeConfiguration.UseTrackBar	存储/设置 Time Configuration 对话框下 Key Steps 项里 Use Track Bar 复选框的状态, 布尔值
Trackbar.filter	存储/设置过滤器, 指定在 Track Bar 里显示哪些类型的关键帧。有效值为: #all、#TOnly、#currentTM、#object、#mat
Trackbar.visible	存储/设置 Trackbar 是否可见, 布尔值
TrackViewNodes	同 globalTracks
Units.DisplayType	存储/设置显示单位类型。有效值为: #Generic、#Metric、#Us、#Custom
Units.MetricType	存储/设置米制单位的类型。有效值为: #Millimeters、#Centimeters、#Meters、#Kilometers
Units.UsType	存储/设置 US 单位类型。有效值为: #Frac_In、#Dec_In、#Frac_Ft、#Dec_Ft、#Ft_Frac_In、#Ft_Dec_In
Units.USFrac	存储/设置 US 单位里分数显示类型。有效值有: #Frac-1-1、#Frac-1-2、#Frac-1-4、#Frac-1-8、#Frac-1-10、#Frac-1-16、#Frac-1-32、#Frac-1-64、#Frac-1-100
Units.CustomName	存储/设置当前用户定义的单位名称, 字符串
Units.CustomValue	存储/设置当前用户定义的单位数值, 浮点数
Units.CustomUnit	存储/设置当前用户定义的单位类型, 有效值为: #Inches、#Feet、#Miles、#Millimeters、#Centimeters、#Meters、#Kilometers

(续表)

3ds max 系统变量	说明
Units.SystemScale	存储/设置当前系统单位的比例因子，浮点数。该变量也可以在 Customize   Units Setup 对话框的 System Unit Scale 组里设定
Units.SystemType	存储/设置当前系统单位比例因子类型。有效值为：#Inches、#Feet、#Miles、#Millimeters、#Centimeters、#Meters、#Kilometers。该变量也可以在 Customize   Units Setup 对话框的 System Unit Scale 组里设定
UseEnvironmentMap	存储/设置是否在渲染环境里使用贴图，布尔值
VideoPostTracks	存储 Video Post Track View 顶级节点里可容纳的最多 Track View 节点数目。该变量在 3D Studio VIZ 里为 undefined，只读
Viewport.activeViewport	存储/设置活动视窗的索引号
Viewport.numViews	存储当前视窗布局中视窗的数目，只读整数

下面的系统变量为 3ds max 默认的 A-Buffer（A 缓存）渲染方式特有的，如果当前渲染方式不为 A-Buffer，它们的值均为 undefined。

系统变量	说明
scanlineRender.antiAliasFilter	存储/设置 anti-Alias 过滤器。如果要获得所有过滤器的清单，可以用下面的语句： <code>showClass "* .filter"</code>
scanlineRender.antiAliasFilterSize	存储/设置 anti-Alias 过滤器大小
scanlineRender.enablePixelSampler	控制是否采用全局超级采样，布尔值

#### 4.5.3 MAXScript 系统变量

MAXScript 系统变量用来在脚本程序中存取 MAXScript 系统的状态。

MAXScript 系统变量	说明
currentTime	存储当前 at time 子句中的时间值，以帧为单位，只读变量。如果当前没有 at time 子句，则返回当前用户界面时间滑标里的时间值；而如果正在渲染时，则返回当前正在渲染的帧数
editorFont	存储/设置定义 Script Editor 窗口里字体名的字符串。该变量对当前打开的和接下来要打开的 Editor 窗口、Listener 窗口都有效，也可以在 Customize   Preference   MAXScript 里设置
editorFontSize	存储/设置 Script Editor 窗口字体的大小，整型。该变量对当前和接下来打开的全部 Editor 窗口和 Listener 窗口都有效，也可以在 Customize   Preference   MAXScript 里设置
editorTabWidth	存储/设置 Script Editor 窗口里 Tab 键的宽度，以字符为单位，整型。该变量对当前和接下来打开的全部 Editor 窗口和 Listener 窗口都有效，也可以在 Customize   Preference   MAXScript 里设置
escapeEnable	存储/设置是否允许用 Esc 键来中止脚本的执行，布尔值。当我们在进行一个很长的计算时，也许不希望用户用 Esc 键来中止程序的运行，可以在界面里放一个进度显示框来显示计算机的进展，再另外设置一个 Cancel 键
HeapFree	存储当前 MAXScript 可用的内存数量，只读整数

(续表)

MAXScript 系统变量	说明
HeapSize	存储/设置当前分配给 MAXScript 的内存尺寸。MAXScript 使用为 3ds max 分配内存之外的内存，用户可以随时改变 HeapSize 的值，为 MAXScript 增加内存，例如： <code>HeapSize += 1000000</code>
inputTextColor	存储/设置 Listener 窗口输入文本的颜色
messageTextColor	存储/设置 Listener 窗口错误信息文本的颜色
outputTextColor	存储/设置 Listener 窗口输出文本的颜色
options.oldPrintStyles	为与 3ds max 以前版本兼容而设，布尔值
stackLimit	堆栈是 MAXScript 里的一些保留内存，用来存储如：过程函数的调用返回地址、传递参数、局部变量等状态数据。StackLimit 默认值为 1MB，可以重新赋值以加大堆栈尺寸，例如： <code>StackLimit *= 2</code>
?	仅在 MAXScript Listener 窗口里使用的特殊变量，保存 Listener 窗口里上一次的求值结果

## 4.6 持续型全局变量

MAXScript 支持限制格式的持续型全局变量。用户可以将一个全局变量声明为持续的，它的值将同场景一起存储，当重新打开场景时，这些变量的值会被恢复。可以用保留关键字 `persistent` 来声明一个持续性全局变量，例如：

```
persistent global foo, baz, bar
```

这样变量 `foo`、`baz`、`bar` 在存储场景时被一起存储到 `max` 场景文件中了，当重新打开场景文件时，变量 `foo`、`baz`、`bar` 将被恢复，并被隐式地声明为全局变量。

目前对持续性全局变量的限制是，仅有几种类型的值能存储到场景文件中，它们是：`Integer`、`Float`、`String`、`Color`、`Time`、`Interval`、`Array`、`Point3`、`Ray`、`Quat`、`AngleAxis`、`EulerAngles`、`Matrix3`、`Point2`、`Undefined`、`OK`、`Boolean` 和所有 3ds max 对象 (`Node`、`modifier`、`Controller`、`Material` 等)，其他类型的值如果作为持续型全局变量，场景恢复后值都为 `undefined`。

对数组 (`Array`) 而言，只有那些数据类型为上表中所列类型的元素能被恢复，另外类型的元素在恢复的数组中也为 `undefined`。

当执行 `File | Reset`、`File | New` 和 `File | Open` 操作时，当前文件里定义的持续型全局变量会从持续型全局变量清单里清除，但它们仍然会作为普通的全局变量存在。由此可知持续型变量并不会在打开和关闭文件时“粘住”每一个文件，成为所有文件的一部分。

下列方法用来显示和删除 `persistent` 变量：

1. `Persistents.show()`  
显示 Persistent 变量清单及变量值。
2. `Persistents.remove <var.name>`

把指定的变量从 Persistent 变量库里删除，但它仍然是一个全局变量，并保持它的值。

### 3. Persistents.Remove All()

清除所有持久性变量，但它们仍然作为全局变量并保持它们的值。

如果一个 Persistent 变量包含一个 3ds max 对象，但场景中并没有这一类的对象，该值将不能在文件存储/打开时被存储/恢复。例：

```

persistent global global_array = #()
global_array[1] = b= Box()
global_array[2] = bm= bend()
global_array[3] = sm= standard()
global_array[4] = fog()
global_array[5] = area()
global_array[6] = bezier_float()
global_array[7] = br= bricks()
global_array[8] = lookat()
global_array[9] = blur()
global_array[10] = MapScaler()
--b.material=sm
--addmodifier b bm
--sm.diffusemap=br
persistents.show()
max hold
max fetch
persistents.show()

```

上例中只有 Box 对象被显示。如果三行注释语句被执行，则 material、modifier 和 map 也会被显示。

## 4.7 变量的几个特性

对 MAXScript 的变量而言，有以下几个特性。

### 类型自由

变量里可存储任意类型的值，而且可以多次赋给同一个变量不同类型的值。例如：可以把一个 Point3 类型的值赋给一个变量，在后面又可以赋给它一个 Float 类型的值，再赋给它一个 Array 类型的值。MAXScript 里的变量与 C 语言和 Java 语言不同，它的变量是类型自由的。这种自由使 MAXScript 的使用更简单。

类型自由的同时，MAXScript 的变量还有“类型安全”的特性，也就是用户不能对一个变量里的值执行错误的操作。例如：

```

x=$Box01.position      --一个 Point3 类型值
x="foo"                --一个 string 类型值
x=x+2

```

试图给 String 添加一个 Number 类型数值最后一句将导致 MAXScript 产生一个错误信息：

```
Error:Unable to convert 2 to type string
```

### 引用赋值

MAXScript 里使用一种名为“引用赋值”(Reference Assignment)的赋值机制。当用户创建一个值时，系统为该值分配一个内存，然后指向该值的一个指针或引用被放在变量里面，而不是值本身，如果用户又把该变量的值赋给另一个变量，对第一个变量内存的指针或引用将放到新的变量里。如果将第一个变量的值赋给一个数组的元素，或将它传递给一个函数的调用参数，其作用原理与上面相同。当给一个变量赋新值或当它超出了它的作用域时，对该变量的引用将取消，如果程序全部取消了对某一值对应内存的引用后，该值所占用的内存就可以重新被系统分配使用。

与引用赋值相对应的一种赋值机制叫做值赋值 (Value Assignment)：系统会将要赋的值对应的内存复制一份赋给变量。值赋值与引用赋值相比有一个优点：每次一个变量或数组元素被重新赋值，旧的变量所占的内存可以马上被重新分配，因为没有别的变量引用它，而对于引用赋值，系统不能马上重新分配旧变量值占用的内存，因为可能还有别的变量仍然指向该内存。

由此在 MAXScript 里产生一个新的任务：内存收集 (Reclaim)，在诸如 C 语言和 Pascal 等编程语言中，当内存里的值不再被使用，必须使用显式地内存收集语句，而在 Java 和 Lisp、Smalltalk 等语言中，提供了内存自动收集的功能，MAXScript 也有内存自动收集功能。

如果将一个变量的值赋给另一个变量，与该值对应的内存引用被放到新的变量里，这样两个变量实际上都在引用同一个值，如果给其中一个变量赋一个新值，这个变量将包含对新值的引用，而另外一个变量仍然包含对旧值的引用。一般而言，这正是我们想要的结果。因为我们不希望两个变量都包含对新值的引用，但是对于一个复合值，会有一点复杂，比如一个三维点 (Point3) 值被几个变量同时引用，如果给它的一个成员赋一个新值，此时并没有创建一个新的复合值，这样原来引用该 Point3 值的所有变量仍然包含对该值的引用，并会“看到”它们成员的值被改变了。如下面的例子：

```
( a = "hello World"           -- 创建一个 string 值，并将其引用放在 a 里
  b = a                      -- 将 string 的引用放到 b 里
  format "a=%;b=%\n" a b      -- 打印 a、b 的值
  b = "thanks for the fish"   -- 创建一个新 string 值，并将其引用放在 b 里
  format "a=%;b=%\n" a b      -- 打印 a、b，彼此不同
  b = a                      -- 将 string 引用放到 b 里
  a[1] = "J"                  -- 改变 a 第一个字符的值
  format "a=%;b=%\n" a b      -- 打印 a、b，彼此相同
  a=b=[10,20,30]              -- 将一个 Point3 类型值的引用放在 a、b 里
  format "a=%;b=%\n" a b      -- 打印 a、b
  a.x = -100                  -- 改变成负值
  format "a=%;b=%\n" a b      -- 打印 a、b，彼此相同
)
```

输出：

```
a=hello World ; b=hello World
a=hello World ; b=thanks for the fish
```

```
a=Jello World ; b=Jello World
a=[10,20,30] ; b=[10,20,30]
a=[-100,20,30] ; b=[-100,20,30]
Ok
```

有时上面这样的结果是我们需要的，有时我们并不希望这样，如果我们不希望这样，MAXScript为大多数数据类型定义了一个copy方法来为值创建一个单独的副本，在一些数据类型如Array（数组）和Structure（结构）里，数据成员值本身可能也是复合值，此时复制操作仅在顶级值本身（如数组）进行，而不包括值里包含的复合值（如数组元素）。

### 变量内存空间分配和空间释放及手工释放内存空间

MAXScript默认的工作内存为5.5MB。该默认值对大多数任务而言已足够，但用户仍然可以通过增加MAXScript系统变量heapSize的值来增加MAXScript可分配的内存，例如：

```
heapSize += 2000000 --增加2MB，总共7.5MB
```

如果用户总是需要将额外的内存分配给MAXScript，可以把上面语句加到Startup.ms文件里或在3ds max的菜单Preference|MAXScript选项里设置。

增加MAXScript可分配内存会减少3ds max其他任务可用的内存空间，所以应谨慎对待系统变量heapSize的大小，一般而言，仅在收到Out of memory的错误信息或经常因系统自动执行内存收集操作而暂停程序运行后才增加heapSize的值。

在3ds max里用户可以执行任意次数的加大heapSize变量值的语句，但不能减小它，重新启动3ds max会将heapSize的值重置为其默认值。

MAXScript使用一种名为内存自动收集的内存管理机制，这意味着当一个值不再需要时，我们不要显式地释放该值所占的内存。当MAXScript发现没有内存可用时，可在分配给它的内存里执行一次内存收集操作，在搜索那些可被重新分配的内存时会导致程序运行的短时间暂停。自动内存收集的优点是用户可以随意创建所需的变量，而MAXScript会自动清除那些不再需要的变量。

一般情况下内存收集工作会自动进行，不需要我们手工干涉，但在有些情况下，我们也许希望因内存收集而导致的运行暂停尽量不出现在运行程序中间，这时我们可以用系统提供的函数gc()，显式地通知系统进行一次内存收集。

函数gc()还可以关闭那些不再使用的文件。有时当错误发生时，一个文件尚处在打开的状态，而对该文件对象的引用存储在一个局部变量里，那么我们没有办法在Listener窗口里关闭这个文件，此后用户如果试图重新打开该文件，会导致一个Already Open的错误。执行函数gc()可以强制关闭该文件并释放引用文件对象的变量。

函数gc()返回本次内存收集后的可用内存数量。如果内存里存在许多小的可重新分配的对象，一次内存收集可能会需要几秒钟的时间。

## 4.8 表达式

MAXScript是一种基于表达式的语言，每一个语句都是表达式，并返回一个计算结果。

比如下面两行代码都是合法的：

```
if a>b then print c else print d
x = if a>b then c else d
```

也可以用下面的语句，包含一个嵌套的 if 表达式和一个嵌套的赋值语句：

```
x = if( if a>b then c else d)<e then f else(g=23)
```

另外一个例子是块表达式，其语法定义为`<block.expr>`，它包含在一对括号里的一系列`<expr>`表达式：

```
(  
    print a  
    print b  
    if a>b then print "the big a"  
)
```

一个 MAXScript 程序是由许多`<expr>`表达式组成，一个`<expr>`可以是下面几种形式：

<code>&lt;simple_expr&gt;</code>	--简单表达式
<code>&lt;variable_decls&gt;</code>	--变量声明
<code>&lt;assignment&gt;</code>	--赋值语句
<code>&lt;context_expr&gt;</code>	--关联语句
<code>&lt;if_expr&gt;</code>	--条件表达式
<code>&lt;while_loop&gt;</code>	--while 循环表达式
<code>&lt;do_loop&gt;</code>	--do 循环表达式
<code>&lt;case_expr&gt;</code>	--case 表达式

## 4.9 简单表达式

简单表达式主要包括赋值和一些常规编程语句，例如：

```
a+b*c  
sinx
```

简单表达式由两部分组成：操作数（a、b、c 和 x）和操作符（=、- 和 sin）。下面分别讨论 MAXScript 里的各种简单表达式。

### 4.9.1 数学表达式

MAXScript 里的数学表达式对应数学里的数学表达式，其构造方法有：

```
<math_operand>+<math_operand>
<math_operand>-<math_operand>
<math_operand>*<math_operand>
<math_operand>/<math_operand>
<math_operand>^<math_operand>
<math_operand> as <class>           类型转换
```

#### 4.9.2 比较表达式

我们用比较表达式来比较两个可比较的值，其比较结果为布尔值：True 或 False，比较表达式有下面几种形式：

```
<compare_operand> == <compare_operand>
<compare_operand> != <compare_operand>
<compare_operand> > <compare_operand>
<compare_operand> < <compare_operand>
<compare_operand> >= <compare_operand>
<compare_operand> <= <compare_operand>
```

#### 4.9.3 逻辑表达式

逻辑表达式用来连接布尔表达式（如比较表达式）来构造复杂的条件表达式，逻辑表达式形式有：

```
<logical_operand> or <logical_operand>
<logical_operand> and <logical_operand>
[not] <logical_operand>
```

和大多数编程语言一样，MAXScript 里的逻辑表达式是非严格的，这句话有两个含义：

- ◆ 在一个 and 表达式里如果第一个操作数的求值结果为 False，其结果一定为 False，这样，第二个操作数就不用求值了。
- ◆ 在一个 or 表达式里，如果第一个操作数的求值结果为 True，其结果一定为 True，这样，第二个操作数就不用求值了。

这样可以节省程序执行时间，如下例：

```
if a != undefined and (sin a) > 0 then .....
```

#### 4.9.4 函数调用表达式

函数调用表达式把变量的值传递给 MAXScript 系统函数或用户自定义的函数，由函数返回的值称为调用结果，函数调用有以下几种格式：

<operand> {<argument>}	一个或几个变量
<operand>()	没有变量

例如：

```
sin a
a
tan z x y
print(n+1)to : debug
move object 1 [10,20,x*3]
```

大多数编程语言的函数调用语法为一对括号括里用逗号隔开的变量列表，但

MAXScript 的函数调用语法和其他编程语言都不同（参见本书 6.1 节），MAXScript 选择这种函数调用语法有下面几个理由：使用简单、方便，特别是如果一个函数有许多可选的关键字变量。

MAXScript 运行的平台 3ds max 是一种命令行窗口，这种语法和 3ds max 其他命令的格式类似。在 3ds max 里，包括 Geometric、Spline、Material 等对象都是使用函数调用来创建的，创建对象的函数名就是对象的类名。例如：

```
standard()          -- 创建一个标准标质
bend angle :45    -- 创建一个 bend 修改器
Box height :20 width :40 -- 创建一个 Box 对象
```

#### 4.9.5 块表达式

块表达式是把一序列表达式作为一组，系统对表达式按顺序求值，其语法为定义：

$(<\text{expr}> \{ ; | <\text{eol}>\} <\text{expr}> )$

在 MAXScript 里也允许用空块表达式：(), 有时我们可以在程序中放一个空块表达式，留待以后填充它，当求值时，空块表达式返回 undefined。例：

```
(  
    d = center.x^2 -(p.x^2 + p.y^2)  
    newz = if d > 0 then sqrt d else p.z  
    print newz  
)  
if x < y then(print x; print y; u += 10; print u)  
(  
    print x; print y  
    u += 10  
    print u; print z  
)
```

块表达式的最后一个表达式求值结果作为块表达式的返回值，因此可以使用下面的语句：

```
$ Box .pog.z =  
( d=x^z-y^z  
    if d>0 then(sqrt d)else z  
)
```

#### 4.10 关联表达式

关联表达式是 MAXScript 专为 3ds max 设计的一种语法形式，它实际上将 3ds max 用户界面中最重要的一些操作引入到 MAXScript 中来了，如 Animate 按钮、时间滑杆、工作坐标系等，关联表达式可以让用户像在用户界面中一样创建动画和操作 3ds max 对象。关联表达式的例子如：

```
animate On  
(
```

```

move $Box01 [ 20, 0, 0 ]
rotate $Box02 45 x_axis
)

```

上面语句在执行块表达式期间打开 Animate 按钮，并自动生成关键帧，关联表达式语法为：

<context> {,<context>} <expr>

其中<context>有以下几种：

[ with ] animate <boolean>

at level <node>

in <node>

at time <time>

[ in ] coordsys <coordsys>

about <center\_spec>

[ with ] undo <boolean>

接下来分别讨论每一种关联语句。

#### 4.10.1 animate

关联语句 animate 对应 3ds max 界面 Animate 按钮的设置，其格式为：

[with] animate <boolean\_expr>

把表达式或操作块表达式放在 animate On 关联语句内，任何对 3ds max 对象的可动画属性的改变都将产生关键帧，而如果将之放在 Animate Off 关联语句内，即使用户界面上 Animate 按钮处在 On 状态，也不会产生关键帧，如果改变 3ds max 对象可动画属性的语句不在 Animate 语句之内，则仅在 Animate 按钮处于 On 状态时才会产生关键帧。

设置 Animate 关联语句的 On/Off 状态并不影响界面上 Animate 按钮的状态。例如：

```

loadheight=obj.pos.z          --将对象抬高，但不产生关键帧
animate Off
    obj.pos.z=loadheight+LiftHeight

```

在时间 tStart 处创建关键帧，如果 tStart != 0 会在 0 帧处自动创建一个关键帧，然后在时间 tEnd 处创建关键帧：

```

with animate On
(
    at time tStart obj.pos.z=loadheight+LiftHeight
    at time tEnd obj.pos.z=loadheight
)

```

#### 4.10.2 at level、in

关联语句 at level 将指定节点对象作为关联语句里指定路径名或创建的场景对象的根节点，其格式为：

at level <node>

in <node>

系统将自动把指定<node>作为关联语句里任何路径名的前缀，或作为关联语句里创建的场景对象的父对象。在顶级名称前加上一个根路径符“/”将告诉系统从对象树的根路径开始查找对象，而不是从当前工作级别开始，例如：

```
scale $/baz/*[1,1,2]
```

又如\$foo 将查找场景里所有名为 foo 的对象，而 \$/foo 将仅在顶级里查找名为 foo 的对象。例子：

```
in $mannequin01...LClav...hand
(
rotate $thumb1 15 y_axis
rotate $finger11 10 y_axis
PalmCenter=$finger21.parent.pos
PalmCenter+=(LHandPos-$finger21.pos)/2.
--创建一个 dummy 对象作为 hand 对的子对象
dummy name: "palmLink" pos:PalmCenter
)
in $dummy
(
sphere name: "ear1" pos:[10,10,10]
sphere name: "ear2" pos:[-10,10,10]
scale $foo/* [1,1,2]
)
```

#### 4.10.3 at time

关联语句 at time 对应 3ds max 用户界面里的时间滑标，其格式为：

```
at time <time>
```

任何可动画(Animatable)属性的改变将在指定的时间发生，而如果同时 Animate 按钮处在 On 状态，或是脚本文件里处在 Animate On 关联语句里，这些改变将产生一个关键帧。

设置 at time 关联语句并不会影响 3ds max 的时间滑标，用户可以用系统变量 SliderTime 来查询 3ds max 时间滑标的当前时间值。例如：

```
animate On
for t in 0 to 100 by 5 do
  at time t
  --生成一个关键帧动画，对象 foo 随机追逐对象 baz
  $foo.pos = $baz.pos + random [-10,-10,-10] [10,10,10]
```

#### 4.10.4 coordsys

关联语句 coordsys 对应 3ds max 主工具栏的 Reference Coordinate System (当前引用坐标系) 下拉列表，其格式为：

```
[in] coordsys<coordsys_spec>
```

用来强迫三维坐标操作在指定坐标系里执行。在一个 coordsys 关联语句里执行的对象 move、scale、rotate 操作或设定转换有关的属性都以指定坐标系为参照坐标系，在一个

`coordsys` 关联语句里存取坐标属性如位置、中心点、转角等也是以指定坐标系作为原点：

`coordsys` 语句有以下几种形式：

```
[in] coordsys word          --使用世界空间坐标系
[in] coordsys local         --使用对象局部坐标系
[in] coordsys parent        --使用父对象的坐标系
[in] coordsys grid          --使用活动网格坐标系
[in] coordsys screen        --使用当前活动视窗
[in] coordsys <node>        --使用指定节点对象的局部坐标系，与 3ds max 界面里的
                             --Pick 坐标系相同
[in] coordsys <matrix3>     --使用由<matrix3>指定的坐标系
```

例如：

```
in coordsys local Selection.pos=random [-20,20,20] [20,20,20]
in coordsys parent rotate Selection(Euler Angles 0 0 90)
```

#### 4.10.5 about

关联语句 `about` 对应于 3ds max 主工具栏里的 Rotate | Scale Transform Center 列表工具，其格式为：

`about <center_spec>`

定义了关联语句里的 Scale、Rotate 操作的中心点，注意和 3ds max 的 Center 列表一样，关联语句仅指定操作的中心点，而不是坐标轴，坐标轴仍然使用由 `coordsys` 语句定义的坐标系定义的坐标轴，`about` 关联语句有下面几种形式：

```
about Selection           --以当前选择集的中心为中心进行 rotate、scale 操作
about pivot                --以每一对象的中心为中心
about coordsys             --以 coordsys 语句指定的坐标系原点为中心
about <node>              --以指定对象的中心点为中心
about <matrix3>            --以指定<matrix3>定义的坐标系原点为中心
about <Point3>             --以指定点为中心
```

例如：

```
--将所有 Box 对象绕 foo 的 y 轴旋转 30 度
about foo rotate Box 30 y_axis
--将所有 planets 对象绕它父对象坐标系的 z 轴旋转 45 度
in coordsys parent about coordsys
rotate $planets* 45 Z_axis
```

#### 4.10.6 undo

关联语句 `undo` 控制在脚本里对场景的改变是否可以取消，其格式为：

`undo <boolean>`

默认值为 `On`，表示在 MAXScript 里执行的对场景的改变是可以取消的。例如：

```
undo On
(
    delete $Box*
```

```

    delete $sphere*
    clearUndoBuffer()
)

```

如果在随后的操作中，用户用 3ds max 菜单里的 undo 命令或用 MAXScript 命令：

```
max undo
```

可以一次性全部取消 undo 关联语句里的全部操作。

下面是几个控制 undo 命令和场景存储状态的函数：

#### 1. ClearUndoBuffer()

清空 Undo 堆栈，重置 undo 状态。

#### 2. setSaveRequired <boolean>

设置 3ds max 的 dirty 标志，当用户执行 File | New 或 File | Reset 操作时，如果 dirty 标志为 True 或 Undo 堆栈为非空，系统会询问用户是否要存储场景。

#### 3. getSaveRequired()

如果 3ds max 的 dirty 标志为 True 或 undo 堆栈为非空时，返回 True。如果 undo 堆栈为非空，即使将 setSaveRequired 设为 False，getSaveRequired()也返回 True。

### 4.10.7 关联语句的迭加

用户可以把关联语句的前缀进行迭加，之间用逗号隔开，系统会按排列顺序对关联体里的语句进行关联操作，例如：

```

animate On, at time(t+1), with undo Off, cooridns local
(
    ...
)
pos0 = at time 35 $Box01.position
pos1 = at time 75 $Box01.position

```

后面两句把 Box01 对第 35 和 75 帧时的位置赋给变量 pos0 和 pos1。

### 4.10.8 关联语句嵌套

关联语句可以任意嵌套，甚至同一类型的关联语句可以在嵌套里覆盖上一层次的设置，例如：

```

with animate On, at time t
(
    move ...
    rotate ...
    animate Off
(
    move ...
    rotate ...
)
scale ...

```

```

    ...
)
```

#### 4.10.9 持续关联语句

用户可以把 coordsys、animate、time 等关联语句强制设置成“持续的”，该设置会在执行接下来的代码时保持有效，直到用户改变这些设置或用关联语句覆盖这些设置，这在 Listener 窗口里是非常有用的，打开一个关联语句之后可以一直执行相关的关联操作，而不用在每一个语句前加一个关联前缀，建立持续关联语句的语法为：

```
set <context>
```

其中<context>可以是 animate、time、in、about、coordsys、level 或 undo。例如：

```

set animate On
set time 30f
move $foo [80,0,0]
scale $baz [1,1,3]
$bar.bend.angle += 23
...
set animate Off
set time Off
```

由上例可以看出，有些关联语句允许使用变量，例如：

```
set time <value> | Off
set level <nodel>
```

还可以把 set 语句的结果赋给一个变量，然后在后面的程序中重新恢复该设置，例如：

```

oc = set coordsys parent      --存储旧坐标系
rotate $foo(quat 30 z.axis)
.....
set coordsys oc              --恢复设置
```

可以设置关联语句的状态设为#default 来将其恢复为默认值。可以指定#default 的关联语句有：animate、in、coordsys、level。

# 第5章 控制MAXScript程序流程

MAXScript语言有几个表达式是用来显式地控制程序的流程：

```
<if_expr>
<case_expr>
<do_loop>
<while_loop>
<for_loop>
<loop_exit>
<try_expr>
```

这些语句在许多其他编程语言中都可以找到，另外MAXScript还有几个隐式地控制程序流程的语句，例如：when、On等。

下面分别讨论以上各语句。

## 5.1 if 表达式

if表达式根据一个逻辑测试表达式的结果来决定执行的走向，其语法有以下两种：

```
if <expr1> then <expr2> [else <expr3>]
if <expr1> do <expr2>
```

其中<expr1>为测试表达式，其求值结果必须为True或False，如果测试结果为True，那么then或do之后的<expr2>被执行，其求值结果作为if表达式的返回值；如果测试结果为False，可选的else <expr3>被执行，其求值结果作为if表达式的返回值，如果没有else <expr>或使用do形式，if表达式的返回值为undefined。

do形式的if表达式是专为在Listener窗口里执行if表达式而设计的。如果使用if...then形式，而不想输入else <expr>，MAXScript将等着看用户是否会输入else <expr>，所以在输完then <expr>并回车后，系统并不会马上执行if表达式。而用do形式的if表达式就不会出现上面情况。

## 5.2 case 表达式

case表达式将一个测试表达式的结果和一系列标签表达式的标签进行比较，根据比较结果从中选择一个标签表达式进行求值，其语法为：

```
case [<expr>] of(<cases>)
```

其中<expr>是测试表达式，而<cases>是带标签的表达式序列，其语法为：

```
<factor>:<expr>
```

```
default : <expr>
```

例如：

```
new_obj = case copy_type.state of
(
    2 : copy $foo
    3 : instance $foo
    default: reference $foo
)
```

仅第一个和测试表达式匹配的标签后面的表达式被求值，所有标签必须和测试表达式的结果具有可比性，如果没有标签与测试表达式匹配，则对默认值标签后面的表达式进行求值；如果 case 表达式中没有指定默认值标签，则 case 表达式会返回 undefined。

如果用表达式作为标签，必须用括号把表达式括起来，例如：

```
case of
(
    (a > b) : print "a is big"
    (b < c) : print "b is little"
    (c <= d*3) : ...
    default: ...
)
```

### 5.3 while 循环和 do 循环

while 和 do 循环用来重复执行一个表达式，直到测试表达式的求值结果为 False。while 和 do 循环只有很小的不同，其语法为：

```
do <expr> while <expr>
```

```
while <expr> do <expr>
```

其中 while 后面的<expr>为测试表达式，如果测试表达式结果为 True，将一直重复执行 do 后面的<expr>。do 循环因为在循环结尾执行测试表达式，所以会至少执行一次循环，而 while 循环如果一开始测试表达式的结果为 False，那么根本不会进入循环。最后一次执行 do 后面<expr>的结果将作为循环语句的返回值，如果 while 循环一开始测试表达式结果就为 False，返回值为 undefined。例如：

```
while x > 0 do
(
    print x
    x -= 1
)
while not eof f do print(read_value f)
do
(
    exchanged=False
```

```

for i=1 to imax do
( j=i+gap
  if(compare array[j] array[i])do
    (swap array[j] array[i]; exchanged=True)
  )
)while exchanged

```

## 5.4 for 循环

for 循环语法为：

for <var\_name>(in | =)<sequence>(do | collect)<expr>

其中<var\_name> 为循环索引变量名，<sequence>为循环的源数值，可以为以下几种形式：

<expr> to <expr> [by <expr> ] [where <expr> ]

<expr> [where <expr> ]

例如：

```

for i = 1 to 10 do print i           --数字序列
for item in [able] do x = x+item.height --数组序列
for t in of to loof by 5f do         --以帧为单位的时间序列
bigones = for obj in $Box*
where obj.height > 100 collect obj      --将高度大于 100 的 Box 放到数组里

```

第一种<sequence>形式是标准的数字循环，其中第一个<expr> 值为循环初值，to <expr> 为循环终值，可选的 by <expr> 为每次循环的增量。允许的数值类型有整数、浮点数和时间，如果省略 by <expr> 循环增量为默认值 1。

第二种<sequence>形式中<expr> 为一个序列集合，如数组或选择集，循环将遍历集合里的每一个元素，并将元素的值赋给循环变量。MAXScript 允许用几种选择集序列，包括路径名、当前选择对象集合、节点的子对象集和对象的修改器堆栈。

两种<sequence>形式都有一个可选的 where <expr>，其求值结果必须为 True 或 False。where 表达式在执行循环之前被求值，只有当它的结果为 True 时，才接着执行循环。使用 do <expr> 形式的 for 循环简单地在每次循环时对<expr> 里的表达式依次求一遍值，循环变量<var\_name>在循环体<expr> 里是可见的。一个 do <expr> 形式的 for 循环如果作为表达式，其返回值总是为 ok。

collect <expr> 形式的 for 循环收集每次循环时<expr> 的求值结果，并把它们存储在一个数组里，然后这个数组被作为 for 循环的返回值。如果在 collect 循环里同时指定 where 表达式，仅收集那些符合条件的表达式求值结果。我们也可以使用一个特殊的值 DontCollect 来过滤那些不需要的循环结果。

正如在“变量作用域”一节中讲到的，一个 for 循环将会创建一个新的作用域。for 循环变量的作用域总是被限定在该新建的作用域里，即使之前同名的变量已经存在，一旦退出 for 循环，循环变量就不能被存取了。例如：

```

while x > 0 do
(  print x
  x -= 1
)
while not eof f do print(read_value f)
do
(  exchanged=False
  for i=1 to imax do
(  j=i+gap
    if(compare array[j] array[i])do
      (swap array[j] array[i]; exchanged=True)
    )
)while exchanged

```

## 5.5 continue语句

MAXScript设计了一个continue语句，用来在for、do、while循环中直接跳至循环末尾，然后开始下一次循环，如下例：

```

for i=1 to 8 do(if i == 5 do continue; print i)--打印 1..4, 6..8
while not eof f do                         --读入文件直至文件末尾
(  local line=readline f                  --读入一行
  if line[1] == "-" do continue           --如果为注释行，跳过本行
  line1=parser1 line                      --调用函数 parser1
  processobjs line1                      --调用函数 processobjs
)

```

如果在一个for...collect循环里执行continue语句，本次循环的表达式结果将不被收集到数组中，例如：

```
for i = 1 to 8 collect(if i == 5 do continue; i)
```

## 5.6 exit语句

MAXScript设计了一个exit语句在for、do、while循环中提前退出循环，即使循环测试表达式的值仍为True。其语法为：

exit [with <expr> ]

例如：

```

while x < y do
(  local delta = x - y
  if delta <= 0 then exit
  $foo.pos.x = compute_x(foo / delta)
  x += 0.1
)

```

其中可选的 with <expr> 让用户指定提前退出循环时循环体的返回值，如果未指定 with <expr>，返回值为 undefined。

从一个 for...do 循环用 exit with <expr> 退出，返回值为 ok。

从一个 for...collect 循环用 exit with <expr> 退出，返回值为退出前收集的元素组成的数组。

## 5.7 try 表达式

MAXScript 提供了一种异常处理机制：try 表达式。try 表达式把一段代码放在一对括号里，并可以捕捉该代码段里任何运行错误，这样当错误发生时，可以采取相应的反应，而不是由 MAXScript 中止程序的运行并打印出错误信息。try 表达式的语法为：

```
try <protected_expr> catch <on_err_expr>
```

其中<protected\_expr>为要捕捉运行异常的表达式，一旦发现运行错误，<on\_err\_expr>将被执行。如果<protected\_expr>为一个块表达式，执行将停在发生错误的那一行，如果没有发生运行错误，<on\_err\_expr>将不被执行。例如：

```
f = openFile "foo.dat"
try
    while not eof f do read_some_data f
catch
(   messageBox "bad data in foo.dat"
    results = undefined
)
close f
```

上例中当调用函数 read\_some\_data()过程中发现的任何错误将被捕捉到，然后系统会显示一个信息框。

# 第6章 MAXScript自定义函数

## 6.1 创建自定义函数

MAXScript语言系统函数功能非常强大，可以用来创建对象、让用户读取、设置对象的属性。在3ds max里的脚本语言里，还允许用户定义自己的函数。

函数创建语法为：

```
<function_def> ::= [mapped](function / fn)<name> {<parameter>} = <expr>
```

其中<name>为函数名，实际是一个变量名，它的作用域为当前MAXScript的作用域。

可选的<parameters>为函数的参数可以为以下几种：

- ◆ <name> 位置型参数
- ◆ <name>:[<operand>] 关键字型参数，<operand>为参数默认值
- ◆ “=” 后面的<expr> 为函数定义体，它包含一系列的表达式

在MAXScript中，可以用函数来执行几乎任何操作。让我们从一个简单的例子开始。如图6-1所示，在Listener窗口里输入如图6-1所示的内容。

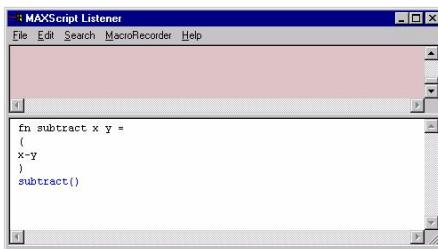


图6-1 自定义函数

MAXScript会返回subtract()，表明它已定义了一个函数：subtract。函数的定义以关键字function或fn开始，后面紧跟着函数名，然后是函数定义需要的参数序列，“=”号表示函数定义的开始，函数定义语句必须用一对括号括起来，即使函数定义语句只有一行。

**注意** 除非另有指定，在函数定义过程中创建的变量都是局部变量。一旦函数被定义，就可以在以后的脚本里调用它。因此最好把函数定义放在脚本的开始处。可以像调用系统内嵌的函数一样调用自定义函数。例如：

```
subtract 4 2
```

在上例中，我们使用的参数称为位置型参数（Positional Argument），当我们调用带位置型参数的函数时，必须按定义函数时的顺序全部给出。另外一种参数形式称为关键字型

参数 (Keyword Argument)，这种参数在所有的对象构造函数里都有用到。下面是一个包含一个关键字型参数的自定义函数：

```
function sign val : 0 =
(
if val == 0
then messageBox("Equal to 0") --等于 0
else if val > 0
then messageBox("Bigger than 0") --大于 0
else messageBox("Less than 0") --小于 0
)
```

其中 `messageBox()` 函数用来把括号中的信息显示在一个信息窗口里。上例中参数 `val` 就是关键字型参数，在冒号后面的值称为默认值。当调用包含关键字参数的函数时，这些参数是可选的，如上例，我们用 `sign()` 调用函数，而不指定参数的值，MAXScript 就使用它的默认值，给出下面如图 6-2 所示的信息窗口。

而按下面形式调用函数：

```
sign val :-5
```

将得到系统给出的反应，如图 6-3 所示。



图 6-2 `messageBox` 函数默认值信息窗口



图 6-3 `messageBox` 函数指定参数信息窗口

在函数定义中，我们可以包含任意数量的关键字型参数，在调用函数时，这些关键字型参数的调用顺序可以随意排列。这一点在对象创建等函数中非常有用，这些函数中有时会有 20~30 个相关的参数，如果我们使用位置型参数来定义这些函数，必须在创建对象时按正确的顺序提供所有的参数，那样将是非常麻烦的事。也可以同时使用位置型参数和关键字型参数，这时，位置型参数必须放在所有关键字型参数之前，关键字参数在变量名后有一个冒号 (“：“），并跟一个可选的默认值，调用函数时可以可选地提供关键字型参数，并可以按任意顺序排列。如果调用者没有提供关键字参数的调用值，它们将使用函数定义中指定的默认值，如果在定义中未指定默认值，系统将给它们赋值 `undefined`。如下例：

```
function mycube side position : [0, 0, 0] =
(
Box length : side width :side height :side pos :position
)
```

在调用上面的函数时，必须提供参数 `side`，而参数 `position` 是可选的。

在函数定义中使用前缀 `mapped` 指定该函数将对一个集合进行操作，这意味着如果函数调用的第一个参数为一个数据集合，函数将自动将集合中每一元素作为参数重复调用函数。例如：

```

function add a b = a + b
fn factorial n = if n <= 0 then 1 else n * factorial(n - 1)
mapped function rand_color x =
    x.wireColor = random(color 0 0 0)(color 255 255 255)
fn starfield count extent:[200,200,200] pos:[0,0,0] =
(
    local f = pos - extent / 2,
    t = pos + extent / 2
    for i = 1 to count do
        sphere name: "star"           \
        radius:(random 0.1 2.0)      \
        position:(random f t)        \
        segs:4 smooth:False
)

```

脚本函数将`<expr>`的值作为返回值，如果`<expr>`是一个块表达式，块表达式的最后一个表达式的值作为函数的返回值，函数可以回归调用（也即自己调用自己），如上例中的`factorial`函数。

如果想定位一个脚本函数的源脚本文件，可以用`showSource()`函数，语法为：

```
showSource <fn>
```

这样可以打开一个`Script Editor`窗口，在窗口里显示包含定义函数的脚本文件，并将鼠标定位在函数定义的开始。

## 6.2 函数变量

当用户定义一个函数时，在MAXScript内部依次发生以下事件：

- ◆ 函数定义表达式`<expr>`被求值，创建一个函数值。
- ◆ 函数`<name>`作为一个新变量被声明。
- ◆ 函数值被赋给新变量。
- ◆ 函数值作为函数定义表达式的值返回。

因为函数本身作为值存储在变量里，用户不仅可以通过函数名变量来调用函数，而且可以简单地通过引用函数变量将函数值赋给另一个变量或数组的某一元素，或作为变元传递给另一个函数。例如：

```

fn saddle x y = sin x * sin y
fn sombrero x y = cos(x^2 + y^2)/(1 +(x^2 + y^2))
print(saddle 1 1)
surface_functions[1] = saddle --将函数存储在数组里
surface_functions[2] = sombrero
z =(surface_functions[i])10 10 --从数组里调用函数
build_math_Mesh sombrero --作为变元传递给另一个函数

```

函数定义内部的变量为局部作用域，其作用域为当前MAXScript范围，除非该变量已

在上一层作用域被定义或将它定义成全局变量，而函数调用参数总是为局部作用域。

如果一个函数变量被赋给另一个变量，对函数值的引用将被存储在新变量里，这样当函数变量的作用域已经结束时，如果新变量的作用域超出了当前的 MAXScript 范围，用户仍然可以通过该变量引用到函数。例如：

```
fn funcA which =
( if which == 1 then
    fn saddle x y = sin x * sin y
  else
    fn sombrero x y = cos(x^2 + y^2)/(1 +(x^2 + y^2))
)
myfunc=funcA 2
z=myfunc 10 10
```

上例中函数变量 saddle 和 sombrero 的作用域仅在函数定义 funcA 里，函数 funcA 的返回值为函数值 saddle() 或 sombrero()，被赋给变量 myfunc，这样我们可以在函数范围 funcA 之外调用函数 saddle() 或 sombrero()。

如上例所示，我们可以在任何表达式内定义函数。

### 6.3 函数参数

函数参数是在函数定义开始的地方隐式声明的局部变量，在函数调用时系统用调用参数值进行初始化，函数参数的作用域到函数定义体<expr>末尾，它们可以被重新赋值或被嵌套的同名局部变量屏蔽。

如前面所述，MAXScript 使用一种叫“引用赋值”的赋值机制。在函数中，每一个函数的参数都包含一个指向调用者传来参数值的引用，给参数赋一个新值会将对新值的引用放在参数里，而不会影响调用值。但是如果调用值为一个复合值如三维点、数组，在函数定义体中仅将复合值的某一成员赋新值，因为参数的引用没有改变，调用值会一起改变，如下例：

```
fn getXYZset val =
( val.x=random -100 100
  val.y=random 100 100
  val.z=random val.x val.y
)
( v1=[0,0,0]
  v4=v1
  getXYZSet v1
  format "v1= %; v4= %\n" v1 v4
  getXYZSet v4
  format "v1= %; v4= %\n" v1 v4
)
```

输出：

```

getXYZset()
v1= [-84,100,78.1224]; v4= [-84,100,78.1224] --第 9 行输出
v1= [10,100,18.6575]; v4= [10,100,18.6575] --第 11 行输出
OK --第 6 到 12 行结果

```

上例中可以看到变量 v1 和 v4 好像都被函数 getXYZset 同时改变。而实际上函数调用中仅传递了一个参数，之所以出现这样的原因是在第 6 行中，一个指向 Point3 类型值[0, 0, 0]的引用被创建并存储在变量 v1 里，在第 7 行里这个引用又被存储在 v4 里，这样 v1 和 v4 实际上指向同一个内存值。当在函数 getXYZset 里改变复合值的成员时，指向 Point3 类型值的引用没有改变，v1 和 v4 仍然引用同一个值，所以对任一变量成员的改变都会导致另一变量的相同成员的改变。

从一个函数返回多个值的正确方法应该像下例中一样，另外构造一个数组或结构，而不是如上例一样直接改变数组中的成员值。

```

fn readDataFile filename =
( f=openfile filename
  data=#()
  avg=0
  nVals=0
  while not eof f do
    ( val = readvalue f
      append data val
      avg += val
      nVals += 1
    )
  close f
  #(data,(avg/nVals))
)
result=readDataFile "c:\\datafiles\\run1.dat"
data=result[1]
avg=result[2]

```

当用户要将一个复合值作为参数传递给函数时，应该用 copy 方法将复合值复制一份，这样可避免引用调用参数的改变。如下例：

```

fn uppercase instring =
( local upper, lower, outstring
  upper="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
  lower="abcdefghijklmnopqrstuvwxyz"
  outstring=copy instring
  for i=1 to outstring.count do
    ( j=findString lower outstring[i]
      if j != undefined do outstring[i]=upper[j]
    )
  return outstring
)

```

## 6.4 return 表达式

和循环语句一样，自定义函数定义也可以提前退出，并且返回一个值，其语法为：

`return <expr>`

如在运行一个函数时遇到 `return` 语句，函数会马上退出，并返回`<expr>`指定的值。

如果 `return` 语句用在一个有 `mapped` 前缀的函数中，而第一个调用参数又是一个数据集合，`return` 语句的返回值为 `OK` 而不是`<expr>`指定的值。例如：

```
fn find_root_twod_fn =
(
    local root, last_root
    while True do
    (
        ...
        if abs(root - last_root) < epsilon then return root
        ...
    )
)
```

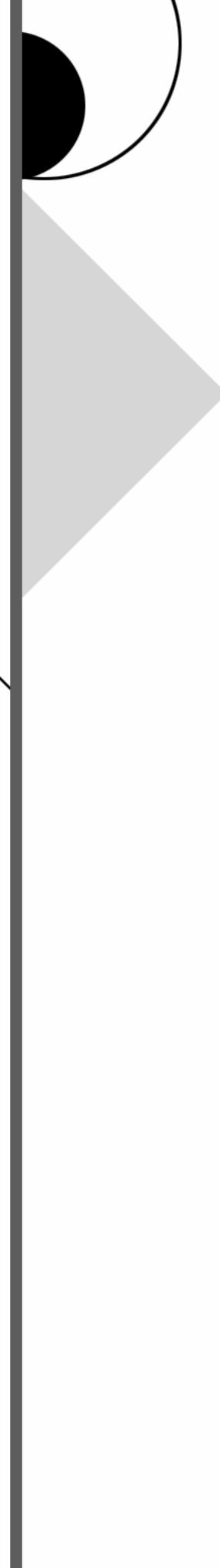
## 6.5 函数调用的优先级

一个函数调用优先级比操作数低，而比数学表达式、比较表达式、逻辑表达式要高，这意味着在将函数的变量用括号括起来时必须非常小心。例如：`sin x+1` 的求值顺序为`(sin x)+1`，一般而言，必须把形式为表达式的变量用括号括起来，例如：

```
sin(x+1)
atan2(y-1)(z-1)
```

但对于负数符号，可以用下面的形式，而不用括号：

```
sin -x
```



## 第 2 部 分

# 用 MAXScript 操作场景对象

# 第7章 对象超级类 MAXWrapper

MAXWrapper 类是 MAXScript 里所有 3ds max 对象类的超级类, 如: Node(场景对象)、Modifier(修改器)、Material(材质)等。一个 MAXWrapper 值包含对与之相连的 3ds max 对象的引用, 这样当一个 3ds max 对象被转换、删除或属性被修改时, MAXScript 会马上知道。

## 7.1 MAXWrapper 的通用属性和方法

### 7.1.1 MAXWrapper 值和类的通用属性

1. <MAXWrapperobject>.category: Name, 只读

<maxclass>.category: Name, 只读

<maxsuperclass>.categories: Array, 只读

.category 属性返回一个包含指定对象所属类别的 Name 值, 它一般是 Create 面板顶部下拉列表中的一个, 如: #Standard\_Primitives、#Compound\_Objects、#Particles\_Only 等。

对修改器而言, .category 属性决定了修改器显示在 Configure Button Sets | Modifier 列表的哪一组, 如: #Max\_Standard、#Max\_Edit、#Max\_Surface 等。

对纹理而言, .category 属性决定了纹理显示在 Material | Map Browser 对话框的哪一组, 如: #2D(2 维贴图)、#3D(3 维贴图)、#Comp(合成材质)等。

.categories 属性适用于 3ds max 的所有超级类, 如 Shape、GeometryClass、Helper、SpaceWarpObject、TextureMap、Modifier 等, 返回值为一个数组, 表示指定超级类可用的类别列表。如:

```
$line01.category      --返回#Splines
Gegon.category        --返回#Extended_Primitives
Shape.categories      --返回#Shape、#Splines、#Nurbs_curves
```

2. <MAXWrapperobject>.classID

<maxclass>.classID

.classID 属性返回 MAXWrapper 类或对象在 3ds max 内部的类码, 返回值为一个包含两个数字的数组: PartA 和 PartB。如:

```
Box.classID           --返回#(16, 0)
b=Box()
b.classID            --返回#(16, 0)
```

PartA 和 PartB 的组合是为了确保类 ID 码的惟一性。

3. <MAXWrapperobject>.superclassID

<maxclass>.superclassID

.superclassID 属性返回指定 MAXWrapper 类或对象在 3ds max 里的超级类类码。

4. <maxclass>.creatable: Boolean, 只读

如果返回 True, 表示该类可在 MAXScript 里被创建。

5. <maxclass>.ispb2based: Boolean, 只读

如果返回 True, 表示该类被定义为 ClassDesc2 类; 否则为 ClassDesc 类。ClassDesc2 使用 ParamBlock2, 而 ClassDesc 使用 ParamBlock (3ds max 3 以前版本)。

6. <maxclass>.localizedName, 只读

返回类的本地类名。

### 7.1.2 MAXWrapper 值和类的通用方法

1. copy <MAXWrapper\_object>

本方法可在 MAXScript 里用于所有 3ds max 场景对象 (Scene Object)、控制器 (Controller)、修改器 (Modifier)、材质 (Material) 等。可以用本方法为源对象创建一个副本。比如下面脚本会让对象 foo 与对象 baz 共享一个 Bend 修改器:

```
addModifier $foo $baz.bend
```

而下面的脚本会先给对象 baz 的 Bend 修改器创建一个独立的副本, 再将这个副本赋给对象 foo。这样如果修改对象 foo 的修改器, 就不会影响对象 baz 里的修改器了, 反之亦然。

```
addModifier $foo(copy $baz.bend)
```

在下面的例子中, 我们让几个对象共享一个 Rotation\_Controller:

```
c = bezier_rotation()
$foo.rotation.Controller = c
$baz.rotation.Controller = c
$bar.rotation.Controller = c
```

如果在后面的脚本里要让某一控制器独立, 可以用下面的脚本:

```
$baz.rotation.Controller = copy $baz.rotation.Controller
```

如果参数<MAXWrapper\_object>指定的是一个场景对象, 系统会创建一个新的场景对象, 新建对象在 3ds max 视窗里可见, 且可以被用户存取; 如果参数<MAXWrapper\_object>指定的不是一个场景对象, 复制仅在 MAXScript 内存里创建。

2. exprForMAXObject <MAXWrapper\_object>

返回一个 String 值, 表示如何用属性存取器或索引方式来命名指定 MAXWrapper 对象。

例如:

```
m=$foo.bend
exprForMAXObject m --返回字符串: "$foo.bend"
```

本方法的返回值与 Listener 窗口的 MacroRecorder 菜单项的设置有关, 请参见下例:

```
b=Box()      --返回$Box:Box02 @ [0.000000,0.000000,0.000000]
c=b.pos.Controller          --返回 Controller:Bezier_Position
exprformaxobject c          --返回 "$Box02.pos.Controller"
bm=bend()                  --返回 Bend:Bend
addModifier b bm            --返回 OK
exprformaxobject bm          --返回 "$Box02.Modifiers[#Bend]"
```

在上面例子中 MacroRecorder 菜单项的选项为 Explicit scene object names, 如果选择 Selection-relative scene object names, 最后一句会改为:

```
exprformaxobject bm          --返回 "$.Modifiers[#Bend]"
```

### 3. createInstance <maxclass> [keyarg1:v] [keyarg2:v] ...

本方法主要用来创建一个临时的 Geometry 类基本对象, 并在后面的 Geometry 脚本插件里把面片 (Mesh) 信息从源对象提取出来。可选参数<keyargs>为对象类构造函数里的关键字型参数, 一般为基本对象的参数。例如:

```
b = createInstance Box length:10 height:40 width:20
```

将创建一个 Box 类实例。该 Box 对象不会显示在 3ds max 视窗里, 因为它并不是一个场景对象。它仅包含与 Box 类对象相关的几何体。但可以用其.Mesh 属性 (返回一个 TriMesh 值) 获取对应的 Mesh 对象。

### 4. replaceInstances <old\_MAXWrapper> <new\_MAXWrapper>

用新对象替换场景里所有旧对象。新旧对象必须具有同样的.superclass 属性。

### 5. isValidObj <value>

如果指定值<value>指向一个 MAX 对象, 且该对象未被删除, 返回 True。

### 6. getClassName <MAXWrapper>

将 MAXWrapper 类名作为一个字符串返回。

在 3ds max 里一个重要的特征就是 3ds max 对象之间互相关联, 比如 Material 类对象依赖于它的各种贴图; Path 类控制器依赖于它的 Percent\_Controller; 一个场景对象依赖于它的基本对象等。下面的几个方法与对象互相关联有关。

### 7. refs.dependencyLoopTest <MAXWrapper\_object> <MAXWrapper\_object>

如果指定的两个 MAXWrapper 对象会创建一个关联循环, 则返回 True。

例如:

```
a = Box pos:[-100,0,0]
$Box:Box01 @ [-100.000000,0.000000,0.000000]
b = Box pos:[100,0,0]
$Box:Box02 @ [100.000000,0.000000,0.000000]
a.material = standardMaterial()
Standardmaterial:Standard
refs.dependencyLoopTest a b --返回 False
refs.dependencyLoopTest a a.material --返回 True
```

### 8. refs.dependents <MAXWrapper\_object>

返回一个 Name 类数组，表示依赖于指定 MAXWrapper 对象的所有 MAXWrapper 对象类名。指定的参数<MAXWrapper\_object>可以为任何 MAXWrapper 对象，比如 Node（场景对象）、Modifier（修改器）、Material（材质）、Controller（控制器）等。如：

```
refs.dependents $foo.Material.diffuseMap
```

将返回：

```
#{Material_#3, Material_#2, Map_#2:Noise, Material_#1}
```

表示 foo 对象里的 diffuseMap 贴图被材质 Material\_#1、Material\_#2、Material\_#3 和噪波贴图 Map\_#2:Noise 引用。

下面的范例先创建几个对象，然后为这些对象设置了控制器，而这些控制器又依赖于其他的对象：

```
theSphere=sphere() --创建 sphere 对象
theCone=cone radius1:0 radius2:20 --创建 cone 对象
theHelix=helix height:100 pos:[100,100,0] --创建 helix 对象
theCone.target=theSphere
--将 theSphere 作为 theCone 的目标对象，一个 lookat_Controller 会自动赋
--给 theCone 对象
theSphere.position.Controller=path path:theHelix
--将一个 path_Controller 赋给 theSphere 对象，其路径为 helix 对象
refs.dependents theSphere --显示依赖于 sphere 对象的对象
refs.dependents theCone --显示依赖于 cone 对象的对象
refs.dependents theHelix --显示依赖于 helix 对象的对象
```

输出：

```
$Sphere:Sphere02 @ [0,0,0] --第 1 行结果
$Cone:Cone02 @ [0,0,0] --第 2 行结果
$Helix:Helix02 @ [100,100,0] --第 3 行结果
$Sphere:Sphere02 @ [0,0,0] --第 4 行结果
Controller:Path --第 7 行结果
#{Controller:Look_At, $Cone:Cone02 @ [0,0,0]} --第 9 行结果
#() --第 10 行结果
--下面是第 11 行的输出
#{Controller:Path,
Controller:Position_Rotation_Scale,
$Sphere:Sphere02 @ [0,0,0],
Controller:Look_At, $Cone:Cone02 @ [0,0,0]}
```

## 7.2 MAXWrapper 的扩展数据

MAXWrapper 的扩展数据提供了将任意数据附加到 3ds max 对象（如 Node、Material、Modifier、Controller 等）的方法，这些扩展数据会与场景对象一起存储在 3ds max 场景文件中。

下面的几个方法用来存取、设置 MAXWrapper 对象的扩展数据。

## 1. setAppData &lt;MAXWrapper\_object&gt; &lt;Integer\_ID&gt; &lt;string&gt;

给指定的 3ds max 对象的指定扩展数据<Integer\_ID>赋新值<string>。如果此前该扩展数据尚未建立，系统会先创建一个新的扩展数据项；如果该 ID 码已存在，用参数<string>指定的内容替代旧值。

## 2. getAppData &lt;MAXWrapper\_object&gt; &lt;Integer\_ID&gt;

返回一个 String 值，表示 3ds max 指定对象的指定扩展数据<Integer\_ID>的内容。如果此前该扩展数据尚未建立，返回 undefined。

## 3. deleteAppData &lt;MAXWrapper\_object&gt; &lt;Integer\_ID&gt;

删除指定 ID 码的扩展数据。

## 4. clearAllAppData &lt;MAXWrapper\_object&gt;

清除指定 MAXWrapper 对象所有与 MAXScript 相关的扩展数据。

可以向对象加入任意数量的扩展数据字符串，在第一次加入时，必须指定一个整型的标识 ID 码，然后就可以用这个 ID 码存取扩展数据了。

**注意**

扩展数据只有当它的对象存在于场景中时才会被存储在场景文件中。比如：如果用 MAXScript 创建一个材质但并未将它赋给任何对象或材质编辑器，该材质及其扩展数据将不会被存储在场景文件里。

如果想将扩展数据添加到整个场景，建议按以下办法：创建一个隐藏的 Dummy 类对象，并给它一个惟一的名字，然后将扩展数据添加到该对象。

下面的例题把一个数组里的数据存在一个场景对象里：

```
ss = StringStream ""
for v in data_Value do print v to:ss
setAppData $foo 1 ss
```

当存储场景后重新装入时，可重新读入：

```
ss = StringStream(getAppData $foo 1)
data_Value = #()
while not eof ss do
    append data_Value(readValue ss)
```

# 第8章 创建Node（节点）对象

Node类代表所有3ds max场景对象如Geometry、Camera、Shape等。每一种3ds max场景对象类都是由Node类派生而来。

## 8.1 Node类构造函数

1. <non\_wild\_card\_pathname>

不使用匹配符的路径名，代表惟一的场景对象。

2. <objectset>[<Integer>]

<wild\_card\_pathname>[<Integer>]

集合中指定索引位置的对象。

3. <Node\_constructor> [ name: <string> ] [ prefix: <string> ] \

[ material: <material> ] [ target: <node> ] \

[ pos: <point3> ] [ position: <point3> ] \

[ rotation: <quat> ] [ scale: <point3> ] \

[ pivot: <point3> ] [ transform: <matrix3> ] \

[ isSelected: <boolean> ] [ dir: <point3> ]

<Node\_constructor>为本书8.4节“Node子类”中列出的任意一种场景对象类，如Box、Sphere、QuadPatch、Spline、Shape等，例如：

```
b=Box name: "foo" pos:[10,10,10] height: 20
```

所有的Node构造函数都可以指定上面的可选参数，其后跟着Node类的通用属性或某子类的特定参数。

参数说明：

- ◆ name 为可选参数，用来命名新创建的对象，3ds max允许将一个或几个场景对象指定同一个.name属性。

- ◆ prefix 为可选参数，用来取代.name属性，它用来指定该对象.name属性的开始几个字符，3ds max会自动在后面加上一系列数字来产生一个惟一的.name属性。如：

```
for i in 1 to 100 do sphere prefix: "baz"
```

将创建100个Sphere类对象，每个Sphere对象的.name属性均以baz开始，如baz01、baz02、baz03等。

- ◆ material 可选参数，为创建对象指定一种材质。

- ◆ target 可选参数，用来自动给对象的 transform 轨迹加上一个 LookAt\_Controller，并将指定的对象作为目标对象。
- ◆ pos 指定新建对象的位置，默认值为[0,0,0]。
- ◆ position 与.pos 属性一样。
- ◆ rotation 指定新建对象的转角，默认值为 0。
- ◆ scale 指定新建对象的缩放比例，默认值为 100%。
- ◆ pivot 指定新建对象的支点位置，默认值为正常对象支点位置。
- ◆ transform 指定新建对象的转换矩阵，默认值为单位矩阵。
- ◆ isSelected 指定新建对象的被创建后是否被选择，默认值为 False。
- ◆ dir 设置新建对象 Local 坐标系的 Z 轴方向。

转换类参数（如 pos、scale、rotation 和 pivot 等）按其在构造函数里的排列顺序施加给新建对象，这样可以控制转换顺序。参数 pos、scale、rotation 和参数 transform 是互相排斥的，用参数 transform 指定的转换矩阵同样可以完成参数 pos、scale、rotation 指定的转换。

**注意** 如果指定了 target 参数，3ds max 会把最后一次指定的目标对象和源对象存储在一起，如果删除源对象，由.target 属性指定的目标对象也会同时被删除。

## 8.2 Node 类方法

### 8.2.1 通用方法

**注意** 本节中讲述的大多数方法都会自动对对象集合进行映射操作，这些方法被标记为 mapped。

#### 1. IsValidNode <var>

如果参数<var>为一个 Node 值，且该对象没有被从场景里删除，返回 True；否则，返回 False。

#### 2. move <node> <point3> mapped, 移动对象

scale <node> <point3> mapped, 缩放对象

rotate <node> <angle> <axis\_point3> mapped, 旋转对象，角度单位为度

rotate <node> <quat> mapped

rotate <node> <eulerangles> mapped

转换运算，所有这些操作都是在当前工作坐标系下进行。

#### 3. copy <node> mapped

reference <node> mapped

instance <node> mapped

三种对象复制方法，在这些方法的结尾都可以跟着任何相应用对象的创建类可选参数，系统会在复制完成后再将复制对象的这些参数进行修改。三种复制方法的区别为：

- ◆ **copy** (复制)方法 根据源对象创建一个独立的副本对象。
- ◆ **reference** (引用)方法 当对源对象进行调整时,会影响到引用对象,但若对引用对象进行调整时,源对象不会受到影响。
- ◆ **instance** (实例)方法 当对源对象或实例对象进行调整时,都会引起另一对象的相同调整。

这三种复制方法都会同时复制源对象的.transform 属性控制器 (Controller) 及其关键帧、.visibility 属性控制器及其关键帧 (如果该对象被赋给一个 Visibility 轨迹)。

#### 4. snapshot <node> mapped

本方法和 3ds max 的 SnapShot 工具按钮的功能类似。本方法生成一个新对象,其包含了源对象<node>在快照调用时的 Mesh 信息的副本。在新对象里,所有源对象现有的 Modifier 被塌陷,所有 SpaceWarp 被施加到对象上。和对象复制方法 (copy、reference 和 instance)一样,可以在方法的结尾加上任何源对象对应的创建类可选参数,如 pos:、name: 等。下面是一个例子:

```
for t in 0 to 100 do at time t snapshot $foo
```

#### 5. delete <node> mapped

删除指定对象或对象集。

#### 6. instanceReplace <dest\_node> <src\_node> mapped

referenceReplace <dest\_node> <src\_node> mapped

将已有的对象变为其他对象的实例或引用。用这个方法可以将指定对象用别的对象代替。

参数<dest\_node>指定的对象会变成参数<src\_node>指定对象的实例或引用。如果为 instanceReplace 方法,原有的 Geometry 和 Modifier 信息会被清除,并被<src\_node>对象的相应信息取代,但源对象的 Node 级属性 (如.material、.transform、.visibility 和.name) 会被保留;如果为 referenceReplace 方法,<dest\_node>对象会完全变成<src\_node>对象的副本,对<src\_node>对象的调整会影响到<dest\_node>对象,但<dest\_node>对象的调整不会影响到<src\_node>对象。例如:

```
instanceReplace $foo* $baz
```

会将所有 foo\*对象变成 baz 对象的实例。

#### 7. attachObjects <node1> <node2> [ move:<boolean> ]

将<node2>对象变成<node1>对象的子对象。除非指定参数 move:False, <node2>对象的位置会变为和<node1>对象的位置一致。

#### 8. getTMController <node>

返回指定对象的属性.transform.controller。

#### 9. bindSpaceWarp <node> <spacewarp\_node>

将指定场景对象和指定 SpaceWarp 对象绑定。SpaceWarp 对象显示在修改器堆栈的顶部,可以和 Modifier 一样进行存取。

#### 10. getPolygonCount <node>

返回一个二元数组。第一个元素表示将对象转换为 Mesh 对象时，新对象包含的面（Face）数目，第二个元素表示新对象包含的顶点（Vertex）数目。

11. **isShapeObject <node>**

如果指定对象为 Shape 类，返回 True；否则，返回 False。

12. **numSurfaces <node>**

返回对象的曲面（Surface）数目。

13. **isSurfaceUVClosed <node> <surface\_index\_Integer>**

返回一个二元数组，分别表示指定对象的指定序号的面在 U 和 V 方向是否闭合。



**注意** 不是所有对象都会执行本方法，但是它们会返回一个默认值# (True, True)。

14. **getTransformAxis <node> <index>**

返回一个 Matrix3 类值，表示指定对象的指定序号的转换轴。一般来说，一个对象仅有一个转换轴，但在某些情况下，也会有多个转换轴存在。如果遇到指定参数<index>大于对象转换轴数目，返回第一个转换轴。由本方法返回的值可以用在一个 in coordsys 关联语句和 about 关联语句里，用来将对象绕指定坐标系中心进行 Move、Rotate 和 Scale 操作。如：

```
fn axisRotate obj rotation =
(local axis
local objA = if classof obj == objectSet or classof obj == array then obj
else #(obj)
if getCoordCenter() != #local then --if 开始
(axis = getTransformAxis objA[1] 0
for obj1 in objA do in coordsys axis about axis rotate obj1 rotation
)
else
(for obj1 in objA do
(axis = getTransformAxis obj1 0
in coordsys axis about axis rotate obj1 rotation
)--for 结束
)--if 结束
)--fn 结束
```

15. **invalidateTM <node>**

将对象转换矩阵所占的缓存回收。

16. **invalidateTreeTM <node>**

将对象转换矩阵所占的缓存回收，并将转换矩阵改变的消息通知对象的子对象。

下面的脚本函数可用来强制刷新那些 Transform 控制器类型为 Scripted\_controller 的对象：

```
mapped fn TMInvalidate obj =
(at time(currenttime-1)obj.transform
nodeInvalRect obj
invalidateTreeTM obj
redrawViews()
```

- )
17. invalidateWS <node>  
使对象的World坐标系无效。
  18. getNodeByName <string> exact:<boolean>  
返回第一个.name属性为指定字符串的对象。除非指定参数exact:True，指定字符不区分大小写。
  19. snapshotAsMesh <node>  
返回一个Mesh对象，表示指定对象此时的一个快照。
  20. getInheritanceFlags <node>  
 setInheritanceFlags <node>(#all|#none | <bitarray>)keepPos:<boolean>  
 存取一个BitArray值，表示指定对象的继承标记。如果某一位为On，表示与之对应的继承被打开。BitArray值各位的顺序为：  
 #{POS\_X,POS\_Y,POS\_Z,ROT\_X,ROT\_Y,ROT\_Z,SCALE\_X,SCALE\_Y,SCALE\_Z}  
 如果指定参数keepPos:False，当继承位被打开或关闭时，对象可能会被移动。
  21. getTransformLockFlags <node>  
 setTransformLockFlags <node>(#all | #none | <bitarray>)  
 存取一个BitArray值，表示指定对象的转换锁标记。如果某一位为On，表示与之对应的转换锁被打开。BitArray值里各位的顺序为：  
 #{POS\_X,POS\_Y,POS\_Z,ROT\_X,ROT\_Y,ROT\_Z,SCALE\_X,SCALE\_Y,SCALE\_Z}

## 8.2.2 与渲染有关的方法

1. getInheritVisibility <node>  
如果指定对象继承了父对象的.visibility属性，返回True；否则，返回False。
  2. setInheritVisibility <node> <boolean>  
设置指定对象是否继承父对象的.visibility属性。
  3. getVisController <node>  
返回指定对象的可见性控制器(.visibility.controller)的一个副本。如果对象的.visibility属性没有设置动画，返回undefined。
  4. getImageBlurMultController <node>  
返回对象的Image\_Motion\_Bur\_Multiplier控制器。如果对象没有该控制器，返回undefined。
  5. setImageBlurMultiplier <node> <time> <number>  
在指定时间为对象设置Image\_Motion\_Bur\_Multiplier控制器的值。
  6. setImageBlurMultController <node> <controller>  
将对象的Image\_Motion\_Bur\_Multiplier控制器设置为指定<controller>。
  7. setMotBlur <node> <Integer>  
为对象设置运动模糊类型：
- ◆ 无运动模糊

- ◆ 对象运动模糊
- ◆ 图像运动模糊

## 8. setRenderable &lt;node&gt; &lt;boolean&gt;

设置是否渲染指定对象。

## 9. getRenderID &lt;node&gt;

返回对象的.renderID 属性。如果场景还没有被渲染过或指定对象不能被渲染，返回整数 65535。

## 10. setRenderID &lt;node&gt; &lt;Integer&gt;

设置对象的.renderID 属性。该值不会和场景对象存储在一起，在下一次渲染时会改变。

## 8.2.3 与组 (Group) 有关的方法

## 1. group &lt;node&gt; [ name:&lt;string&gt; ] [ prefix:&lt;string&gt; ] \

[ select:<Boolean> ] mapped

将指定对象设置为一个组。可以指定组名或组名前缀，为了确保组名的惟一性，建议最好不指定组名或组名前缀。如果指定参数 select:True，组在形成以后会被选择。例如：

--将所有名称以 Box 开头的对象组成一个组，组名为 Boxes

group \$Box\* name: "Boxes"

Group selection --将当前被选择的对象组成一个组

## 2. ungroup &lt;group\_head\_node&gt; mapped

取消一级组对象。如：

ungroup \$group01

## 3. explodeGroup &lt;group\_head\_node&gt; mapped

取消组对象里所有级别的组。

## 4. isGroupHead &lt;node&gt;

如果指定对象为组头，返回 True；否则，返回 False。

## 5. isGroupMember &lt;node&gt;

如果指定对象为一个组成员，返回 True；否则，返回 False。

## 6. isOpenGroupMember &lt;node&gt;

如果指定对象为一个组成员，且该组被打开，返回 True；否则，返回 False。

## 7. isOpenGroupHead &lt;node&gt;

如果指定对象为一个组头，且该组被打开，返回 True；否则，返回 False。

## 8. setGroupOpen &lt;group\_head\_node&gt; &lt;boolean&gt;

设定指定组的开关状态。如果<boolean>为 True，组被设为打开；如果<boolean>为 False，组被设为关闭。

使用下面的方法时应小心，因为你为对象设置的状态可能是 3ds max 系统无法执行的：

## 9. setGroupMember &lt;node&gt; &lt;boolean&gt;

设置对象是否为组成员。如果<boolean>为 True，对象被设为一个组成员；如果为 False，

而之前对象为一个组成员，对象被解除与组头之间的连接。

**注意** 如果用本方法将一个对象设置为组成员，还必须接着将该对象设置为一个组头对象的子对象；否则该对象名不会显示在Select By Name对话框里。如：

```
setGroupHead $dummy01 True
append $group03.children $dummy01
```

#### 10. setGroupHead <node> <boolean>

设置指定对象是否为一个组头对象。

**注意** 如果用本方法将一个对象设置为组头对象，对象的网格(Mesh)将不会显示在视窗里，其属性也不会在Modify面板里显示。如果用本方法将一个对象设置为非组头对象，将无法通过Group菜单命令来打开或炸开该对象所在的组。

下面的示例创建一系列的Sphere对象集，并将它们形成一个组，然后测试组头和组成员：

```
mySpheres=for i=1 to 5 collect sphere \
pos:(random [-100,-100,0] [100,100,0])
group MySpheres name:"MyGroup"
isGroupHead $MyGroup           --返回 True
isGroupMember $sphere01         --返回 True
--测试组是否被打开。打开组并测试组成员
isOpenGroupHead $MyGroup       --返回 False
setGroupHeadOpen $MyGroup True
isOpenGroupMember $sphere01     --返回 False
--创建一系列新的sphere对象集，并把上一个组的对象合并到对象集里,
--然后将它们组成一个大组
NewSpheres=for i=1 to 3 collect sphere \
pos:(random [-100,-100,0] [100,100,0])
append NewSpheres $MyGroup
group NewSpheres name: "BiggerGroup"
--打开组，测试组成员，然后关闭组
setGroupHeadOpen $BiggerGroup True
isOpenGroupMember $MyGroup
setGroupHeadOpen $MyGroup False
setGroupHeadOpen $BiggerGroup False
```

### 8.2.4 与视窗状态有关的方法

- |                 |                 |
|-----------------|-----------------|
| 1. hide <node>  | mapped, 隐藏对象    |
| unhide <node>   | mapped, 解除对象的隐藏 |
| freeze <node>   | mapped, 冻结对象    |
| unfreeze <node> | mapped, 解冻对象    |

#### 2. flagForeground <node> <boolean> mapped

控制对象是显示在视窗的前景平面还是背景平面里，这样可以改变对象的交互执行方式。如果对象显示在前景平面，其会被单独刷新，此时如果用脚本Rollout里的Spinner控

件来改变对象的属性，运行速度会比较快。

如果参数<boolean>为 True，表示对象显示在前景平面里；如果为 False，表示对象显示在背景平面里。用户在将对象放置在前景平面里时要谨慎，因为如果将太多对象放置在前景平面里会降低系统运行的速度。记住：如果不再需要对对象进行交互改变时，就应该将它放到背景平面里去。

### 3. getTrajectoryOn <node>

如果显示对象的运动轨迹，返回 True；否则，返回 False。

### 4. setTrajectoryOn <node> <boolean>

设置是否显示对象的运动轨迹。

### 5. isBoneOnly <node>

如果对象的.showLinksOnly 属性为 True，返回 True；否则，返回 False。

### 6. getCVertMode <node>

如果在 Shade（着色）视窗里用顶点颜色来显示对象，返回 True；否则，返回 False。

### 7. setCVertMode <node> <boolean>

设置在 Shade（着色）视窗里是否用顶点颜色来显示对象。如果<boolean>为 True，对象用顶点颜色来显示；如果为 False，对象用线框（Wireframe）颜色来显示。

### 8. getShadeCVerts <node>

如果在视窗里顶点显示颜色被着色，返回 True；否则，返回 False。

### 9. setShadeCVerts <node> <boolean>

设置在视窗里顶点显示颜色是否被着色。如果<boolean>为 True，顶点颜色没有被着色，用纯 RGB 颜色显示；如果为 False，顶点颜色被着色，此时颜色看起来就像另一种颜色。

## 8.2.5 与对象选择有关的方法

### 1. clearSelection()

clearNodeSelection [ redraw:<boolean> ]

取消对当前所有场景对象的选择。除非指定 redraw:False，对象都会被刷新。

### 2. deselect <node> mapped

取消对指定对象的选择。

### 3. deselectNode <node>

取消对单个对象的选择。

### 4. select <node> mapped

先取消对当前所有被选场景对象的选择，然后选择指定对象。

### 5. selectMore <node> mapped

将指定对象添加到当前对象选择集里。

## 8.2.6 与修改器堆栈（Modifier Stack）有关的方法

### 1. validModifier [ <node> | <objectset> | <group> ] <modifier>

测试指定<modifier>是否可以被添加到指定对象或对象集/组的所有对象里。如果返回

True，表示可以；返回False，则表示不可以。

如果<objectset>为一个空对象集，返回值为True；而如果指定<group>的所有成员都可以添加指定Modifier，返回值也为True。但是在这两种情况下，使用addModifier()方法来添加指定Modifier都会不成功，因为在第一种情况下，对象集为空，而第二种情况下，组<group>包含对象的父对象为一个Dummy类对象，而修改器是不能被添加到Dummy类对象里的。

## 2. addModifier <node> <modifier> [before:index] mapped

将指定<modifier>施加到指定对象上。可选参数before:用来指定新<modifier>在修改器堆栈里的插入位置，从堆栈的顶部开始计数。如果指定对象在Modify面板下被选择，且处在子对象选择层级，<modifier>也可以被施加到指定对象的任何合适的子对象选择集上。我们可以用3ds max系统变量subObjectLevel来获取Modify面板下当前的子对象选择层级。如：

```
max modify mode          --打开Modify面板
select $Box01             --在Modify面板下选择对象Box01
subObjectLevel = 3         --设置子对象选择层级为Face
addModifier $Box01(ffd_2x2x2()) --添加一个FFD_Modifier到这些Face
```

如果参数<node>为一个对象集，则集合里每一个对象成员都会被添加一个<modifier>的副本。和在3ds max界面下交互施加Modifier不同，每一个Modifier副本定位架(Gizmo)的位置、大小和被施加Modifier的对象一致。如果要像3ds max界面下交互施加Modifier的效果一样，可以使用方法modPanel.addModToSelection()。

## 3. deleteModifier <node> <modifier\_or\_index> mapped

从修改器堆栈里删除指定Modifier。参数<modifier\_or\_index>可以为一个Modifier值或一个序号。

## 4. collapseStack <node> mapped

塌陷指定对象的修改器堆栈。如果调用本方法时修改器堆栈里还没有任何Modifier，本方法什么也不会执行；如果想强制将一个对象转换为一个Editable\_Mesh对象，可以使用方法convertToMesh()。

### 8.2.7 与对象Modifier关联转换有关的方法

#### 1. getModContextTM <node> <modifier>

返回一个Matrix3值，表示指定Modifier的子对象使用Local坐标系的转换矩阵。在MAXScript里存取Modifier的子对象(如.gizmo或.center)，返回值都处在这个Local坐标系下。如果Modifier不是被施加到子对象层级上，或者Modifier为在3ds max界面下交互地施加到一个对象集，这个转换矩阵就是对象的Local坐标系本身；然而如果Modifier被施加到子对象层级上，或者Modifier被施加到一个对象集，这个转换矩阵表示了对象集的位置和方向，可以用getModContextTM()方法来获取Modifier子对象的World坐标系的.transform属性。

#### 2. getModContextBBoxMin <node> <modifier>

```
getModContextBBoxMax <node> <modifier>
```

返回一个 Point3 值, 表示指定 Modifier 边界框 World 坐标系坐标值的最大值和最小值。

### 8.2.8 与对象转换有关的方法

1. convertToMesh <node> mapped

本方法将指定场景对象转换为 Editable\_Mesh 类对象。本方法与 collapseStack()类似, 但有一点不同: 本方法总是会将指定对象转换为 Editable\_Mesh 类对象, 即使该对象没有被施加任何 Modifier。

本方法适用于任何可以被施加 Edit\_Mesh\_Modifier 的对象, 比如 Geometry 和 Shape, 但是不适用于 Helper、Spacewarp、Light 等。如果指定对象不能被转换, 返回值 undefined。

2. convertToSplineShape <node> mapped

本方法将指定场景对象转换为 SplineShape 类对象。如果指定对象不能被转换(如 Shape 类对象), 返回值 undefined。对象里的所有 Modifier 都会被塌陷。

3. canConvertTo <node> <class>

用来检测指定对象是否可以转换为指定类。如:

```
if canConvertTo $foo NURBSSurface then ...
```

对象可以转换的类有: Mesh、SplineShape、NURBSCurve、NURBSSurface 等。

4. convertTo <node> <class> mapped

本方法为其他类型转换方法如 convertToMesh()、convertToSplineShape()的通用格式。比如 convertToSplineShape()可以改成下面的格式:

```
convertTo $circle01 SplineShape
```

如果系统不支持指定转换, 返回值 undefined。

5. convertToNURBSSurface <node> mapped

```
convertToNURBSCurve <node> mapped
```

上面方法适用于基本几何体 (Primitive Geometry) 和 Shape 类对象。如果系统不支持指定转换, 返回值 undefined。

6. isDeleted <MAXWrapper\_object>

如果指定对象已经被删除, 返回 True; 否则, 返回 False。本方法主要用在下面的情况: 如果一个变量或数组引用了 3ds max 对象, 用本方法可以检测引用的对象是否被从场景里删除了。对一个已被删除的对象执行本操作会导致一个运行异常。本方法适用于任何 3ds max 对象, 如 Node、Modifier、Controller、Material 等。如:

```
sel = selection as array
...
--删除一些对象
...
for obj in sel
where not isDeleted obj do
move obj [10,0,0]
```

## 7. distance &lt;node&gt; &lt;node&gt;

返回指定的两对象支点(pivot)之间的距离。

## 8. intersectRay &lt;node&gt; &lt;ray&gt;

返回射线<ray>和指定对象表面之间最近的交点。返回值为一个Ray值，定义了交点的位置和该点表面法线的方向。如果两者不相交，返回值undefined。

## 9. intersectRayEx &lt;node&gt; &lt;ray&gt;

返回射线<ray>和指定对象表面之间最近的交点。返回值为一个三元数组，其元素代表的意义分别为：一个Ray值，定义了交点的位置和该点表面法线的方向、相交面的序号、相交面的重心坐标。

如果两者不相交，返回值undefined。

重心坐标定义了交点p相对于三角面三个顶点坐标的权重之和。比如p点的重心坐标为b0、b1、b2，那么p点的坐标值为：

$p = b0 * p0 + b1 * p1 + b2 * p2;$

其中p0、p1、p2为三角面三个顶点的位置。

下面的示例演示了如何在交点查找UV坐标：

```
s = sphere material:(standardMaterial diffuseMap:(checker()))
showTextureMap s.material s.material.diffuseMap on
--添加一个normalModifier，便于将sphere对象转换为Mesh对象
addModifier s(normalModifier())
r = ray [-100,5,0](s.center[-100,5,0])      --获取交点信息
arr =(intersectRayEx s r)                      --在交点处创建一个Dummy对象
dummy pos:(arr[1]).pos                         --获取纹理面
tf = getTVFace s arr[2]                          --获取面的UVW vert
tv1 = getTVert s tf.x
tv2 = getTVert s tf.y
tv3 = getTVert s tf.z
--在barycentric坐标系计算纹理顶点
tv = tv1*arr[3].x + tv2*arr[3].y + tv3*arr[3].z
deleteModifier s 1     --删除modifier
```

## 10. intersects &lt;node&gt; &lt;node&gt;

如果指定两对象的边界框重叠，返回True；否则，返回False。

## 11. printStack &lt;node&gt;

打印指定对象Modifier Stack(堆栈)里的内容。

## 8.2.9 与用户定制属性有关的方法

## 1. getUserProp &lt;node&gt; &lt;key\_string&gt;

返回一个String值，表示指定对象定制属性的当前值。定制属性参数<key\_string>可以为String或Name类值。如果该属性不存在，返回undefined。

## 2. setUserProp &lt;node&gt; &lt;key\_string&gt; &lt;value&gt;

给对象的定制属性赋给指定值。

## 3. getUserPropBuffer &lt;node&gt;

返回一个 String 值，包含指定对象所有的定制属性值。

## 4. setUserPropBuffer &lt;node&gt; &lt;string&gt;

为对象的定制属性缓存赋值。

## 8.2.10 与 IK 属性有关的方法

下面的方法用来存取 Hierarchy | IK 面板下 Parameters 卷展栏里的值。

## 1. getRotTaskWeight &lt;node&gt;

返回对象的 Rotation Binding Weight 参数值。

## 2. setRotTaskWeight &lt;node&gt; &lt;Float&gt;

设置对象的 Rotation Binding Weight 参数值。

## 3. getPosTaskWeight &lt;node&gt;

返回对象的 Position Binding Weight 参数值。

## 4. setPosTaskWeight &lt;node&gt; &lt;Float&gt;

设置对象的 Position Binding Weight 参数值。

## 5. getTaskAxisState &lt;node&gt; &lt;pos\_or\_rot\_Integer&gt; &lt;axis\_Integer&gt;

如果指定轴被打开，用来进行 Position 或 Rotation 绑定，返回 True；否则，返回 False。

如果返回 False，表示该轴不再受跟随对象或 IK Controller Position 影响。参数 <pos\_or\_rot\_Integer> 表示方法返回的状态：

0: Position; 1: Rotation

参数 <axis\_Integer> 指定了要检测的轴：

0: X 轴; 1: Y 轴; 2: Z 轴

## 6. setTaskAxisState &lt;node&gt; &lt;pos\_or\_rot\_Integer&gt; &lt;axis\_Integer&gt; &lt;boolean&gt;

设置指定轴是否要打开来进行 position 或 rotation 绑定。参数说明见上条。

## 7. mirrorIKConstraints &lt;node&gt; &lt;axis\_Integer&gt; &lt;pos\_or\_rot\_Integer&gt;

镜像对象转换 Controller 的 IK 约束。

<axis\_Integer> 指定了镜像轴：

0: X 轴; 1: Y 轴; 2: Z 轴

参数 <pos\_or\_rot\_Integer> 用来指定镜像的约束类型：

0: Position; 1: Rotation

## 8. nodeIKParamsChanged &lt;node&gt;

当某一 Node 级 IK 参数被修改时调用本方法。

## 9. OKToBindToNode &lt;ik\_node&gt; &lt;node&gt;

如果 <ik\_node> 对象可以被绑定为 <node> 对象的跟随对象，返回 True；否则，返回 False。

即使 <ik\_node> 对象不是 IK 系统的一部分，本方法也会返回 True。本方法用来检测 <node> 对象的转换是否有可能依赖于 IK 系统，如果是，返回 False。

### 8.2.11 Node类其他方法

#### 1. classOf <node>

返回对象所属的类名。如果对象没有被施加 Modifier，返回值为基本对象的类名；如果对象被施加 Modifier，返回值为最后一个 Modifier 被施加后的对象类名。例如：假设施加一个 Bend Modifier 给一个 Box 对象，Bend Modifier 会将 Box 对象转换为一个 Mesh 类对象，调用本方法的返回值为 Editable\_Mesh。如果要知道基本对象的类型，可以使用下面的方法：

```
classOf <node>.baseObject
```

#### 2. isPointSelected <node> <point\_index>

如果指定点<point\_index>被选择，返回 True；否则，返回 False。“点”的定义根据对象类型不同而不同。对 Mesh 类对象，为 Mesh 顶点；对 Spline 对象，为结点（Knot）；对 NURBS 类对象，为顶点或控制点。

#### 3. pointSelection <node> <point\_index>

如果对象支持“软选择”，返回一个浮点权重型点选择集。对绝大多数对象类型而言，如果点被选，返回 1.0；否则，返回 0.0。目前只有 NURBS 和 Editable\_Mesh 类对象支持权重型点选择集。

#### 4. nodeInValRect <node>

将对象在视窗里占据的矩形标记为 invalid，这样该区域会在下一次屏幕重绘时被刷新。

#### 5. stopCreating <node>

本方法停止当前对象的创建（如果有的话）。本方法主要用来确保一个 NURBS 对象被完整创建，本方法同时会使 Create 面板下的对象创建按钮失效。

相关方法：

#### 6. uniqueName <prefix>

通过在指定前缀字符串后面添加一系列字符来产生一个唯一的场景对象名。本方法仅保证该名称在下一个场景对象创建之前的唯一性。如：

```
$foo.name = uniqueName "foo"
```

## 8.3 Node类对象属性

### 8.3.1 Node通用属性

下面所列属性可由任何 Node 子类存取，除非特别注明，这些属性不能设置动画：

#### 1. <node>.name: string，默认值：不定

存取对象的名字。

#### 2. <node>.baseObject: Node 的子类，默认值：不定，只读

返回 Node 对象的基本对象类名。适用于那些被修改器堆栈里的修改器改变了对象类型的对象。例如：对一个 Line 类对象施加一个 Extrude 修改器后变成了一个 Shape 类对象，

而在修改器堆栈的顶部，它又变成了 Editable\_Mesh，最后的类型我们称之为 World State。方法 classof()让用户可以获知 World State 对象的类名，而.baseobject 属性可以返回 World State 对象的原始对象类名。我们选择上面对象后执行下面的操作：

```
classof $ --返回 Editable_Mesh
$. BaseObject --返回 Line

3. <node>.material: material, 默认值: undefined
```

存取对象材质。

```
4. <node>.parent: Node, 默认值: undefined
```

存取对象的父对象，如果对象为顶级对象，.parent 属性返回 undefined，如：

```
foo.mom=$foo.parent
$foo.parent=$baz
```

设置对象的.parent 属性会解除其与原来父对象的主从关系，重新与指定对象建立父子关系。

通过将对象的.parent 属性赋为 undefined 可以强制指定该对象为顶级对象，如：

```
$foo.parent=undefined
```

```
5. <node>.children: NodeChildrenArray, 默认值: undefined
```

返回一个特殊的子类数组 (NodeChildrenArray)，如：

```
c=$foo.children [i]
move $foo.children do print c
```

可以用.count 属性获取子对象个数。

```
num.children=$foo.children.count
```

不能通过直接增加 NodeChildrenArray 数组的长度来给对象添加一个子对象，但可以用 append() 和 deleteItem() 方法来增减子对象，如：

```
$foo.children[$foo.children.count+1]=$baz --不起作用
append $foo.children $baz --可以起作用
deleteItem $foo.children $baz --将 baz 子对象从集合中移走
```

子对象的索引顺序与 3ds max 的 Hierarchy 视窗里显示的顺序相同，都是按加入父对象的先后顺序排序。

6. <node>.mesh: TriMesh

对 Editable\_Mesh 类或可转化为 Editable\_Mesh 类的场景对象或基本对象，该属性返回一个 Trimesh 值，包含源对象在所有的修改器被施加到该对象上 (SpaceWarps 除外) 之后的 Mesh 对象。该属性一般是只读的，但对那些场景对象类型或基本对象类型为 Editable\_Mesh 的对象，可进行存取操作。如果对.mesh 属性赋值，指定的 Trimesh 值会取代基本对象。

### 8.3.2 与Target/LookAt有关的属性

#### 1. <node>.isTarget: Boolean, 默认值: False

如果Node为另一个对象LookAt控制器的目标对象，返回True；否则，返回False。如果对象为Light或Camera的LookAt控制器自动创建的目标对象，把该属性设为False后，删除target对象不会同时删除Light或目标对象。

#### 2. <node>.lookAt: Node, 默认值: undefined

返回一个以<node>为目标的对象，或指定一个以<node>为目标对象的对象。本属性与<node>.target为一对互逆属性。如果<node>不是其他对象LookAt控制器的目标对象，返回值为undefined；如果<node>为不止一个对象的目标对象，仅返回一个对象；如果用赋值语句设置<node>为另一对象的目标对象，一个LookAt控制器会自动添加给<node>对象。

#### 3. <node>.target: Node, 默认值: undefined

存取<node>对象LookAt控制器的目标对象。如果该对象没有目标对象或没有LookAt控制器，返回undefined；如果用赋值语句来给<node>指定另一对象作为目标对象，一个LookAt控制器会自动添加给<node>对象。在实践中使用.target属性赋值语句应注意：当为对象指定目标对象时，3ds max会将目标对象和它最后一次被指定为目标对象的源对象存储在一起，如果删除目标对象，会同时删除掉属性.lookAt指定的对象；如果目标对象是一个Targetobject类对象，删除该对象(.lookAt属性指定的对象)会同时删除该目标对象，即使另一对象的.lookAt属性也指向该target对象。

#### 4. <node>.targetDistance: Float, 默认值: undefined

存取<node>对象到它的目标对象之间的距离。如果<node>没有目标对象，返回undefined。给该属性赋新值会将目标对象沿对象与目标对象连线移动至指定距离，如：

```
$spot01.targetDistance=100
```

### 8.3.3 与视窗有关的属性

#### 1. <node>.isSelected: Boolean, 默认值: False

存取对象是否被选中。

#### 2. <node>.isHidden: Boolean, 默认值: False

存取对象在视窗中是否被标记为Hidden。一个对象即使它的.isHidden属性为False，也可能被隐藏。如果在Display面板的Hide by Category卷展栏里将对象所属的Category标记为Hidden，或者在Display面板的Hide卷展栏里Hide Frozen Objects可选框设为On时将对象标记为Frozen，对象都会被隐藏，而不管属性.isHidden的值。

#### 3. <node>.xray: Boolean, 默认值: False

当在see Through模式时，存取在shaded视窗里是否显示对象。

#### 4. <node>.ignoreExtents: Boolean, 默认值: False

当执行Zoom Extents命令时，对象的范围是否被忽略。

#### 5. <node>.BoxMode: Boolean, 默认值: False

当在Box模式时，是否在视窗里显示对象。

6. <node>.allEdges: Boolean, 默认值: False  
存取对象的所有边是否全部显示在视窗里。
7. <node>.backFaceCull: Boolean, 默认值: False  
存取对象的背面是否显示在视窗里。
8. <node>.wirecolor: color, 默认值: 随机颜色  
存取对象线框颜色。
9. <node>.showLinks: Boolean, 默认值: False  
存取是否显示对象之间层级关系的线框。
10. <node>.showLinksOnly: Boolean, 默认值: False  
存取是否仅显示对象之间层级关系的线框。将本属性设为 True 会同时将属性.showLinks 设为 True。
11. <node>.isFrozen: Boolean, 默认值: False  
存取对象在视窗里是否被冻结。
12. <node>.showTrajectory: Boolean, 默认值: False  
存取对象的运动轨迹是否显示在视窗里。
13. <node>.showVertexColors: Boolean, 默认值: False  
存取在 Shaded 视窗里是否显示对象顶点颜色。
14. <node>.VertexColorsShaded: Boolean, 默认值: False  
存取在视窗里是否对对象顶点颜色着色。如果设为 True, 顶点颜色不会被着色, 用纯 RGB 颜色显示在视窗里。
15. <node>.isDependent: Boolean, 默认值: False  
如果指定<node>对象的 Dependent 标志被打开, 返回 True; 否则, 返回 False。本属性为只读属性。如果打开了 3ds max 下拉菜单 Views | Show Dependencies 选项, 当 Modify 面板被打开时, 与当前被编辑的 Modifier 或对象存在父子关系的 Node 对象都会用绿色亮显, 这些 Node 对象的 Dependent 标志被打开; 如果 Modify 面板被关闭, 或 Views | Show Dependencies 选项没有被选择, 本属性均返回 False。

#### 8.3.4 与层有关的属性

1. <node>.displayByLayer: Boolean, 默认值: True  
当设置为 On 时, 对象的可见性将由层控制。
2. <node>.motionByLayer: Boolean, 默认值: True  
当设置为 On 时, 运动模糊将由层控制。
3. <node>.renderByLayer: Boolean, 默认值: True  
当设置为 On 时, 渲染将由层控制。
4. <node>.colorByLayer: Boolean, 默认值: True  
当设置为 On 时, 对象将使用层的颜色; 否则, 使用对象颜色。
5. <node>.globalIlluminationByLayer: Boolean, 默认值: True  
当设置为 On 时, 高级灯光的包含/排除功能将由层控制。

### 8.3.5 与渲染有关的属性

1. <node>.castShadows: Boolean, 默认值: True

存取对象渲染时是否投射阴影, 本属性对应鼠标右击对象后弹出菜单: Object properties | General | Cast Shadows。

2. <node>.receiveShadows: Boolean, 默认值: True

存取对象渲染时是否接受阴影, 本属性对应: Object properties | General | Receive Shadows。

3. <node>.gbufferChannel: Integer, 默认值: 0

存取对象的 G-buffer 对象 ID 通道值, 有效的对象 ID 值为 0~65535。本属性对应: Object properties | General | G-Buffer | Object Channel。

4. <node>.visibility: Boolean, 默认值: True 可动画

存取对象是否可见。True 或 on 表示可见; False 或 off 表示不可见。可在渲染时改变本属性来控制对象的可见性。如:

```
animate on
(
  at time 0 $foo.visibility = on
  at time 35 $foo.visibility = off
  at time 57 $foo.visibility = on
)
```

本属性的控制器还可用下面方法进行存取:

<node> [1].Controller 可见性控制器

也可以用 getVisController() 方法来获得对象的可见性控制器。

5. <node>.inheritVisibility: Boolean, 默认值: True

存取对象是否继承父对象的可见性属性。本属性对应: Object properties | General | InheritVisibility。

6. <node>.renderable: Boolean, 默认值: True

存取是否渲染该对象。

7. <node>.renderOccluded: Boolean, 默认值: False

存取是否渲染阻光对象, 本属性对应: Object properties | General | Render Occluded Objects。

8. <node>.motionBlurOn: Boolean, 默认值: True

存取对象是否使用运动模糊, 本属性对应: Object properties | General | Motion Blur Enabled。

9. <node>.motionBlurOnController: Controller, 默认值: undefined

存取对象运动模糊 On/Off 控制器。本属性对应: Object properties | Motion Blur | Enabled。

10. <node>.MotionBlur: Name, 默认值: #none

存取对象运动模糊类型, 有效的参数值有: #none、#Object、#image。为与前面版本兼

容，本属性也可设为 False 表示没有运动模糊，或者设为 True 表示存在运动模糊。本属性对应：Object properties | Motion Blur | Type。

11. <node>.imageMotionBlurMultiplier: Float，默认值：1.0，可动画  
存取对象的图像运动模糊倍增器值。本属性对应：Object properties | Motion Blur | Multiplier。

本属性的控制器还可用下面方法存取：

<node> [6].Controller

也可以用下面的方法来获取对象的 imageMotionBlurMultiplier 控制器：

```
getImageBlurMultController()  
setImageBlurMultController()
```

12. <node>.rcvcaustics: Boolean，默认值：True

存取对象是否接受焦散。3ds max 的 Scanline 渲染器不支持焦散。

13. <node>.rcvGlobalIllum: Boolean，默认值：True

存取对象是否接受全局照明。3ds max 的 Scanline 渲染器不支持全局照明。

14. <node>.generateGlobalIllum: Boolean，默认值：False

存取对象是否产生全局照明。3ds max 的 Scanline 渲染器不支持全局照明。

### 8.3.6 与转换有关的属性

除非另外注明，下面与 Transform 有关的属性总是由当前活动的 coorsys 关联语句定义的坐标系为参照进行解释，默认的坐标系为 World。对象的 pos、rotation 和 scale 属性是对象 Transform 控制器相应子控制器的别名。如果 Transform 控制器没有某项子控制器，存取与之相应的属性值会返回一个 Unknown property 的错误信息。如：LookAt 控制器没有 Rotation 子控制器，IK 控制器没有任何子控制器。

1. <node>.transform: Matrix3，可读写

Node 对象的主转换矩阵。如果给一个 Node 对象的.transform 属性赋值，该 Matrix3 值会被分解成 Position、Rotation 和 Scale 值，并分别存储在相应的 Position、Rotation、Scale 控制器里（如果这些控制器已经存在且是可写的）。

**注意** 在内部转换里转角的正方向遵循左手定律，这一点与 3ds max 用户界面和 MAXScript 正好相反。

2. <node>.pos: Point3

也可用.position 来代替。

3. <node>.pos.controller: Controller

也可用.pos.track 来代替.pos.controller。

4. <node>.pos.isAnimated: Boolean

只读。如果属性.pos 被设置动画，返回 True。

5. <node>.pos.keys: MAXKeyArray

Position 控制器的关键帧。

6. <node>.pos.track: Controller  
也可用.pos.controller 来代替。
7. <node>.rotation: Quat
8. <node>.rotation.x\_rotation: Node  
对象在 X 轴方向的转角。
9. <node>.rotation.y\_rotation: Node  
对象在 Y 轴方向的转角。
10. <node>.rotation.z\_rotation: Node  
对象在 Z 轴方向的转角。
11. <node>.rotation.controller: Controller
12. <node>.rotation.isAnimated: Boolean  
只读。如果属性.rotation 被设置动画，返回 True。
13. <node>.rotation.keys: MAXKeyArray  
Rotation 控制器的关键帧。
14. <node>.rotation.track: Controller  
也可用.rotation.controller 代替。
15. <node>.scale: Point3
  - <node>.scale.controller: Controller  
也可用.scale.track 来代替.scale.controller。
16. <node>.scale.isAnimated: Boolean  
只读。如果属性.scale 被设置动画，返回 True。
17. <node>.scale.keys: MAXKeyArray  
Scale 控制器的关键帧。
18. <node>.scale.track: Controller  
也可用.scale.controller 来代替。
19. <node>.dir: Point3  
表示 Local 坐标系 Z 轴的矢量。
20. <node>.max: Point3  
只读。对象边界框的最大坐标值。
21. <node>.min: Point3,  
只读。对象边界框的最小坐标值。
22. <node>.center: Point3  
对象边界框的中心坐标值。
23. <node>.pivot : Point3  
Node 对象的支点坐标。
24. <node>.objectOffsetPos: Point3  
说明见 8.3.7 节。
25. <node>.objectOffsetRot: Quat

说明见 8.3.7 节。

26. <node>.objectOffsetScale: Point3

说明见 8.3.7 节。

27. <node>.objectTransform: Matrix3 只读

说明见 8.3.7 节。

### 8.3.7 使用 Node 对象的转换属性

接下来，我们要讨论与 Node 对象有关的三种转换：Node 转换矩阵（Node Transform Matrix）、对象偏移转换矩阵（Object-Offset Transform Matrix）和对象转换矩阵（Object Transform Matrix）。前面两种矩阵由 3ds max 进行存储，而第三种矩阵是由前面两种矩阵导出。

#### 对象转换矩阵——支点

当一个 Node 对象被选择时，一个代表 Local 坐标系的三角架被显示在对象上，三脚架的原点被称为转换中心，或支点（Pivot Point），是对象旋转、缩放的中心。所有 3ds max 里的 Node 对象都有它自己的支点。读者可以把支点看作 Node 对象的中心和它的 Local 坐标系原点。Node 对象的支点主要有下面几种用途：

- ◆ 作为旋转、缩放的中心。
- ◆ 为链接子对象定义转换原点。
- ◆ 为 Inverse Kinematics 操作定义连接位置。

实际上，显示在支点上的三脚架正是 Node 转换矩阵的一个形象的表示。Node 转换矩阵实际上由 Transform 控制器进行控制，而后者决定着 Node 对象在场景里的位置。

#### 构造用于 PRS 转换控制器的 Node 转换矩阵（Node Transform Matrix）

PRS 控制器使用子控制器来创建最后的转换。共有三种子控制器：一个用于位置（Position）、一个用于旋转（Rotation）、一个用于缩放（Scale）。首先，Node 对象父对象的转换矩阵被传递给 PRS 控制器。如果 Node 对象没有父对象，一个单位矩阵（Identity Matrix）被传递给 PRS 控制器，然后 PRS 控制器依次调用 Position 控制器、Rotation 控制器、Scale 控制器。在经过 Position、Rotation 和 Scale 控制器处理过后的转换矩阵就变成了 Node 转换矩阵，3ds max 用它来对 Node 对象在场景里进行定位、旋转、缩放。这样，完整的转换就是：

Node 转换矩阵 = Scale 控制器 \* Rotation 控制器 \* Position 控制器 \* 父对象转换矩阵

某些 Position 控制器实际上不仅仅可以对转换矩阵施加位置变化。比如：当 Path 控制器的 Follow 开关被打开时，为了使 Node 对象与路径保持相切，实际会施加一些旋转变化，这样当 Rotation 控制器开始处理转换矩阵时，实际已经有旋转被施加到转换矩阵上了。

#### 对象偏移转换矩阵（Object-Offset Transform Matrix）

对象偏移转换矩阵用来计算对象的几何体与 Node 之间的偏移。

在3ds max里选择一个对象，我们可以看到一个三脚架放在对象的支点处，然后进入Hierarchy面板，单击Pivot选项卡，选择Affect Pivot Only或Affect Object Only按钮，我们会看到在对象的支点处显示一个大的三脚架，表示支点的位置和方向。选择上面两个按钮中任一个，然后使用Move/Rotate/Scale工具栏，我们可以独立于支点来操作对象的几何体，也可以独立于对象的几何体来操作支点。

3ds max系统之所以能够独立操作对象的支点和几何体，是因为在系统内部使用了对象偏移转换矩阵，对象偏移转换矩阵定义了对象的几何体和Node之间的关系，即两者之间的转换。

为了理解对象偏移转换矩阵，我们在3ds max界面里执行下面的操作：

在Hierarchy面板的Pivot选项卡下选择Affect Object Only按钮。此时我们可以任意移动对象的几何体，而支点不动。在这个过程中我们实际是在对对象偏移转换矩阵进行转换。这个转换是一个施加于对象几何体的额外偏移，它独立于Node，对象偏移转换矩阵不会继承给对象的子对象。

在Hierarchy面板的Pivot选项卡下选择Affect Pivot Only按钮。此时我们可以任意移动支点而不影响对象几何体的位置。在这个过程中我们首先对对象的Node转换矩阵(Node Transform Matrix)进行替换(支点被重新定位)，然后对象偏移矩阵会被重新计算，以使对象的几何体保持在原来的位置。

#### 构造对象偏移矩阵 (Object-Offset Transform Matrix)

对象偏移矩阵分别由位置(Position)、旋转(Rotation)和缩放(Scale)三种转换组成。和Node转换矩阵相似，对象偏移矩阵的计算顺序为：

对象偏移矩阵= Scale 偏移 \* Rotation 偏移 \* Position 偏移

与Node转换矩阵不同的是：对象偏移矩阵不能继承给它的子对象。

#### 对象转换矩阵 (Object Transform Matrix)

对象转换矩阵用来将对象几何体转换至World坐标系的转换矩阵，它包含了父对象的转换、Node转换和对象偏移转换。这样，对象上某一点的全部转换为：

对象转换矩阵 = 对象偏移转换矩阵 \* Node 转换矩阵

对一个Mesh类对象，为了计算某一顶点在World坐标系里的位置，应该将该顶点在Object坐标系里的坐标乘以对象转换矩阵。

#### 在MAXScript里使用Node转换

与Node转换有关的属性是由与Node对象相连的Controller导出的，并且直接可以修改这些控制器，但这些属性并不直接与Node的主转换矩阵相关，两者之间有下面的关系：

如果赋值给动画的转换属性，仅会在相对应的控制器(如果有的话)里增加关键帧；而如果直接修改Node的转换矩阵，会在该Node所有与转换有关的控制器里增加关键帧。

属性.rotation及所有MAXScript里与旋转相关的函数都使用与3ds max界面里相同的右手规则(绕坐标轴的正方向逆时针旋转为正角度)。但是，3ds max将旋转存储到Node转换矩阵时使用左手规则。在使用属性.transform和.objectTransform时必须要记住这一点，如果要将它们和3ds max及MAXScript的标准旋转混合使用时，要先对它们进行转化操作。

(可以将坐标轴乘以[-1,-1,-1]或对 Quaternion 角度执行 inverse()函数)。

某些控制器，如 path Controller (当 Bank 和 Follow 开关被打开时)，会通过直接修改 Node 的转换矩阵来调整 Node 的转角，而不会改变 Rotation 控制器的值。这意味着，在这种情况下，由 Bank 和 Follow 开关引起的旋转不能通过.rotation 属性进行存取，而需要直接从 Node 的转换矩阵进行存取，如：

```
r = $foo.transform.rotationPart
```

它会返回一个 Quaternion 角度。记住，因为这个角度为直接从转换矩阵获得，使用的方向规则为左手规则，如果要在其他旋转操作里使用这个值，必须先对它进行转化操作：

```
$baz.rotation = inverse r
```

Node 对象的.transform 属性包含了 Node 转换矩阵，反映了 Node 对象支点的 Position、Rotation、Scale。属性.pivot 和.pos 会返回相同的值。如果直接给.pivot 属性赋值，会将支点移到指定位置，但 Node 对象的几何体不动。

Node 对象的.objectoffsetpos、.objectoffsetrot、.objectoffsetscale 属性分别包含了对象偏移转换矩阵的 Position、Rotation、Scale。而.objecttransform 属性包含了对象转换矩阵，因为这个矩阵为 Node 转换矩阵和对象偏移矩阵的乘积，所以这个属性为只读。注意这些属性的返回值都是在 World 坐标系下的。

下面的脚本先打印出一个 Box 对象（大小为 25x25x25，位置在[75,75,0]）的.transform 属性，然后将它的支点移到点[50,50,0]，并将支点绕 Z 轴旋转 35°。每次转换后都将 Box 对象的一个顶点的位置打印出来。

```
fn DumpXForms obj =
( --输出 Node.transform 属性
  format "%:\t%\n" "transform" obj.transform
  format "%:\t%\n" "position" obj.pos
  format "%:\t%\n" "rotation" obj.rotation
  --输出 Node 支点位置
  format "%:\t%\n" "pivot" obj.pivot
  --输出对象偏移
  format "%:\t%\n" "objectoffsetpos" obj.objectoffsetpos
  format "%:\t%\n" "objectoffsetrot" obj.objectoffsetrot
  format "%:\t%\n" "objectoffsetscale" obj.objectoffsetscale
  --输出对象转换矩阵
  format "%:\t%\n" "objecttransform" obj.objecttransform
  --分别输出第一个顶点在 local 和 world 坐标系里的坐标
  format "%:\t%\n" "vert 1(local)" \
  (in coordsys local getvert obj 1)
    format "%:\t%\n" "vert 1(world1)" \
  (in coordsys world getvert obj 1)
    --计算并输出第一个顶点在 world 坐标系里的位置
    local v_pos=(in coordsys local getvert obj 1)* obj.objecttransform
    format "%:\t%\n" "vert 1(world2)" v_pos
) --函数 DumpXForms 定义结束
--
```

```
--定义一个仅旋转支点的函数
fn RotatePivotOnly obj rotation=
( local rotValInv=inverse(rotation as quat)
  animate off in coordsys local obj.rotation*=RotValInv
  obj.objectoffsetrot*=RotValInv
  obj.objectoffsetpos*=RotValInv
)
--
(
--创建一个 25x25x25 的 Box 对象, 第一个顶点在[62.5,62.5,0](world)
b=Box pos:[75,75,0]
--将 Box 对象转换为 Mesh 对象, 这样我们可以存取顶点位置
convertToMesh b
--打印转换矩阵
DumpXForms b
--移动支点
b.pivot=[50,50,0]
--重新打印转换矩阵
DumpXForms b
--将支点绕 z 轴旋转 35°
RotatePivotOnly b(EulerAngles 0 0 35)
--重新打印转换矩阵
DumpXForms b
)
```

输出结果:

```
DumpXForms()           --函数定义
RotatePivotOnly()--函数定义
--初始转换矩阵
transform: (matrix3 [1,0,0], [0,1,0], [0,0,1], [75,75,0])
position: [75,75,0]
rotation: (quat 0 0 0 1)
pivot : [75,75,0]
objectoffsetpos : [0,0,0]
objectoffsetrot : (quat 0 0 0 1)
objectoffsetscale: [1,1,1]
objecttransform : (matrix3 [1,0,0], [0,1,0], [0,0,1], [75,75,0])
vert 1(local) : [-12.5,-12.5,0]
vert 1(world1): [62.5,62.5,0]
vert 1(world2): [62.5,62.5,0]
--移动后转换矩阵
transform: (matrix3 [1,0,0], [0,1,0], [0,0,1], [50,50,0])
position: [50,50,0]
rotation: (quat 0 0 0 1)
pivot : [50,50,0]
objectoffsetpos : [25,25,0]
objectoffsetrot : (quat 0 0 0 1)
objectoffsetscale: [1,1,1]
objecttransform : (matrix3 [1,0,0], [0,1,0], [0,0,1], [75,75,0])
vert 1(local) : [-12.5,-12.5,0]
```

```

vert 1(world1): [62.5,62.5,0]
vert 1(world2): [62.5,62.5,0]
--旋转后转换矩阵
transform: (matrix3 [0.819152,0.573576,0], \
[-0.573576,0.819152,0], [0,0,1], [50,50,0])
position: [50,50,0]
rotation: (quat 0 0 0.300706 0.953717)
pivot : [50,50,0]
objectoffsetpos : [34.8182,6.13939,0]
objectoffsetrot : (quat 0 0 0.300706 0.953717)
objectoffsetscale: [1,1,1]
objecttransform : (matrix3 [1,0,0], [0,1,0], [0,0,1], [75,75,0])
vert 1(local) : [-12.5,-12.5,0]
vert 1(world1): [62.5,62.5,0]
vert 1(world2): [62.5,62.5,0]

```

### 8.3.8 定制 Node 属性

本节中的函数可以存取场景对象的用户定制属性，打开对象的 Object property | User Defined 选项卡，可以看到这些属性。存取这些属性有两种方法：一种方法是把所有的属性内容作为一个字符串；另一种方法是用 key | property 格式进行存取：

<key1> = <property1>

<key2> = <property2>

...

其中<key>为标识符（数据类型为 Name 或 String）；<property>为属性值，可以为 Number、Boolean（True/False）或 String。在 MAXScript 里有两种方法可以存取单独的用户定制属性，有两种方法可以将用户定制属性视为一个字符串进行存取。

**注意** 目前在 3ds max SDK 里有一个缺陷：读取字符串属性时，系统会截去字符串第一个空格后的字符。

1. getUserProp <node> <key\_string>

获取指定<node>的指定用户定制属性<key\_string>的值。其中<key\_string>数据类型为 String 或 Name。如果指定<key\_string>尚未定义，返回值 undefined。

2. setUserProp <node> <key\_string> <value>

给指定<node>的指定用户定制属性<key\_string>赋给值<value>。

3. getUserPropBuffer <node>

获取指定<node>的全部用户定制属性缓存，返回值为一个包含所有属性设置的字符串。

假如给一个对象定义了两个定制属性：Pro1 和 Pro2，其中 Pro1 数据类型为 Number，值为 123，Pro2 数据类型为 String，值为"123"，调用本函数返回值为：

```

"Prop1 = 123
Prop2="123"

```

4. setUserPropBuffer <node> <string>

给指定<node>的指定用户定制属性缓存赋给值<string>。

下面的脚本程序可以对场景对象的用户定制属性进行存盘和装载操作。

第一个程序段首先创建一个文件，对每一个对象进行循环，依次输出各对象的.name 属性和用户定制属性缓冲字符串。

第二个程序段读取由第一个程序创建的文件，并将用户定制属性添加到当前场景里同名的对象上。

```
--为所有对象创建文件
--将对象名和用户定制属性输出到该文件里
f = createFile "foo.dat"
for o in objects do
(
    format "$%\n" o.name to:f          --以路径名格式输出对象名
    print(getUserPropBuffer o)to:f     --将缓存作为字符串输出
)
close f --关闭文件
--打开文件，读取对象名及用户定制属性缓存
f = openFile "foo.dat"
while not eof f do
(   o = readValue f --读取对象
--如果场景里有同名对象，读取用户定制属性缓存
if o != undefined then
    setUserPropBuffer o(readValue f)
)
close f
```

## 8.4 Node子类

在下列Node子类中，列出的属性都可以作为可选的关键字型变量和Node的通用关键字型变量一起出现在构造函数中。属性中一起列出了名称、数据类型、默认值。下面是所有3ds max的Node子类：GeometryClass、Shape、Light、Camera、Helper、SpacewarpObject。

### 8.4.1 GeometryClass: Node

下面列表显示了所有的3ds max Geometry类对象：

Geometry对象类型	子类名
Standard Primitive	Box、Cone、Cylinder、Geosphere、Plane、Pyramid、Sphere、Teapot、Torus、Tube
Extended Primitive	Capsule、ChamferBox、ChamferCyl、C_Ext、Gengon、Hedra、L_Ext、OilTank、Prism、RingWave、Spindle、TargetObject、Torus_Knot
Compound Object	OldBoolean、Boolean2、Conform、Connect、Loft、Morph、Scatter、ShapeMerge、Terrain
Particle System	Blizzard、PArray、PCloud、Snow、Spray、SuperSpray
Patch Grid	QuadPatch、TriPatch

(续表)

Geometry 对象类型	子类名
NURBS Surface	NURBSSurf、Point_Surf
Dynamics Object	Damper、Spring
Door	BiFold、Pivot、SlidingDoor
Window	Awning、Casement、Fixed、Pivoted、Projected、SlidingWindow
AEC Extended Object	Terrain、Foliage Railing、Wall_GeometryClass
Stair	L_Type_Stair、Spiral_Stair、Straight_Stair、U_Type_Stair
Miscellaneous	LinkComposite、Host_Composite

### 8.4.2 GeometryClass 的操作符和方法

在 MAXScript 里 Mesh 操作优先于 3ds max 的 Boolean Compound Object (即我们常说的布尔运算)。布尔运算符有“=”、“-”、“\*”三种，可作用于任何两个可转化成 Mesh 的场景对象。除了粒子系统以外，这种运算生成的对象都属于 GeometryClass 类。

#### 操作符

<Node1> + <Node2> Node1 和 Node2 的联合  
 <Node1>-<Node2> 从 Node1 减去 Node2  
 <Node1> \* <Node2> Node1 和 Node2 的交集  
 例如：

```
$foo-$baz
$sphere01 + $sphere02 + $sphere03 + $sphere04
```

布尔操作符将改变后的第一个操作对象作为运算结果，而第二个操作对象保持不变。在上面例子中，第一个操作对象完成布尔运算后，原始的第一个操作对象将不再存在。为了保留第一个操作对象，可以使用 copy 函数，如：

```
result =(copy $foo)-$baz
```

将在 foo 对象的副本上进行布尔运算，而保持原始的第一个操作对象不变。

#### 方法

**GetTriMeshFaceCount <node>**  
 返回一个包含两个整数的数组：表示 TriMesh 对象在修改器堆栈顶部的面数和顶点数。

### 8.4.3 Geometry-Standard Primitives (标准基本体)

#### 8.4.3.1 Box: GeometryClass (长方体)

##### 构造函数

Box ...

## 属性

属性名称	数据类型	默认值	说明
<Box>.length	Float	25.0	设置 Box 长度。可动画
<Box>.width	Float	25.0	设置 Box 宽度。可动画
<Box>.height	Float	25.0	设置 Box 高度。可动画
<Box>.lengthsegs <Box>.length_segments	Integer	1	设置 Box 长度方向的分段精度。可动画
<Box>.widthsegs <Box>.width_segments	Integer	1	设置 Box 宽度方向的分段精度。可动画
<Box>.heightsegs <Box>.height_segments	Integer	1	设置 Box 高度方向的分段精度。可动画
<Box>.mapcoords	Boolean	False	当设置为 On 时, 需要为 Box 对象的材质指定贴图坐标

## 8.4.3.2 Cone: GeometryClass(圆锥体)

## 构造函数

Cone ...

## 属性

属性名称	数据类型	默认值	说明
<Cone>.radius1 <Cone>.radius_1	Float	15.0	设置圆锥体的第一个半径。可动画
<Cone>.radius2 <Cone>.radius_2	Float	0.0	设置圆锥体的第二个半径。可动画
<Cone>.height	Float	25.0	设置圆锥体的高度。可动画
<Cone>.heightsegs <Cone>.height_Segments	Integer	5	设置圆锥体高度方向的分段精度。可动画
<Cone>.capsegs <Cone>.cap_segments	Integer	1	设置圆锥体端面的分段精度。可动画
<Cone>.sides	Integer	24	设置圆锥体端面圆周的分段精度。边数过少则生成棱锥。可动画
<Cone>.smooth	Boolean	True	当设置为 On 时创建圆台、圆锥; 否则, 创建棱锥、棱台
<Cone>.slice <Cone>.sliceon	Boolean	False	设置圆锥体是否进行切片操作
<Cone>.slice_On	Integer	0	属性.slice 的整数别名: 0: Off; 1: On
<Cone>.sliceFrom <Cone>.slice_From	Float	0.0	围绕局部坐标系的 Z 轴, 设置圆锥体切片操作的开始角度
<Cone>.sliceTo <Cone>.slice_To	Float	0.0	围绕局部坐标系的 Z 轴, 设置圆锥体切片操作的结束角度。可动画
<Cone>.mapCoords	Boolean	False	当设置为 On 时, 需要为圆锥体的材质指定贴图坐标

### 8.4.3.3 Cylinder: GeometryClass (圆柱体)

构造函数

Cylinder ...

属性

属性名称	数据类型	默认值	说明
<Cylinder>.radius	Float	15	设置圆柱体的半径。可动画
<Cylinder>.height	Float	25.0	设置圆柱体的高度。可动画
<Cylinder>.heightsegs <Cylinder>.height_segments	Integer	1	设置圆柱体高度方向的分段精度。可动画
<Cylinder>.capsegs <Cylinder>.cap_segments	Integer	1	设置圆柱体端面的分段精度。可动画
<Cylinder>.sides	Integer	24	设置圆柱体端面圆周的分段精度。边数过少则生成棱柱体。可动画
<Cylinder>.smooth	Boolean	True	当设置为 On 时，对柱体表面进行光滑处理
<Cylinder>.slice <Cylinder>.sliceon	Boolean	False	设置圆柱体是否进行切片操作
<Cylinder>.slice_On	Integer	0	属性.slice 的整数别名。可动画。0: Off; 1: On
<Cylinder>.sliceFrom	Float	0.0	围绕局部坐标系的 Z 轴，设置圆柱体切片操作的开始角度
<Cylinder>.sliceTo <Cylinder>.slice_To	Float	0.0	围绕局部坐标系的 Z 轴，设置圆柱体切片操作的结束角度。可动画
<Cylinder>.mapCoords	Boolean	False	当设置为 On 时，需要为圆柱体的材质指定贴图坐标

### 8.4.3.4 Geosphere: GeometryClass (经纬球)

构造函数

GeoSphere ...

属性

属性名称	数据类型	默认值	说明
<GeoSphere>.creationMethod <GeoSphere>.creation_Method	Integer	1	设置创建经纬球的方式： 0: Diameter (从边缘到边缘的方式拖动创建经纬球) 1: Center (从中心拖动创建经纬球)
<GeoSphere>.typeInPos	Point3	[0,0,0]	设置经纬球的位置
<GeoSphere>.typeInRadius	Float	25.0	设置经纬球的半径
<GeoSphere>.radius	Float	25.0	设置经纬球的半径。可动画
<GeoSphere>.segs <GeoSphere>.segments	Integer	4	设置经纬球表面三角结构面的数量。最大的数值 200 可以产生 800 000 个三角结构面。可动画

(续表)

属性名称	数据类型	默认值	说明
<GeoSphere>.baseType <GeoSphere>.base_Type	Integer	2	设置经纬球测量基准类型: 0: Tetra (表面基于四边形) 1: Octa (表面基于八边形) 2: Icosa (表面基于二十边形)
<GeoSphere>.smooth	Boolean	True	当设置为 On 时, 进行表面光滑处理。可动画
<GeoSphere>.hemisphere	Boolean	False	当设置为 On 时, 创建一个半球。可动画
<GeoSphere>.baseToPivot <GeoSphere>.base_To_Pivot	Boolean	False	设置是否沿着经纬球的局部坐标 Z 轴, 将经纬球中心移至底部
<GeoSphere>.mapCoords <GeoSphere>.Generate_mapping_coords	Boolean	False	当设置为 On 时, 需要为经纬球材质指定贴图坐标

#### 8.4.3.5 Plane: GeometryClass (平面)

##### 构造函数

Plane ...

meshGrid ...

##### 属性

属性名称	数据类型	默认值	说明
<Plane>.typeinCreationMethod <Plane>.creation_Method	Integer	0	设置平面的创建方式: 0: Rectangle (拖动鼠标从平面一个顶点到相对顶点创建长方体的平面) 1: Square (创建正方形的平面) 可动画
<Plane>.typeinPos	Point3	[0,0,0]	设置平面在 World 坐标系中的位置
<Plane>.typeinLength	Float	25.0	设置平面对象的长度
<Plane>.typeinWidth	Float	25.0	设置平面对象的宽度
<Plane>.length	Float	25.0	设置平面对象的长度。可动画
<Plane>.width	Float	25.0	设置平面对象的宽度。可动画
<Plane>.lsegs <Plane>.length_segments	Integer	4	设置平面对象长度方向的分段精度。可动画
<Plane>.wsegs <Plane>.width_segments	Integer	4	设置平面对象宽度方向的分段精度。可动画
<Plane>.density	Float	1.0	指定长度和宽度分段数在渲染时的倍增因子。可动画
<Plane>.renderScale <Plane>.render_Scale	Float	1.0	指定长度和宽度在渲染时的倍增因子。将从中心向外执行缩放。可动画
<Plane>.mapCoords <Plane>.mapping	Boolean	False	当设置为 On 时, 需要为平面对象材质指定贴图坐标

**注意** 属性.renderScale 为渲染时作用到属性.length 和.width 上的放大系数。属性.density 为渲染时作用到属性.lsegs 和.wsegs 上的放大系数。

### 8.4.3.6 Pyramid: GeometryClass (四棱锥)

构造函数

Pyramid ...

属性

属性名称	数据类型	默认值	说明
<Pyramid>.width	Float	25.0	设置四棱锥底面的宽度。可动画
<Pyramid>.depth	Float	25.0	设置四棱锥底面的深度。可动画
<Pyramid>.height	Float	25.0	设置四棱锥底面的高度。可动画
<Pyramid>.widthsegs <Pyramid>.width_segments	Integer	1	设置四棱锥底面宽度方向的分段精度。可动画
<Pyramid>.depthSegs <Pyramid>.depth_segments	Integer	1	设置四棱锥底面深度方向的分段精度。可动画
<Pyramid>.heightsegs <Pyramid>.height_segments	Integer	1	设置四棱锥底面高度方向的分段精度。可动画
<Pyramid>.mapCoords	Boolean	False	当设置为 On 时，需要为棱柱对象材质指定贴图坐标

### 8.4.3.7 Sphere: GeometryClass (球体)

构造函数

Sphere ...

属性

属性名称	数据类型	默认值	说明
<Sphere>.radius	Float	25.0	设置球体的半径。可动画
<Sphere>.segs <Sphere>.segments	Integer	16	设置球体表面的分段精度。可动画
<Sphere>.smooth	Boolean	True	设置是否进行光滑处理。可动画
<Sphere>.hemisphere	Float	0.0	使该值过大将从底部切断球体，以创建部分球体。取值范围：从 0.0 至 1.0。默认值是 0.0，可以生成完整的球体。设置为 0.5 可以生成半球，设置为 1.0 会使球体消失。可动画
<Sphere>.chop	Integer	0	设置是否使用切除。可动画
<Sphere>.slice	Boolean	False	设置是否进行切片操作。可动画
<Sphere>.sliceFrom	Float	0.0	围绕局部坐标系的 Z 轴，设置球体切片操作的开始角度。可动画
<Sphere>.sliceTo	Float	0.0	围绕局部坐标系的 Z 轴，设置球体切片操作的结束角度。可动画
<Sphere>.recenter	Boolean	False	当设置为 On 时，沿着球体的局部坐标系 Z 轴，将中心移动到球体的底部
<Sphere>.mapCoords	Boolean	False	当设置为 On 时，需要为球体对象材质指定贴图坐标

#### 8.4.3.8 Teapot: GeometryClass (茶壶)

构造函数

Teapot ...

属性

属性名称	数据类型	默认值	说明
<Teapot>.radius	Float	25.0	设置茶壶的半径。可动画
<Teapot>.segs <Teapot>.segments	Integer	4	设置茶壶表面的分段精度。可动画
<Teapot>.smooth	Boolean	True	设置是否进行光滑处理
<Teapot>.body	Boolean	True	设置茶壶部件 Body (壶身) 是否被创建。可动画
<Teapot>.handle	Boolean	True	设置茶壶部件 Handle (茶壶把手) 是否被创建。可动画
<Teapot>.spout	Boolean	True	设置茶壶部件 Spout (壶嘴) 是否被创建。可动画
<Teapot>.lid	Boolean	True	设置茶壶部件 Lid (壶盖) 是否被创建。可动画
<Teapot>.mapCoords	Boolean	False	当设置为 On 时, 需要为茶壶对象的材质指定贴图坐标

#### 8.4.3.9 Torus: GeometryClass (圆环)

构造函数

Torus ..

属性

属性名称	数据类型	默认值	说明
<Torus>.segs <Torus>.segments	Integer	24	设置圆环对象轴向上的分段精度, 数值过小会形成几何棱环。可动画
<Torus>.radius1 <Torus>.radius_1	Float	25.0	设置圆环对象轴向的半径 1。可动画
<Torus>.radius2 <Torus>.radius_2	Float	10.0	设置圆环对象的剖面半径 2。可动画
<Torus>.tubeRotation	Float	0.0	设置圆环对象剖面沿径向的旋转角度 (以度为单位), 旋转之后每一个圆环对象剖面的初始顶点位置发生错位。可动画
<Torus>.tubeTwist <Torus>.twist	Float	0.0	设置圆环对象扭曲的角度。可动画
<Torus>.sides	Integer	12	设置圆环对象剖面的边数。可动画
<Torus>.smooth	Integer	0	设置圆环对象的光滑方式: 0: None (不对圆环对象进行光滑处理) 1: Sides (对圆环对象的剖面方向进行光滑处理) 2: All (对圆环对象的全部表面进行光滑处理) 3: Segments (对圆环对象分段方向上进行光滑处理) 可动画

(续表)

属性名称	数据类型	默认值	说明
<Torus>.slice	Boolean	False	设置是否进行切片操作。可动画
<Torus>.sliceFrom <Torus>.slice_From	Float	0.0	设置圆环对象切片操作的开始角度。可动画
<Torus>.sliceTo <Torus>.slice_To	Float	0.0	设置圆环对象切片操作的结束角度。可动画
<Torus>.mapCoords	Boolean	False	当设置为 On 时，需要为圆环对象材质指定贴图坐标

#### 8.4.3.10 Tube: GeometryClass (圆管)

构造函数

Tube ...

属性

属性名称	数据类型	默认值	说明
<Tube>.radius1 <Tube>.radius_1	Float	25.0	设置圆管的底面半径 1。可动画
<Tube>.radius2 <Tube>.radius_2	Float	20.0	设置圆管的底面半径。可动画
<Tube>.height	Float	50.0	设置圆管的高度。可动画
<Tube>.heightsegs <Tube>.height_segments	Integer	1	设置圆管沿高度方向的分段精度。可动画
<Tube>.capsegs <Tube>.cap_segments	Integer	1	设置圆管端面的分段精度。可动画
<Tube>.sides	Integer	24	设置圆管端面圆周的分段精度，边数过小则生成棱状圆管。可动画
<Tube>.smooth	Boolean	True	设置是否进行光滑处理。可动画
<Tube>.slice	Boolean	False	设置是否进行切片操作。可动画
<Tube>.slice_On	Integer	0	设置是否进行切片操作。可动画 0: Off; 1: On
<Tube>.sliceFrom <Tube>.slice_From	Float	0.0	设置圆管切片操作的开始角度。可动画
<Tube>.sliceTo <Tube>.slice_To	Float	0.0	设置圆管切片操作的结束角度。可动画
<Tube>.mapCoords	Boolean	False	当设置为 On 时，需要为圆管对象材质指定贴图坐标

#### 8.4.4 Geometry-Extended Primitives (扩展基本体)

##### 8.4.4.1 C\_Ext: GeometryClass (C 形墙)

构造函数

C\_Ext ...

## 属性

属性名称	数据类型	默认值	说明
<C_Ext>.back_length	Float	0.0	设置 C 形墙底面后边的长度。可动画
<C_Ext>.side_length	Float	0.0	设置 C 形墙底面侧边的长度。可动画
<C_Ext>.front_length	Float	0.0	设置 C 形墙底面前边的长度。可动画
<C_Ext>.back_width	Float	0.0	设置 C 形墙底面后边的厚度。可动画
<C_Ext>.side_width	Float	0.0	设置 C 形墙底面侧边的厚度。可动画
<C_Ext>.front_width	Float	0.0	设置 C 形墙底面前边的厚度。可动画
<C_Ext>.height	Float	0.0	设置 C 形墙高度。可动画
<C_Ext>.back_segments	Integer	1	设置 C 形墙底面后边的分段精度。可动画
<C_Ext>.side_segments	Integer	1	设置 C 形墙底面侧边的分段精度。可动画
<C_Ext>.front_segments	Integer	1	设置 C 形墙底面前边的分段精度。可动画
<C_Ext>.width_segments	Integer	1	设置 C 形墙宽度方向的分段精度。可动画
<C_Ext>.height_segment	Integer	1	设置 C 形墙高度方向的分段精度。可动画
<C_Ext>.mapCoords	Integer	1	设置是否为 C 形墙指定贴图坐标: 0: 不指定贴图坐标; 1: 指定贴图坐标

## 8.4.4.2 Capsule: GeometryClass (胶囊)

## 构造函数

Capsule ...

## 属性

属性名称	数据类型	默认值	说明
<Capsule>.radius	Float	15.0	设置胶囊的底面半径, 可动画
<Capsule>.height	Float	25.0	设置胶囊的高度, 可动画
<Capsule>.heighttype	Integer	1	设置胶囊的高度设定类型: 0: Overall (胶囊的高度设定包含两个端面的高度) 1: Centers (胶囊的高度设定不包含两个端面的高度)
<Capsule>.sides	Integer	24	设置胶囊端面圆周的分段精度。边数过小则生成棱状 胶囊。可动画
<Capsule>.heightsegs <Capsule>.height_ segments	Integer	1	设置胶囊高度方向的分段精度。可动画
<Capsule>.smooth	Boolean	True	设置胶囊表面是否进行光滑处理
<Capsule>.smooth_on	Integer	1	Smooth_on 为属性.smooth 的整数别名。可动画 0: Off; 1: On
<Capsule>.sliceon	Boolean	False	设置胶囊是否进行切片操作
<Capsule>.slice_on	Integer	0	Slice_on 为属性.sliceon 的整数别名。可动画 0: Off; 1: On

(续表)

属性名称	数据类型	默认值	说明
<Capsule>.slicefrom <Capsule>.slice_from	Float	0.0	围绕局部坐标系的 Z 轴, 设置胶囊切片操作的开始角度。可动画
<Capsule>.sliceto <Capsule>.slice_to	Float	0.0	围绕局部坐标系的 Z 轴, 设置胶囊切片操作的结束角度。可动画
<Capsule>.mapCoords	Boolean	False	当设置为 On 时, 需要为胶囊的材质指定贴图坐标

#### 8.4.4.3 ChamferBox: GeometryClass (倒角长方体)

构造函数

ChamferBox ...

属性

属性名称	数据类型	默认值	说明
<ChamferBox>.length	Float	0.1	设置倒角长方体的长度。可动画
<ChamferBox>.width	Float	0.1	设置倒角长方体的宽度。可动画
<ChamferBox>.height	Float	0.1	设置倒角长方体的高度。可动画
<ChamferBox>.fillet	Float	0.01	设置倒角长方体倒圆的半径大小。可动画
<ChamferBox>.length_segments	Integer	1	设置倒角长方体长度方向的分段精度。可动画
<ChamferBox>.width_segments	Integer	1	设置倒角长方体宽度方向的分段精度。可动画
<ChamferBox>.height_segments	Integer	1	设置倒角长方体高度方向的分段精度。可动画
<ChamferBox>.fillet_segments	Integer	3	设置倒角长方体倒圆的分段精度。可动画
<ChamferBox>.mapCoords	Integer	1	设置是否指定贴图坐标: 1: 指定贴图坐标; 0: 不指定贴图坐标

#### 8.4.4.4 ChamferCyl: GeometryClass (倒角圆柱体)

构造函数

ChamferCyl ...

属性

属性名称	数据类型	默认值	说明
<ChamferCyl>.radius	Float	0.0	设置倒角圆柱体的底面半径。可动画
<ChamferCyl>.height	Float	0.0	设置倒角圆柱体的高度。可动画
<ChamferCyl>.fillet	Float	0.0	设置倒角圆柱体倒圆的半径大小。可动画
<ChamferCyl>.height_segments	Integer	1	设置倒角圆柱体高度方向的分段精度。可动画
<ChamferCyl>.fillet_segments	Integer	1	设置倒角圆柱体倒圆的分段精度。可动画
<ChamferCyl>.sides	Integer	12	设置倒角圆柱体端面圆周的分段精度。边数过少则生成球棱柱。可动画
<ChamferCyl>.cap_segments	Integer	1	设置倒角圆柱体端面的分段精度。可动画

(续表)

属性名称	数据类型	默认值	说明
<ChamferCyl>.smooth	Boolean	True	设置倒角圆柱体是否进行光滑处理
<ChamferCyl>.smooth_On	Integer	1	.smooth 属性的整型别名。可动画 0: Off; 1: On
<ChamferCyl>.sliceOn	Boolean	False	设置倒角圆柱体是否进行切片操作。可动画
<ChamferCyl>.slice_From	Float	0.0	围绕局部坐标系的 Z 轴, 设置倒角圆柱体切片操作的开始角度。可动画
<ChamferCyl>.slice_To	Float	0.0	围绕局部坐标系的 Z 轴, 设置倒角圆柱体切片操作的结束角度。可动画
<ChamferCyl>.mapCoords	Integer	1	设置是否指定贴图坐标: 1: 指定贴图坐标; 0: 不指定贴图坐标

#### 8.4.4.5 Gengon: GeometryClass (球棱柱)

构造函数

Gengon ...

属性

属性名称	数据类型	默认值	说明
<Gengon>.sides	Integer	5	设置球棱柱的底面边数, 即棱的数量。可动画
<Gengon>.radius	Float	0.0	设置球棱柱的半径。可动画
<Gengon>.fillet	Float	0.0	设置球棱柱倒圆的半径大小。可动画
<Gengon>.height	Float	0.0	设置球棱柱的高度。可动画
<Gengon>.side_segments	Integer	1	设置球棱柱底面边的分段精度。可动画
<Gengon>.height_segments	Integer	1	设置球棱柱高度方向的分段精度。可动画
<Gengon>.fillet_segments	Integer	1	设置球棱柱倒圆的分段精度。可动画
<Gengon>.smooth	Integer	0	设置是否进行光滑处理。 1: 进行光滑处理; 0: 不进行光滑处理
<Gengon>.mapCoords	Integer	1	设置是否指定贴图坐标: 1: 指定贴图坐标; 0: 不指定贴图坐标

#### 8.4.4.6 Hedra: GeometryClass (多面体)

构造函数

Hedra ...

属性

属性名称	数据类型	默认值	说明
<Hedra>.family	Integer	0	设置多面体的创建类型: 0: Tetra (四面体) 1: Cube/Octa (立方体或八面体) 2: Dodec/Ico (十二面体或二十四面体) 3: Star1 (星状体 1) 4: Star2 (星状体 2)
<Hedra>.p <Hedra>.p_Value	Float	0.0	调整多面体的顶点和面。取值范围在 0.0 到 1.0 之间。两个参数相互关联，其和必须小于或等于 1.0。如果将 P 或 Q 任一个设为最大值 1.0，另一个将自动设为 0。可动画
<Hedra>.q <Hedra>.q_Value	Float	0.0	同上。可动画
<Hedra>.scalep <Hedra>.p_scale	Float	1.0	设置多面体 P 面轴向比率。可动画
<Hedra>.scaler <Hedra>.r_scale	Float	1.0	设置多面体 R 面轴向比率。可动画
<Hedra>.scaleq <Hedra>.q_Scale	Float	1.0	设置多面体 Q 面轴向比率。可动画
<Hedra>.vertices <Hedra>.vertType	Float	0	设置多面体每个面的顶点创建方式: 0: Basic (面的细分不能超过最小值) 1: Center (每个表面都依据面中心附加的顶点进行细分，以面中点到面顶点的连线进行细分) 2: Center & Sides (每个表面都依据面中心附加的顶点进行细分，以面中点到面边线中点的连线进行细分)
<Hedra>.radius	Float	25.0	设置多面体大小。可动画
<Hedra>.mapCoords	Boolean	False	当设置为 On 时，需要为多面体材质指定贴图坐标

**注意** 缩放类参数集 (scalep、scaleq、scaler) 和 (p\_scale、q\_scale、r\_scale) 互为别名，但是第一个参数集存储格式为分数，而第二个参数集存储格式为百分数。在命令面板和 Track View 里显示的格式与第二个参数集相同。

为了存取 hedra 对象的.vertices 属性，必须通过属性.baseobject，如：

```
MyHedra.baseobject.vertices=1 --设置顶点创建方式为 center
```

#### 8.4.4.7 L\_Ext: GeometryClass (L 形墙)

构造函数

L\_Ext ...

属性

属性名称	数据类型	默认值	说明
<L_Ext>.side_length	Float	0.0	设置 L 形墙的底面侧边长度。可动画
<L_Ext>.front_length	Float	0.0	设置 L 形墙的底面前边长度。可动画
<L_Ext>.side_width	Float	0.0	设置 L 形墙的底面侧边厚度。可动画
<L_Ext>.front_width	Float	0.0	设置 L 形墙的底面前边厚度。可动画
<L_Ext>.height	Integer	1	设置 L 形墙的高度。可动画
<L_Ext>.side_segments	Integer	1	设置 L 形墙的底面侧边长度方向的分段精度。可动画
<L_Ext>.front_segments	Integer	1	设置 L 形墙的底面前边长度方向的分段精度。可动画
<L_Ext>.width_segments	Integer	1	设置 L 形墙的底面前边宽度方向的分段精度。可动画
<L_Ext>.height_segments	Float	0.0	设置 L 形墙的底面前边高宽度方向的分段精度。可动画
<L_Ext>.mapCoords	Integer	1	设置是否指定贴图坐标: 1: 指定贴图坐标; 0: 不指定贴图坐标

#### 8.4.4.8 OilTank: GeometryClass (油罐)

构造函数

OilTank...

属性

属性名称	数据类型	默认值	说明
<OilTank>.radius	Float	0.0	设置油罐的底面半径。可动画
<OilTank>.height	Float	0.0	设置油罐的高度。可动画
<OilTank>.cap_Height	Float	0.0	设置油罐的端面隆起量。最小值是半径的 2.5%; 最大值是半径的两倍。可动画
<OilTank>.blend	Float	0.0	设置油罐的端面与侧面的融合量, 如果设定为 0, 端面与侧面间是不融合的斜角。可动画
<OilTank>.sides	Integer	12	设置油罐圆周的分段精度, 边数过小则生成球棱柱。可动画
<OilTank>.height_segments	Integer	1	设置油罐高度方向的分段精度。可动画
<OilTank>.smooth_On	Integer	1	设置是否进行光滑处理。可动画 0: Off; 1: On
<OilTank>.slice_On	Integer	0	设置是否进行切片操作。可动画 0: Off ; 1: On
<OilTank>.slice_From	Float	0.0	围绕局部坐标系的 Z 轴, 设置油罐切片操作的开始角度。可动画
<OilTank>.slice_To	Float	0.0	围绕局部坐标系的 Z 轴, 设置油罐切片操作的结束角度。可动画
<OilTank>.mapCoords	Integer	1	设置是否指定贴图坐标: 1: 指定贴图坐标; 0: 不指定贴图坐标

#### 8.4.4.9 Prism: GeometryClass (棱柱)

构造函数

Prism ...

#### 属性

属性名称	数据类型	默认值	说明
<Prism>.side1Length <Prism>.side_1_Length	Float	25.0	设置棱柱底面三角形第一条边的长度。可动画
<Prism>.side2Length <Prism>.side_2_Length	Float	25.0	设置棱柱底面三角形第二条边的长度。可动画
<Prism>.side3Length <Prism>.side_3_Length	Float	25.0	设置棱柱底面三角形第三条边的长度。可动画
<Prism>.height	Float	10.0	设置棱柱对象的高度
<Prism>.side1Segs <Prism>.side_1_Segments	Integer	1	设置棱柱底面三角形第一条边的分段精度。可动画
<Prism>.side2Segs <Prism>.side_2_Segments	Integer	1	设置棱柱底面三角形第二条边的分段精度。可动画
<Prism>.side3Segs <Prism>.side_3_Segments	Integer	1	设置棱柱底面三角形第三条边的分段精度。可动画
<Prism>.heightsegs <Prism>.height_Segments	Integer	1	设置棱柱对象高度方向的分段精度。可动画
<Prism>.mapCoords	Boolean	False	当设置为 On 时, 需要为棱柱对象材质指定贴图坐标

#### 8.4.4.10 RingWave: GeometryClass (环形波)

##### 构造函数

RingWave ...

#### 属性

属性名称	数据类型	默认值	说明
<RingWave>.'max diameter' <RingWave>.radius	Float	500.0	设置环形波外环的半径。可动画
<RingWave>.'Radius Segs' <RingWave>.radial_Segments	Integer	1	设置环形波外环半径与内环半径之间, 沿半径方向的分段精度。可动画
<RingWave>.'ring width' <RingWave>.ring_Width	Float	1.0	设置环形波外环半径与内环半径之间的宽度。可动画
<RingWave>.'ring segments' <RingWave>.sides	Integer	200	设置环形波外环、内环与端面的边数。可动画
<RingWave>.height	Float	0.0	设置环形波的高度。可动画
<RingWave>.'height Segs' <RingWave>.height_Segments	Integer	1	设置环形波高度方向的分段精度。可动画
<RingWave>.repeat	Integer	2	设置环形波的动画方式: 0: Cyclic Growth (循环增长) 1: Grow and Stay (增长并终止) 2: No Growth (不增长)
<RingWave>.'time on' <RingWave>.start_Time	Integer	0	设置环形波增长动画的开始时间点

(续表)

属性名称	数据类型	默认值	说明
<RingWave>.'time growing' <RingWave>.grow_Time	Integer	9600	设置环形波从最小尺寸变为最大尺寸所用的时间
<RingWave>.'display until' <RingWave>.end_Time	Integer	16000	设置环形波增长动画的结束时间点
<RingWave>.'outer Edge Breakup' <RingWave>.outer_Edge_Breakup	Boolean	False	设置环形波外形锯齿是否开启
<RingWave>.'Major Cycles Outer' <RingWave>.outer_Edge_Major_Cycles	Integer	1	设置环形波外环上的主波数量
<RingWave>. 'Major Cycle Flux Outer' <RingWave>.outer_Edge_Major_Flux	Float	0.0	设置环形波外环主波的振幅，该数值依据环形波宽度的百分数值进行设定
<RingWave>. 'Major Cycle Flux Per Outer' <RingWave>.outer_Edge_Major_Crawl_Time	Integer	16000	设置主波沿环形波外环波动的帧数
<RingWave>. 'Minor Cycles Outer' <RingWave>.outer_Edge_Minor_Cycles	Integer	1	设置每个环形波上次级波的数量
<RingWave>. 'Minor Cycle Flux Outer' <RingWave>.outer_Edge_Minor_Flux	Float	0.0	设置环形波主波上次级波的振幅，该数值依据环形波宽度的百分数值进行设定
<RingWave>. 'Minor Cycle Flux Per Outer' <RingWave>.outer_Edge_Minor_Crawl_Time	Integer	16000	设置次级波沿环形波外环波动的帧数
<RingWave>.'Inner Edge Breakup' <RingWave>.inner_Edge_Breakup	Boolean	True	设置环形波内环锯齿是否开启
<RingWave>.'Major Cycles Inner' <RingWave>.inner_Edge_Major_Cycles	Integer	11	设置环形波内环上的主波数量
<RingWave>. 'Major Cycle Flux Inner' <RingWave>.inner_Edge_Major_Flux	Float	25.0	设置环形波内环主波的振幅，该数值依据环形波宽度的百分数值进行设定
<RingWave>. 'Major Cycle Flux Per Inner' <RingWave>.inner_Edge_Major_Crawl_Time	Integer	19360	设置主波沿环形波内环波动的帧数

(续表)

属性名称	数据类型	默认值	说明
<RingWave>. 'Minor Cycles Inner' <RingWave>. inner_Edge_Minor_Cycles	Integer	29	设置每个环形波内环上次级波的数量
<RingWave>. 'Minor Cycle Flux Inner' <RingWave>. inner_Edge_Minor_Flux	Float	0.0	设置环形波主波内环上次级波的振幅，该数值依据环形波宽度的百分数值进行设定
<RingWave>. 'Minor Cycle Flux Per Inner' <RingWave>. inner_Edge_Minor_Crawl_Time	Integer	-4320	设置次级波沿环形波内环波动的帧数
<RingWave>.repeats	Integer	2	设置环形波重复周期数
<RingWave>.'Mapping Coords'	Boolean	False	当设置为 On 时，需要为环形波对象材质指定贴图坐标
<RingWave>.smoothing	Boolean	False	设置是否进行光滑处理

**注意** 有些属性可以用多种格式输入，无论用何种格式，它们都互相联系。

#### 8.4.4.11 Spindle: GeometryClass (纺锤体)

构造函数

Spindle ...

属性

属性名称	数据类型	默认值	说明
<Spindle>.radius	Float	0.0	设置纺锤体的半径。可动画
<Spindle>.height	Float	0.0	设置纺锤体的高度。可动画
<Spindle>.Cap_Height	Float	0.0	设置纺锤体的端面尖顶高度。最小值是 0.1，最大值依据底面半径的绝对值设置。可动画
<Spindle>.blend	Float	0.0	设置纺锤体的端面与侧面的融合量。如果设定为 0，端面与侧面间是不融合的斜角。可动画
<Spindle>.sides	Integer	12	设置纺锤体圆周的分段精度。边数过小则生成纺锤体。可动画
<Spindle>.cap_Segments	Integer	5	设置纺锤体端面的分段精度。可动画
<Spindle>.height_Segments	Integer	5	设置纺锤体高度方向的分段精度。可动画
<Spindle>.smooth_On	Integer	1	设置是否进行光滑处理。可动画 0: Off; 1: On
<Spindle>.slice_On	Integer	0	设置是否进行切片操作。可动画 0: Off; 1: On
<Spindle>.slice_From	Float	0.0	围绕局部坐标系的 Z 轴，设置纺锤体切片操作的开始角度。可动画

(续表)

属性名称	数据类型	默认值	说明
<Spindle>.slice_To	Float	0.0	围绕局部坐标系的Z轴,设置纺锤体切片操作的结束角度。可动画
<Spindle>.mapCoords	Integer	1	设置是否指定贴图坐标: 1: 指定贴图坐标; 0: 不指定贴图坐标

#### 8.4.4.12 TargetObject: GeometryClass (目标对象)

本类对象为 Camera、Spotlight 等类型的对象指定目标对象。在 MAXScript 里,如果这些对象需要一个目标对象,必须显式地为其构造,如:

```
c = targetCamera pos:[x,y,z] target:(targetObject pos:[xt, yt, zt])
```

构造函数

TargetObject ...

属性

除通用属性外, TargetObject 类对象没有额外的属性。

#### 8.4.4.13 Torus\_Knot: GeometryClass (环形结)

构造函数

Torus\_Knot ...

属性

属性名称	数据类型	默认值	说明
<Torus_Knot>.base_Curve	Integer	0	设置环形结基准曲线类型: 0: Knot (基准曲线可以由各个参数控制其复杂的空间形态) 1: Circle (圆形类型的基准曲线是平面曲线)
<Torus_Knot>.radius	Float	0.0	设置环形结基准曲线的大小。可动画
<Torus_Knot>.segments	Integer	120	设置环形结基准曲线的分段精度。可动画
<Torus_Knot>.p	Float	2.0	设置环形结基准曲线 P-顶的数值,该数值影响环形结基准曲线的缠绕方式。可动画
<Torus_Knot>.q	Float	3.0	设置环形结基准曲线 Q-底的数值,该数值影响环形结基准曲线的缠绕方式。可动画
<Torus_Knot>.warp_Count	Float	0.0	设置环形结基准曲线上的点数。可动画
<Torus_Knot>.warp_Height	Float	0.0	设置环形结基准曲线上星形点的突起量。可动画
<Torus_Knot>.radius2	Float	10.0	设置环形结对象剖面的大小。可动画
<Torus_Knot>.sides	Integer	12	设置环形结对象剖面的边数。可动画
<Torus_Knot>.eccentricity	Float	1.0	设置环形结对象剖面的偏心率。可动画

(续表)

属性名称	数据类型	默认值	说明
<Torus_Knot>.twist	Float	0.0	设置环形结对象剖面围绕基准曲线的扭曲量。可动画
<Torus_Knot>.lumps	Float	0.0	设置环形结对象剖面囊块 Lumps 的数量。可动画
<Torus_Knot>.lump_Height	Float	0.0	设置环形结对象剖面囊块 Lumps 的隆起高度。可动画
<Torus_Knot>.lump_Offset	Float	0.0	设置环形结对象剖面囊块 Lumps 的沿环形结的偏移量。可动画
<Torus_Knot>.smooth	Integer	2	设置环形结对象的光滑方式： 0: None (对环形结对象不进行光滑处理) 1: Sides (对沿基准曲线的环形结表面进行光滑处理) 2: All (对环形结对象的全部表面进行光滑处理)
<Torus_Knot>.gen_UV	Integer	0	当设置为 On 时，为环形结对象材质指定贴图坐标。 0: Off; 1: On
<Torus_Knot>.u_Tile	Float	1.0	设置环形结对象贴图沿 U 方向的重复次数。可动画
<Torus_Knot>.v_Tile	Float	1.0	设置环形结对象贴图沿 V 方向的重复次数。可动画
<Torus_Knot>.u_Offset	Float	0.0	设置环形结对象贴图沿 U 方向的偏移量。可动画
<Torus_Knot>.v_Offset	Float	0.0	设置环形结对象贴图沿 V 方向的偏移量。可动画

## 8.4.5 Geometry-Dynamics Objects (动力学对象)

动力学对象有下面两种：

Damper

Spring

### 8.4.5.1 Damper: GeometryClass (阻尼器)

构造函数

Damper ...

属性

属性名称	数据类型	默认值	说明
<Damper>.End_Placement_Method	Integer	1	设置阻尼器端点绑定方式： 0: Reference Objects (将 Damper 的两个端点分别绑定到其他对象的轴心点上) 1: Free Spring (指定 Damper 为一个独立的动力学对象，不绑定到其他对象，也不会被用于动力学模拟器中)
<Damper>.Free_Damper_Height	Float	2.0	指定未绑定阻尼器时，底座底部中心到活塞顶部中心之间的距离。可动画
<Damper>.Renderable_Spring	Integer	1	设置阻尼器是否最终被渲染输出

(续表)

属性名称	数据类型	默认值	说明
<Damper>.Generate_Mapping_Coordinates	Integer	0	设置阻尼器是否指定默认的贴图坐标。 0: Off; 1: On
<Damper>.Base_Stud_Diameter	Float	0.5	设置阻尼器基准构件直径。可动画
<Damper>.Ase_Stud_Height	Float	0.2	设置阻尼器基准构件的高度。可动画
<Damper>.Cylinder_Diameter	Float	1.0	设置阻尼器活塞筒构件的直径。可动画
<Damper>.Cylinder_Height	Float	1.0	设置阻尼器活塞筒构件的高度。可动画
<Damper>.Cylinder_Sides	Integer	8	设置阻尼器活塞筒构件和基准构件的边数。可动画
<Damper>.Cylinder_Fillet_1	Float	0.0	设置阻尼器活塞筒构件底边倒角的大小。可动画
<Damper>.Cylinder_Fillet_1_Segments	Integer	0.0	设置阻尼器活塞筒构件底边倒角的分段精度。可动画
<Damper>.Cylinder_Fillet_2	Float	0.0	设置阻尼器活塞筒构件顶边倒角的大小。可动画
<Damper>.Cylinder_Fillet_2_Segments	Integer	0.0	设置阻尼器活塞筒构件顶边倒角的分段精度。可动画
<Damper>.Inside_Diameter	Float	0.0	设置阻尼器活塞筒构件的内部直径。可动画
<Damper>.Smooth_Cylinder	Integer	1	设置阻尼器活塞筒构件和基准构件是否进行光滑处理。0: Off; 1: On
<Damper>.Piston_Diameter	Float	0.2	设置阻尼器活塞构件的直径。可动画
<Damper>.Piston_Height	Float	1.0	设置阻尼器活塞构件的高度。可动画
<Damper>.Piston_Sides	Integer	6	设置阻尼器活塞构件的边数。可动画
<Damper>.Smooth_Piston	Integer	1	设置阻尼器活塞构件是否进行光滑处理: 0: Off ; 1: On
<Damper>.Enable_Boot	Integer	0	设置是否为阻尼器加入附加构件胶垫: 0: Off; 1: On
<Damper>.Boot_Small_Diameter	Float	0.25	设置阻尼器可折叠胶垫的最小直径。可动画
<Damper>.Boot_Large_Diameter	Float	1.0	设置阻尼器可折叠胶垫的最大直径。可动画
<Damper>.Boot_Sides	Integer	8	设置阻尼器胶垫对象的边数。可动画
<Damper>.Boot_Folds	Integer	4	设置阻尼器胶垫对象的折叠数。可动画
<Damper>.Boot_Fold_Resolution	Integer	4	设置阻尼器胶垫构件的分段精度。可动画
<Damper>.Boot_Stop_Diameter	Float	0.4	设置阻尼器胶垫顶部止动环的直径。可动画
<Damper>.Boot_Stop_Height	Float	0.2	设置阻尼器胶垫顶部止动环的高度。可动画
<Damper>.Boot_Stop_Setback	Float	0.2	设置阻尼器止动环与活塞顶部之间的距离。可动画

(续表)

属性名称	数据类型	默认值	说明
<Damper>.Boot_Stop_Fillet	Float	0.0	设置阻尼器止动环倒角的大小。可动画
<Damper>.Boot_Stop_Fillet_Segements	Integer	1	设置阻尼器止动环倒角的分段精度。可动画
<Damper>.Smooth_Boot	Integer	1	设置是否对阻尼器胶垫构件进行自动光滑处理: 0: Off; 1: On
<Damper>.Dynamics_Object_Type	Integer	0	指定哪个对象是动态的: 0: Damper (阻尼器); 1: Actuator (加速器)
<Damper>.Drag	Float	0.0	设置阻尼器在每个直线速率上表现出的反作用力。可动画
<Damper>.Drag_Units	Integer	0	设置阻尼器阻力计量单位: 0: Pounds per inch/Second (磅每英寸/秒) 1: Newtons per meter/second (牛顿每米/秒)
<Damper>.Damper_Direction	Integer	2	设置阻尼器工作方式: 0: Compression Only (只对压缩力产生反作用) 1: Extension Only (只对挤出力产生反作用) 2: Both (同时对压缩力和挤出力产生反作用)
<Damper>.Force	Float	0.0	设置阻尼器施加到两个绑定对象上的作用力大小, 正值使两个对象分开; 负值使两个对象靠近。可动画
<Damper>.Force_Units	Integer	0	设置阻尼器作用力计量单位: 0: Pounds (磅/英寸); 1: Newtons (牛顿/米)

#### 8.4.5.2 Spring: GeometryClass (弹簧)

构造函数

Spring ...

属性

属性名称	数据类型	默认值	说明
<Spring>.End_Placement_Method	Integer	1	设置弹簧端点绑定方式: 0: Reference Objects (将两个端点分别绑定到其他对象的轴心点上); 1: Free Spring (指定 Spring 为一个独立的动力学对象, 不绑定到其他对象, 也不会被用于动力学模拟器中)
<Spring>.Free_Spring_Height	Float	1.0	设置弹簧未受力情况下在垂直方向上的高度。可动画
<Spring>.Diameter	Float	1.0	设置弹簧外轮廓的直径。可动画
<Spring>.Number_of_Turns	Float	1.0	设置弹簧环绕的圈数。可动画
<Spring>.Turn_Direction	Integer	0	设置弹簧是顺时针方向还是逆时针方向环绕: 0: 逆时针方向; 1: 顺时针方向

(续表)

属性名称	数据类型	默认值	说明
<Spring>.Segmentation_Method	Integer	0	设置弹簧的分段方式: 0: Automatic Segments (自动分段, 弹簧的每一圈具有相同的分段精度) 1: Manual Segments (手动分段, 为弹簧对象指定一个固定的分段精度, 如果增加了弹簧圈数, 分段精度不会自动增加, 只能重新指定新的分段数)
<Spring>.Segments_Per_Turn	Integer	16	设置弹簧环绕 360 度的分段精度。可动画
<Spring>.Segments_Along_Turn	Integer	16	为手动分段方式设置弹簧的分段整数。可动画
<Spring>.Smooth_Spring	Integer	0	设置弹簧的光滑方式: 0: None (未应用光滑) 1: Sides (沿着线框的长度而不是绕着其周界进行光滑) 2: All (光滑全部曲面) 3: Segments (绕着线框的周界而不是沿着其长度进行光滑)
<Spring>.Wire	Integer	0	设置弹簧的线框横截面: 0: Round (圆形线框) 1: Rectangular (长方形线框) 2: D-Section (D 截面线框) 可动画
<Spring>.Generate_Mapping_Coordinates	Integer	0	设置弹簧是否指定默认的贴图坐标: 0: Off; 1: On
<Spring>.Round_Wire_Diameter	Float	0.2	设置弹簧圆形线框横截面的直径。可动画
<Spring>.Round_Wire_Side_Count	Integer	6	设置弹簧圆形线框横截面的边数。可动画
<Spring>.Rectangular_Wire_Width	Float	0.2	设置弹簧长方形线框横截面的宽度。可动画
<Spring>.Rectangular_Wire_Depth	Float	0.2	设置弹簧长方形线框横截面的深度。可动画
<Spring>.Rectangular_Wire_Fillet_Size	Float	0.0	设置弹簧长方形线框横截面的倒角量。可动画
<Spring>.Rectangular_Fillet_Sides	Integer	0	设置弹簧长方形线框横截面的倒角的分段精度。可动画
<Spring>.Rectangular_Wire_Rotation_Angle	Float	0.0	设置弹簧长方形线框横截面沿弹簧线方向的旋转角度。可动画
<Spring>.D_Section_Wire_Width	Float	0.2	设置弹簧 D 截面线框的宽度。可动画
<Spring>.D_Section_Wire_Depth	Float	0.2	设置弹簧 D 截面线框的深度。可动画
<Spring>.D_Section_Round_Sides	Integer	4	设置弹簧 D 截面线框圆边的边数
<Spring>.D_Section_Wire_Fillet_Size	Float	0.0	设置弹簧 D 截面线框的倒角量。可动画

(续表)

属性名称	数据类型	默认值	说明
<Spring>.D_Section_Wire_Fillet_Sides	Integer	0	设置弹簧 D 截面线框倒角的分段精度。可动画
<Spring>.D_Section_Wire_Rotation_Angle	Float	0.0	设置弹簧 D 截面线框沿弹簧线方向的旋转角度。可动画
<Spring>.Dynamics_Spring_Free_Height	Float	1.0	设置弹簧未受外力作用的情况下松弛高度。可动画
<Spring>.Dynamics_K_Constant_Value	Float	1.0	设置弹簧在松弛高度的基础上，弹簧每挤出或压缩一个单位长度所表现出的反弹力（单位：磅/英寸）。可动画
<Spring>.Dynamics_K_Constant_Units	Integer	0	设置弹簧常量单位： 0: Pounds per inch (磅/英寸) 1: Newtons per meter (牛顿/米)
<Spring>.Dynamics_Spring_Direction	Integer	0	设置弹簧弹力工作方式： 0: Compression Only (仅当压缩弹簧使其高度小于松弛高度时，表现压缩力) 1: Extension Only (仅当挤出弹簧使其高度大于松弛高度时，表现扩张力) 2: Both (当压缩弹簧使其高度小于松弛高度时，表现压缩力；当挤出弹簧使其高度大于松弛高度时，表现扩张力)
<Spring>.Generate_Texture_Coordinates	Integer	0	设置弹簧是否指定默认的贴图坐标。 0: Off; 1: On

#### 8.4.6 Geometry-Compound Objects (复合对象)

下面给出了所有 Compound 类对象：

OldBoolean (旧版布尔复合对象)

Boolean2 (布尔复合对象)

Conform (一致复合对象)

Connect (连接复合对象)

Loft (放样复合对象)

Morph (变形复合对象)

Scatter (离散复合对象)

ShapeMerge (图形合并复合对象)

Terrain (地形复合对象)

##### 8.4.6.1 OldBoolean: GeometryClass (旧版布尔复合对象)

在 MAXScript 里不能直接构造 OldBoolean 类复合对象。装载由 3ds max 以前版本创建的 Boolean 对象时系统自动将它们视为 OldBoolean 类对象，并自动把它们转换为新的 Boolean2 对象。

#### 8.4.6.2 Boolean2: GeometryClass (布尔复合对象)

构造函数

```
boolObj.createBooleanObject <operand_A> \
    [<operand_B> <add_method> <mat_method>]
```

由 Node 对象<operand\_A>和可选参数<operand\_B>创建一个 Boolean2 类对象。

参数<add\_method>指定如何使用<operand\_B>:

- 1: instance, 操作对象为源 Node 对象的实例 (instance)
- 2: reference, 操作对象为源 Node 对象的参照 (reference)
- 3: copy, 操作对象为源 Node 对象的副本 (copy)
- 4: move, 删除源 Node 对象

参数<mat\_method> 指定如何处理两个操作对象的材质:

- 1: 不改变原来的材质和 ID
- 2: 用材质 ID 码匹配材质, 然后组合材质
- 3: 用材质匹配材质 ID 码, 然后组合材质
- 4: 放弃原有材质, 使用新生成对象的材质
- 5: 放弃新生成对象的材质, 使用原有材质

属性

下面的属性仅在 Boolean2 对象被创建后才可使用。下面的<bool\_obj>为一个由两个 Sphere 类源对象生成的 Boolean2 对象。

属性名称	数据类型	默认值
<bool_obj>.Sphere02	SubAnim	SubAnim:Sphere02
<bool_obj>.Operand_A_Transform	SubAnim	Operand_A_Transform
<bool_obj>.Sphere01	SubAnim	SubAnim:Sphere01
<bool_obj>.Operand_B_Transform	SubAnim	SubAnim:Operand_B_Transform

方法

1. boolObj.setOperandB <bool\_obj> <operand\_B> <add\_method> <mat\_method>

设置操作对象 B。参数<add\_method>和<mat\_method>的说明见上面。

2. boolObj.getOperandSel <bool\_obj> <Integer>

boolObj.setOperandSel <bool\_obj> <Integer> <boolean>

获取或设置选取哪个操作对象: <Integer> = 1: operand\_A; 2: operand\_B

3. boolObj.getBoolOp <bool\_obj>

boolObj.setBoolOp <bool\_obj> <Integer>

获取或设置 Boolean 操作类型, 有效值为:

- 1: Union (并集)
- 2: Intersection (交集)
- 3: Subtraction (A-B) (A 减 B)

4: Subtraction (B-A) (B 减 A)

5: Cut (切割)

4. boolObj.getBoolCutType <bool\_obj>

boolObj.setBoolCutType <bool\_obj> <Integer>

获取或设置当 Boolean 操作类型为 5 时的切割类型，有效值为：

1: Refine (优化)

2: Split (分割)

3: Remove Inside (移除内部)

4: Remove Outside (移除外部)

5. boolObj.getDisplayResult <bool\_obj>

boolObj.setDisplayResult <bool\_obj> <boolean>

获取或设置显示哪个对象：

◆ True 显示运算结果

◆ False 显示操作对象

6. boolObj.getShowHiddenOps <bool\_obj>

boolObj.setShowHiddenOps <bool\_obj> <boolean>

获取或设置是否显示运算结果和隐藏的操作对象：

◆ True 显示运算结果和隐藏的操作对象

◆ False 显示由 boolObj.SetDisplayResult()指定的对象

7. boolObj.getUpdateMode <bool\_obj>

boolObj.setUpdateMode <bool\_obj> <Integer>

获取或设置刷新模式，有效值为：

1: Always (自动刷新)

2: When Rendering (渲染时刷新)

3: Manually (手工刷新)

8. boolObj.getOptimize <bool\_obj>

boolObj.setOptimize <bool\_obj> <boolean>

获取或设置是否更新布尔对象。

#### 8.4.6.3 Conform: GeometryClass (一致复合对象)

Conform 类复合对象通过将某个对象（称为“包裹器”）的顶点投影至另一个对象（称为“包裹对象”）的表面而创建。Conform 类复合对象不能由 MAXScript 构造。

属性

属性名称	数据类型	默认值	说明
<Conform>.Projection_Distance	Float	1.0	如果包裹器对象与包裹目标对象之间不发生交错，包裹器上的顶点将依据该距离投影到包裹目标对象上。可动画
<Conform>.Standoff_Distance	Float	1.0	设定拟合投影后包裹对象与包裹目标对象顶点之间的距离。可动画
下面的属性仅在 Conform 对象被创建后才可使用。其中<Conform>为一个由两个 Sphere 类源对象生成的 Conform 对象			
<Conform>.S_Sphere02	SubAnim	默认值: SubAnim:S_Sphere02 说明: <Conform>的第二个源对象	
<Conform>.Operand_A_Transform	SubAnim	默认值: SubAnim:Operand_A_Transform 说明: <Conform>的第一个源对象的转换矩阵	
<Conform>.D_Sphere01	SubAnim	默认值: SubAnim:D_Sphere01 说明: <Conform>的第一个源对象	
<Conform>.Operand_B_Transform	SubAnim	默认值: SubAnim:Operand_B_Transform 说明: <Conform>的第二个源对象的转换矩阵	

#### 8.4.6.4 Connect: GeometryClass (连接复合对象)

Connect 类复合对象不能由 MAXScript 构造。

##### 属性

属性名称	数据类型	默认值	说明
<Connect>.segments	Integer	0	设置在连接桥部位的分段精度。可动画
<Connect>.tension	Float	0.5	控制连接桥表面的曲率。如果设置为 0，连接桥处没有曲率；增大该数值可以使连接桥处的曲线更为光滑，连接过渡更为柔和，但如果连接桥部位的分段精度设置为 0，张力的设置无效。可动画
下面的属性仅在 Connect 对象被创建后才可使用。其中<Connect>为一个由两个 Sphere 类源对象生成的 Conform 对象			
<Connect>.Op_0_Sphere02	SubAnim	SubAnim:Op_0_Sphere02	
<Connect>.Op_0_Transform	SubAnim	SubAnim:Op_0_Transform	
<Connect>.Op_1_Sphere01	SubAnim	SubAnim:Op_1_Sphere01	
<Connect>.Op_1_Transform	SubAnim	SubAnim:Op_1_Transform	

#### 8.4.6.5 Loft: GeometryClass (放样复合对象)

Loft 类复合对象不能由 MAXScript 构造。

##### 属性

属性名称	数据类型	默认值	备注
<Loft>.Def_Scale_X	SubAnim	SubAnim:Def_Scale_X	
<Loft>.Def_Scale_Y	SubAnim	SubAnim:Def_Scale_Y	
<Loft>.Def_Twist	SubAnim	SubAnim:Def_Twist	
<Loft>.Def_Teeter_X	SubAnim	SubAnim:Def_Teeter_X	可动画
<Loft>.Def_Teeter_Y	SubAnim	SubAnim:Def_Teeter_Y	可动画
<Loft>.Def_Bevel	SubAnim	SubAnim:Def_Bevel	
<Loft>.Def_Fit_X	SubAnim	SubAnim:Def_Fit_X	可动画
<Loft>.Def_Fit_Y	SubAnim	SubAnim:Def_Fit_Y	可动画
<Loft>.circle	SubAnim	SubAnim:Circle	
下面的属性仅在 Loft 对象被创建后才可使用。其中<Loft>为一个由 Ellipse (Path) 和 Circle (Shape) 对象生成的 Loft 对象			
<Loft>.ellipse	SubAnim	SubAnim:Ellipse	
<Loft>.circle	SubAnim	SubAnim:Circle	

#### 8.4.6.6 Morph: GeometryClass (变形复合对象)

构造函数

```
createMorphObject <source_Node>
```

属性

下面的属性仅在 Morph 对象被创建后才可使用。其中<Morph>为一个由三个 Sphere 对象生成的 Morph 对象。

属性名称	数据类型	默认值
<Morph>.M_Sphere01	SubAnim	SubAnim:M_Sphere01
<Morph>.M_Sphere02	SubAnim	SubAnim:M_Sphere02
<Morph>.M_Sphere03	SubAnim	SubAnim:M_Sphere03
<Morph>.Morph	Barycentric_Morph_Controller   Controller:Barycentric_Morph_Controller	

#### 8.4.6.7 Scatter: GeometryClass (离散复合对象)

Scatter 类复合对象不能由 MAXScript 构造。

属性

属性名称	数据类型	默认值	说明
<Scatter>.Duplicates	Integer	1	设置源对象在离散对象表面的副本数量。可动画
<Scatter>.Base_Scale	Float	100.0	指定源对象的放缩比率。可动画，百分数
<Scatter>.Vertex_Chaos	Float	0.0	指定源对象随机分布的混乱程度。可动画

(续表)

属性名称	数据类型	默认值	说明
<Scatter>.Animation_Offset	Time	0f	用于在进行源对象分布数量动画的时候,指定源对象副本出现的帧偏移数目。可动画
<Scatter>.x_rotation	Float	0.0	指定源对象副本在X轴向的随机旋转角度。可动画
<Scatter>.y_rotation	Float	0.0	指定源对象副本在Y轴向的随机旋转角度。可动画
<Scatter>.z_rotation	Float	0.0	指定源对象副本在Z轴向的随机旋转角度。可动画
<Scatter>.X_Translation	Float	0.0	指定源对象副本在X轴向的随机平移量。可动画
<Scatter>.Y_Translation	Float	0.0	指定源对象副本在Y轴向的随机平移量。可动画
<Scatter>.Z_Translation	Float	0.0	指定源对象副本在Z轴向的随机平移量。可动画
<Scatter>.A_Translation_on_Face	Float	0.0	指定源对象副本在A轴向的随机平移量。A、B是表面的局部坐标纵横方向。可动画
<Scatter>.B_Translation_on_Face	Float	0.0	指定源对象副本在B轴向的随机平移量。可动画
<Scatter>.N_Translation_on_Face	Float	0.0	指定源对象副本在N轴向的随机平移量。N是表面的法线方向。可动画
<Scatter>.Local_X_Scaling	Float	0.0	指定源对象副本在X轴向的随机放缩量。可动画
<Scatter>.Local_Y_Scaling	Float	0.0	指定源对象副本在Y轴向的随机放缩量。可动画
<Scatter>.Local_Z_Scaling	Float	0.0	指定源对象副本在Z轴向的随机放缩量。可动画
<Scatter>.S_Box04	SubAnim	SubAnim:S_Box04	
<Scatter>.Operand_A_Transform	SubAnim	SubAnim:Operand_A_Transform	
<Scatter>.D_Box05	SubAnim	SubAnim:D_Box05	
<Scatter>.Operand_B_Transform	SubAnim	SubAnim:Operand_B_Transform	

**注意** 许多与 Scatter 对象有关的属性都不能在 MAXScript 里进行存取。

#### 8.4.6.8 ShapeMerge: GeometryClass (图形合并复合对象)

ShapeMerge 类复合对象不能由 MAXScript 构造。

##### 属性

属性名称	数据类型	默认值	说明
<ShapeMerge>.Operation_Type	Integer	1	设置图形合并的操作方式: 0: Cookie Cutter(饼切, 切去网格对象曲面外部的图形) 1: Merge(合并, 将图形与网格对象曲面合并)
<ShapeMerge>.Remove_Interior_Exterior	Integer	0	设置反转饼切或合并效果: 0: Invert off(切除曲线内部的网格对象表面) 1: Invert on(切除曲线外部的网格对象表面)

(续表)

属性名称	数据类型	默认值	说明
<ShapeMerge>.Output_Sub_Mesh_Selection	Integer	0	用于指定哪一级的次级结构层级被传递到堆栈中 0: None (输出整个对象) 1: Edge (输出合并图形的边) 2: Face (输出合并图形内的面) 3: Vertex (输出由图形样条曲线定义的顶点)
下面的属性仅在 ShapeMerge 对象被创建后才可使用。其中< ShapeMerge >为一个由一个 Sphere 和一个 Circle 对象生成的 ShapeMerge 对象			
<ShapeMerge>.mesh_sphere01	SubAnim	SubAnim:Mesh_Sphere01	
<ShapeMerge>.mesh_transform	SubAnim	SubAnim:Mesh_Transform	
<ShapeMerge>.shape_1_circle01	SubAnim	SubAnim:Shape_1_Circle01	
<ShapeMerge>.shape_1_transform	SubAnim	SubAnim:Shape_1_Transform	

#### 8.4.6.9 Terrain: GeometryClass (地形复合对象)

##### 构造函数

terrain()

##### 属性

属性名称	数据类型	默认值	说明
<Terrain>.Form	Integer	1	设置地形复合对象的形态项目： 0: Graded Surface (分级曲面，在轮廓上创建网格的分级曲面) 1: Graded Solid (分级实体，创建侧面带有裙子的分级曲面和底面) 2: Layered Solid (分层实体，依据等高线创建包含底面的阶梯状分层实体)
<Terrain>.stitchBorder	Boolean	False	开/关缝合边界
<Terrain>.retriangulate	Boolean	False	开/关重新三角化
<Terrain>.display	Integer	0	设置地形复合对象的显示项目： 0: Terrain (只显示地形合成对象基于等高线框架的三角网格表面) 1: Contours (只显示地形合成对象的等高线框架) 2: Both (显示地形合成对象的等高线框架和其三角网格表面)
<Terrain>.update	Integer	0	设置地形复合对象的更新方式： 0: Always (在地形合成操作调整过程中总是自动更新) 1: When Rendering (在场景最终渲染输出时才进行场景更新) 2: Manually (手动更新)

(续表)

属性名称	数据类型	默认值	说明
<Terrain>.numOps	Integer	0	合成对象的数量, 只读
<Terrain>.horizSimplification	Integer	0	0: No Simplification (不简化); 1: Use 1/2 of Points (使用 1/2 的点) 2: Use 1/4 of Points (使用 1/4 的点) 3: Interpolate Points * 2 (使用 2 倍顶点) 4: Interpolate Points * 4 (使用 4 倍顶点)
<Terrain>.vertSimplification	Integer	0	0: No Simplification (不简化) 1: Use 1/2 of Lines (使用 1/2 的点) 2: Use 1/4 of Lines (使用 1/4 的点)
下面的属性仅在 Terrain 对象被创建后才可使用。其中<Terrain>为一个由三个 Circle 对象生成的 Terrain 对象			
属性名称	数据类型	默认值	
<Terrain>.Op_0_Circle01	SubAnim	SubAnim:Op_0_Circle01	
<Terrain>.Op_0_Transform	SubAnim	SubAnim:Op_0_Transform	
<Terrain>.Op_1_Circle02	SubAnim	SubAnim:Op_1_Circle02	
<Terrain>.Op_1_Transform	SubAnim	SubAnim:Op_1_Transform	
<Terrain>.Op_2_Circle03	SubAnim	SubAnim:Op_2_Circle03	
<Terrain>.Op_2_Transform	SubAnim	SubAnim:Op_2_Transform	

## 方法

### 1. TerrainOps.addOperand <Terrain> <node>

将指定<node>作为等高线操作对象添加到指定<Terrain>对象。注意该操作是在 Move 模式下进行, 所以指定<node>在操作完成后会被删除。如果暂时还不想删除该对象, 必须使用其他合适的 MAXScript 函数, 例如:

```
TerrainOps.addOperand $Terrain01 (instance $line03)
```

### 2. TerrainOps.deleteOperand <Terrain> <index\_Integer>

删除指定序号的操作对象, 序号从 1 开始。

### 3. TerrainOps.getOperand <Terrain> <index\_Integer>

返回指定序号的等高线操作对象 (作为一个 3ds max 基本对象而不是 Node 对象)。序号从 1 开始。

### 4. TerrainOps.getOperandTM <Terrain> <index\_Integer>

获取指定序号的操作对象的 Local 坐标转换矩阵, 数据类型为 Matrix3, 这个转换矩阵为相对于<Terrain>对象的 Node 转换矩阵。

### 5. TerrainOps.setOperandTM <Terrain> <index\_Integer> <matrix3>

将指定序号的操作对象的 Local 坐标转换矩阵赋给指定值<Matrix3>, 这个转换矩阵为相对于<Terrain>对象的 Node 转换矩阵。

### 6. TerrainOps.update <Terrain>

以手工方式强制刷新指定<Terrain>对象。

**注意**

上面这些方法由动态链接库文件 Landform.DLL 定义，所以只有在 Landform.DLL 文件被装载后才可以使用。在安装 3ds max 时，如果选择了 delayed plugin loading 项，当装载一个包含 Terrain 类对象的场景或通过 MAXScript 在界面里创建 Terrain 类对象时，系统会自动装载 Landform.DLL 文件。

### 8.4.7 Geometry-Door 和 Window（门窗建筑对象）

下面列出了所有的 Door 和 Window 类对象：

Awning（遮蓬式窗口）

BiFold（BiFold 门）

Casement（窗扉窗口）

Fixed（固定窗口）

Pivot（轴窗口）

Projected（投射窗口）

SlidingDoor（滑动门）

SlidingWindow（滑动窗口）

#### 8.4.7.1 Awning: GeometryClass（遮蓬式窗口）

构造函数

Awning ...

属性

属性名称	数据类型	默认值	备注
<Awning>.height	Float	0.0	可动画
<Awning>.width	Float	0.0	可动画
<Awning>.depth	Float	0.0	可动画
<Awning>.Horizontal_Frame_Width	Float	2.0	可动画
<Awning>.Vertical_Frame_Width	Float	2.0	可动画
<Awning>.Frame_Thickness	Float	0.5	可动画
<Awning>.Glazing_Thickness	Float	0.25	可动画
<Awning>.Rail_Width	Float	1.0	可动画
<Awning>.Number_of_Panels	Integer	1	
<Awning>.Percent_Open	Integer	0	可动画
<Awning>.Generate_Mapping_Coords	Boolean	False	

#### 8.4.7.2 BiFold: GeometryClass（BiFold 门）

构造函数

BiFold ...

## 属性

属性名称	数据类型	默认值	备注
<BiFold>.height	Float	0.0	可动画
<BiFold>.width	Float	0.0	可动画
<BiFold>.depth	Float	0.0	可动画
<BiFold>.Double_Doors	Integer	0	0: False; 1: True
<BiFold>.Flip_Swing	Boolean	False	
<BiFold>.Flip_Hinge	Boolean	False	
<BiFold>.open	Float	0.0	可动画
<BiFold>.Create_Frame	Boolean	True	
<BiFold>.Frame_Width	Float	2.0	可动画
<BiFold>.Frame_Depth	Float	1.0	可动画
<BiFold>.Door_Offset	Float	0.0	可动画
<BiFold>.Generate_Mapping_Coords	Boolean	False	
<BiFold>.Leaf_Thickness	Float	2.0	可动画
<BiFold>.Stiles_Top_Rail	Float	4.0	可动画
<BiFold>.Bottom_Rail	Float	12.0	可动画
<BiFold>.Number_of_Panels_Horizontally	Integer	1	
<BiFold>.Number_of_Panels_Vertically	Integer	1	
<BiFold>.Muntin	Float	2.0	可动画
<BiFold>.Panel_Type	Integer	1	0: None; 1: Glass; 2: Beveled
<BiFold>.Glass_Thickness	Float	0.25	可动画
<BiFold>.Bevel_Angle	Float	45.0	可动画
<BiFold>.Panel_Thickness_1	Float	0.25	可动画
<BiFold>.Panel_Thickness_2	Float	0.5	可动画
<BiFold>.Panel_Middle_Thickness	Float	0.25	可动画
<BiFold>.Panel_Width_1	Float	1.0	可动画
<BiFold>.Panel_Width_2	Float	0.5	可动画

## 8.4.7.3 Casement: GeometryClass(窗扉窗口)

## 构造函数

Casement ...

## 属性

属性名称	数据类型	默认值	备注
<Casement>.height	Float	0.0	可动画
<Casement>.width	Float	0.0	可动画
<Casement>.depth	Float	0.0	可动画
<Casement>.Horizontal_Frame_Width	Float	2.0	可动画
<Casement>.Vertical_Frame_Width	Float	2.0	可动画
<Casement>.Frame_Thickness	Float	0.5	可动画
<Casement>.Glazing_Thickness	Float	0.25	可动画
<Casement>.Panel_Width	Float	1.0	可动画
<Casement>.Number_of_Casements	Integer	1	1: One; 2: Two
<Casement>.Percent_Open	Integer	0	可动画
<Casement>.Opens_to_Inside	Boolean	False	
<Casement>.Generate_Mapping_Coords	Boolean	False	

#### 8.4.7.4 Fixed: GeometryClass (固定窗口)

构造函数

Fixed ...

属性

属性名称	数据类型	默认值	备注
<Fixed>.height	Float	0.0	可动画
<Fixed>.width	Float	0.0	可动画
<Fixed>.depth	Float	0.0	可动画
<Fixed>.Horizontal_Frame_Width	Float	2.0	可动画
<Fixed>.Vertical_Frame_Width	Float	2.0	可动画
<Fixed>.Frame_Thickness	Float	0.5	可动画
<Fixed>.Glazing_Thickness	Float	0.25	可动画
<Fixed>.Rail_Width	Float	1.0	可动画
<Fixed>.Number_of_Panels_Horizontally	Integer	1	
<Fixed>.Number_of_Panels_Vertically	Integer	1	
<Fixed>.Chamfered_Profile	Boolean	False	
<Fixed>.Generate_Mapping_Coords	Boolean	False	

#### 8.4.7.5 Pivot: GeometryClass (轴窗口)

构造函数

Pivot ...

属性

属性名称	数据类型	默认值	备注
<pivot>.height	Float	0.0	可动画
<pivot>.width	Float	0.0	可动画
<pivot>.depth	Float	0.0	可动画
<pivot>.Double_Doors	Integer	0	0: False; 1: True
<pivot>.Flip_Swing	Boolean	False	
<pivot>.Flip_Hinge	Boolean	False	
<pivot>.Open_degrees	Float	0.0	可动画
<pivot>.Create_Frame	Boolean	True	
<pivot>.Frame_Width	Float	2.0	可动画
<pivot>.Frame_Depth	Float	1.0	可动画
<pivot>.Door_Offset	Float	0.0	可动画
<pivot>.Generate_Mapping_Coords	Boolean	False	
<pivot>.Leaf_Thickness	Float	2.0	可动画
<pivot>.Stiles_Top_Rail	Float	4.0	可动画
<pivot>.Bottom_Rail	Float	12.0	可动画
<pivot>.Number_of_Panels_Horizontally	Integer	1	
<pivot>.Number_of_Panels_Vertically	Integer	1	
<pivot>.Muntin	Float	2.0	可动画
<pivot>.Panel_Type	Integer	1	0: None; 1: Glass; 2: Beveled
<pivot>.Glass_Thickness	Float	0.25	可动画
<pivot>.Bevel_Angle	Float	45.0	可动画
<pivot>.Panel_Thickness_1	Float	0.25	可动画
<pivot>.Panel_Thickness_2	Float	0.5	可动画
<pivot>.Panel_Middle_Thickness	Float	0.25	可动画
<pivot>.Panel_Width_1	Float	1.0	可动画
<pivot>.Panel_Width_2	Float	0.5	可动画
<Pivoted>.height	Float	0.0	可动画
<Pivoted>.width	Float	0.0	可动画
<Pivoted>.depth	Float	0.0	可动画
<Pivoted>.Horizontal_Frame_Width	Float	2.0	可动画
<Pivoted>.Vertical_Frame_Width	Float	2.0	可动画
<Pivoted>.Frame_Thickness	Float	0.5	可动画
<Pivoted>.Glazing_Thickness	Float	0.25	可动画
<Pivoted>.Rail_Width	Float	1.0	可动画
<Pivoted>.Vertical_Rotation	Boolean	False	
<Pivoted>.Percent_Open	Integer	0	可动画
<Pivoted>.Generate_Mapping_Coords	Boolean	False	

#### 8.4.7.6 Projected: GeometryClass (投射窗口)

构造函数

projected ...

#### 属性

属性名称	数据类型	默认值	备注
<projected>.height	Float	0.0	可动画
<projected>.width	Float	0.0	可动画
<projected>.depth	Float	0.0	可动画
<projected>.Horizontal_Frame_Width	Float	2.0	可动画
<projected>.Vertical_Frame_Width	Float	2.0	可动画
<projected>.Frame_Thickness	Float	0.5	可动画
<projected>.Glazing_Thickness	Float	0.25	可动画
<projected>.Rail_Width	Float	1.0	可动画
<projected>.Middle_Height	Float	0.0	可动画
<projected>.Bottom_Height	Float	0.0	可动画
<projected>.Percent_Open	Integer	0	可动画
<projected>.Generate_Mapping_Coords	Boolean	False	

#### 8.4.7.7 SlidingDoor: GeometryClass (滑动门)

##### 构造函数

SlidingDoor...

#### 属性

属性名称	数据类型	默认值	备注
<SlidingDoor>.height	Float	0.0	可动画
<SlidingDoor>.width	Float	0.0	可动画
<SlidingDoor>.depth	Float	0.0	可动画
<SlidingDoor>.Flip_Swing	Boolean	False	
<SlidingDoor>.Flip_Hinge	Boolean	False	
<SlidingDoor>.open	Float	0.0	可动画
<SlidingDoor>.Create_Frame	Boolean	True	
<SlidingDoor>.Frame_Width	Float	2.0	可动画
<SlidingDoor>.Frame_Depth	Float	1.0	可动画
<SlidingDoor>.Door_Offset	Float	0.0	可动画
<SlidingDoor>.Generate_Mapping_Coords	Boolean	False	
<SlidingDoor>.Leaf_Thickness	Float	2.0	可动画
<SlidingDoor>.Stiles_Top_Rail	Float	4.0	可动画
<SlidingDoor>.Bottom_Rail	Float	12.0	可动画
<SlidingDoor>.Number_of_Panels_Horizontally	Integer	1	
<SlidingDoor>.Number_of_Panels_Vertically	Integer	1	
<SlidingDoor>.Muntin	Float	4.0	可动画

(续表)

属性名称	数据类型	默认值	备注
<SlidingDoor>.Panel_Type	Integer	1	0: None; 1: Glass; 2: Beveled
<SlidingDoor>.Glass_Thickness	Float	4.0	可动画
<SlidingDoor>.Bevel_Angle	Float	0.45	可动画
<SlidingDoor>.Panel_Thickness_1	Float	0.25	可动画
<SlidingDoor>.Panel_Thickness_2	Float	0.5	可动画
<SlidingDoor>.Panel_Middle_Thickness	Float	0.25	可动画
<SlidingDoor>.Panel_Width_1	Float	1.0	可动画
<SlidingDoor>.Panel_Width_2	Float	0.5	可动画
<SlidingDoor>.Double_Doors	Integer	0	

**注意** SlidingDoor类对象不使用属性.Double\_Doors。

#### 8.4.7.8 SlidingWindow: GeometryClass (滑动窗口)

构造函数

slidingWindow ...

属性

属性名称	数据类型	默认值	备注
<SlidingWindow>.height	Float	0.0	可动画
<SlidingWindow>.width	Float	0.0	可动画
<SlidingWindow>.depth	Float	0.0	可动画
<SlidingWindow>.Horizontal_Frame_Width	Float	2.0	可动画
<SlidingWindow>.Vertical_Frame_Width	Float	2.0	可动画
<SlidingWindow>.Frame_Thickness	Float	0.5	可动画
<SlidingWindow>.Glazing_Thickness	Float	0.25	可动画
<SlidingWindow>.Rail_Width	Float	1.0	可动画
<SlidingWindow>.Number_of_Panels_Vertically	Integer	1	
<SlidingWindow>.Chamfered_Profile	Boolean	False	
<SlidingWindow>.Hung	Boolean	True	
<SlidingWindow>.Percent_Open	Integer	0	可动画
<SlidingWindow>.Generate_Mapping_Coords	Boolean	False	

#### 8.4.8 Stair: GeometryClass (楼梯建筑对象)

##### 8.4.8.1 L\_Type\_Stair: GeometryClass (L形楼梯)

构造函数

L\_Type\_Stair ...

LTypeStair ...

属性

属性名称	数据类型	默认值	说明
<L_Type_Stair>.angle	Float	90.0	控制平台与第二段楼梯的角度。范围为-90° 至 90° 。 可动画
<L_Type_Stair>.CarriageContext	数据类型: Integer 默认值: 18		<p>包含沿路径分布对象方式的下拉列表选项:</p> <p>0: 自由中心 (从路径始端开始, 沿直线朝路径末端等距分布对象。样条线或一对点可定义路径。可以指定对象数量和间隔)</p> <p>1: 均匀分隔, 对象位于端点 (沿样条线分布对象。对象组以样条线的中间为中心)</p> <p>2: 居中, 指定间距 (沿路径分布对象。对象组以路径的中间为中心)</p> <p>3: 末端偏移, 指定间距 (沿直线分布指定数量的对象。对象从指定的偏移距离处开始分布。此距离是从样条曲线的端点到其起点, 或者从一对点的第二个点到第一个点。也可以指定对象之间的间隔)</p> <p>4: 末端偏移, 均匀分隔 (在样条曲线或一对点的始端与指定的末端偏移之间, 分布指定数量的对象。始终将对象放置在末端或其偏移处。如果指定了多个对象, 则在始端始终放有对象)</p> <p>5: 末端偏移, 指定间距 (从末端或其偏移处开始, 朝样条曲线或一对点的始端分布对象。始终将对象放置在末端或其偏移处。可以指定对象之间的间隔以及距末端的偏移)</p> <p>6: 始端偏移, 指定间距 (沿直线分布指定数量的对象。对象从指定的偏移距离处开始分布。此距离是从样条曲线的始端到其端点, 或者从一对点的第一个点到第二个点。也可以指定对象之间的间隔)</p> <p>7: 始端偏移, 均匀分隔 (从指定的距始端的偏移处开始, 在样条曲线或一对点的末端之间分布指定数量的对象。始终将对象放置在始端或其偏移处。如果指定了多个对象, 则在末端始终放有对象)</p> <p>8: 始端偏移, 指定间距 (从始端开始, 朝样条曲线或一对点的末端分布对象。始终将对象放置在始端或其偏移处。可以指定对象之间的间隔以及距始端的偏移)</p> <p>9: 指定偏移和间距 (沿样条曲线或在一对点之间分布尽可能多的等距对象。可以指定对象之间的间隔。如果指定距始端和末端的偏移, 则会在这两个偏移之间分布等距对象。在始端和末端并非始终放有对象)</p> <p>10: 指定偏移, 均匀分隔: (沿样条曲线或在一对点之间分布指定数量的对象。如果指定一个对象, 会将其置于路径的中央。如果指定一个以上的对象, 始终会在始端偏移和末端偏移上放置对象。如果指定两个以上的对象, 会在上述偏移之间均匀地分布对象)</p> <p>11: 从末端间隔, 无限的 (从样条曲线或一对点的末端朝始端沿直线分布指定数量的对象。可以指定对象之间的间隔。会锁定末端偏移, 以便末端偏移与间隔相同)</p> <p>12: 从末端间隔, 指定数量 (从末端开始, 朝样条曲线或一对点的始端分布指定数量的对象。根据对象数量以及样条曲线长度或一对点之间的距离来确定对象之间的间距。会锁定末端偏移, 以便末端偏移与间隔相同)</p> <p>13: 从末端间隔, 指定间距 (从末端开始, 朝样条曲线或一对点的始端分布尽可能多的等距对象。可以指定对象之间的间隔。会锁定末端偏移, 以便末端偏移与间隔相同)</p> <p>14: 从始端间隔, 无限的 (从始端开始, 朝样条曲线或一对点的末端沿直线分布指定数量的对象。可以指定对象之间的间隔。会锁定始端偏移, 以便始端偏移与间隔相同)</p> <p>15: 从始端间隔, 指定数量 (从始端开始, 朝样条曲线或一对点的末端分布指定数量的对象。根据对象数量以及样条曲线长度或一对点之间的距离来确定对象之间的间距。会锁定始端偏移, 以便始端偏移与间隔相同)</p> <p>16: 从始端间隔, 指定间距 (从始端开始, 朝样条曲线或一对点的末端分布尽可能多的等距对象。可以指定对象之间的间隔。会锁定始端偏移, 以便始端偏移与间隔相同)</p> <p>17: 指定间距, 匹配偏移 (沿样条曲线或在一对点 (及其偏移) 之间分布尽可能多的等距对象。可以指定间隔。会锁定始端和末端偏移, 以便这些偏移与间隔相同)</p> <p>18: 均匀分隔, 没有对象位于端点 (沿样条曲线或在一对点 (及其偏移) 之间分布指定数量的对象。会锁定始端和末端偏移, 以便这些偏移与间隔相同)</p>

(续表)

属性名称	数据类型	默认值	说明
<L_Type_Stair>.CarriageCount	Integer	1	设置要分布对象的数量。可动画
<L_Type_Stair>.CarriageExtOffs	Float	3.0	指定距路径始端偏移的单位数量。可动画
<L_Type_Stair>.CarriageHeight	Float	8.0	控制支撑梁离地面的深度。可动画
<L_Type_Stair>.CarriageIntOffs	Float	3.0	指定距路径末端偏移的单位数量。可动画
<L_Type_Stair>.CarriageSpace	Float	10.0	指定对象之间的间距。可动画
<L_Type_Stair>.CarriageSpacingType	Integer	0	设置间距计算类型: 0: 边至边的间距。通过各对象边界框的相对边确定间隔 1: 中心至中心的间距, 将分布对象的轴点与样条曲线的切线对齐
<L_Type_Stair>.CarriageSpringFloor	Integer	1	控制支撑梁是从地面开始, 还是与第一个梯级竖板的开始平齐, 或是否支撑梁延伸到地面以下。可动画 0: Off; 1: On
<L_Type_Stair>.CarriageWidth	Float	3.0	控制支撑梁的宽度。可动画
<L_Type_Stair>.GenerateCarriage	Integer	1	是否设置支撑梁。可动画
<L_Type_Stair>.GenerateInsideRailing	Integer	0	是否创建内扶手
<L_Type_Stair>.GenerateMapping	Integer	0	设置是否对楼梯应用默认的贴图坐标: 0: Off; 1: On
<L_Type_Stair>.GenerateOutsideRailing	Integer	0	是否创建外扶手
<L_Type_Stair>.GenerateStringers	Integer	0	设置是否创建侧弦。可动画
<L_Type_Stair>.length	Float	0.0	控制第一段楼梯的长度。可动画
<L_Type_Stair>.length2	Float	0.0	控制第二段楼梯的长度。可动画
<L_Type_Stair>.RailingHeight	Float	42.0	控制栏杆离台阶的高度。可动画
<L_Type_Stair>.RailingOffs	Float	2.0	控制栏杆离台阶端点的偏移。可动画
<L_Type_Stair>.RailingRadius	Float	1.0	控制栏杆的厚度。可动画
<L_Type_Stair>.RailingSegments	Integer	3	指定栏杆的分段数目。值越高, 栏杆显得越光滑。可动画
<L_Type_Stair>.StepCount	Integer	2	控制梯级竖板数。梯级竖板总是比台阶多一个。隐式梯级竖板位于上板和楼梯顶部台阶之间。可动画
<L_Type_Stair>.StepDepth	Float	0.0	当 StepDepth_X 设置为 On 时, 控制台阶的深度。可动画
<L_Type_Stair>.StepDepth_X	Integer	0	是否需要设置台阶的深度: 0: Off; 1: On
<L_Type_Stair>.StepHeight	Float	0.0	控制梯级竖板的高度。可动画
<L_Type_Stair>.StepThickness	Float	2.0	控制台阶的厚度。可动画
<L_Type_Stair>.StepType	Integer	0	设置楼梯的类型: 0: Open (创建开放式的梯级竖板楼梯) 1: Closed (创建封闭式的梯级竖板楼梯) 2: Box (创建带有封闭式梯级竖板和两侧有封闭式侧弦的楼梯)

(续表)

属性名称	数据类型	默认值	说明
<L_Type_Stair>.Stepwidth	Float	39.0	控制楼梯的宽度，包括台阶和平台。可动画
<L_Type_Stair>.StringerDepth	Float	6.0	控制侧弦离地板的深度。可动画
<L_Type_Stair>.StringerOffset	Float	1.0	控制地板与侧弦的垂直距离。可动画
<L_Type_Stair>.StringerSpringFloor	Integer		控制侧弦是从地面开始，还是与第一个梯级竖板的开始平齐，或是否侧弦延伸到地面以下。使用偏移选项可以控制侧弦延伸到地面以下的量。可动画 0: Off; 1: On
<L_Type_Stair>.StringerWidth	Float	1.0	控制侧弦的宽度。可动画
<L_Type_Stair>.UpperOffset	Float	0.0	控制平台与第二段楼梯的距离。相应调整平台的长度。可动画

#### 8.4.8.2 Spiral\_Stair: GeometryClass (螺旋楼梯)

构造函数

Spiral\_Stair ...

属性

属性名称	数据类型	默认值	说明
<Spiral_Stair>.CarriageContext	Integer	18	包含沿路径分布对象方式的下拉列表选项（参见<L_Type_Stair>.Carriage Context）
<Spiral_Stair>.CarriageCount	Integer	1	设置要分布的对象的数量。可动画
<Spiral_Stair>.CarriageExtOffs	Float	3.0	指定距路径始端偏移的单位数量。可动画
<Spiral_Stair>.CarriageHeight	Float	8.0	控制支撑梁离地面的深度。可动画
<Spiral_Stair>.CarriageIntOffs	Float	3.0	指定距路径末端偏移的单位数量。可动画
<Spiral_Stair>.CarriageSpace	Float	10.0	指定对象之间的间距（以单位计）。可动画
<Spiral_Stair>.CarriageSpacingType	Integer	0	设置间距计算类型： 0: 边至边的间距。通过各对象边界框的相对边确定间隔 1: 中心至中心的间距，将分布对象的轴点与样条曲线的切线对齐
<Spiral_Stair>.CarriageSpringFloor	Integer	1	控制支撑梁是从地面开始，还是与第一个梯级竖板的开始平齐，或是否支撑梁延伸到地面以下。可动画 0: Off; 1: On
<Spiral_Stair>.CarriageWidth	Float	3.0	控制支撑梁的宽度。可动画
<Spiral_Stair>.CenterPoleHeight	Float	0.0	控制中柱的高度。可动画
<Spiral_Stair>.CenterPoleHeight_X	Integer	0	设置是否使用中柱
<Spiral_Stair>.CenterPoleRadius	Float	10.0	控制中柱的半径大小。可动画
<Spiral_Stair>.CenterPoleSegments	Integer	16	指定中柱的分段数。值越高，中柱显示得越光滑。可动画
<Spiral_Stair>.direction	Integer	0	设置楼梯的旋转方向。可动画 0: CCW; 1: CW

(续表)

属性名称	数据类型	默认值	说明
<Spiral_Stair>.GenerateCarriage	Integer	1	是否设置支撑梁。可动画
<Spiral_Stair>.GenerateCenterPole	Integer	0	是否设置中柱。可动画
<Spiral_Stair>.GenerateInsideRailing	Integer	0	是否创建内扶手
<Spiral_Stair>.GenerateMapping	Integer	0	设置是否对楼梯应用默认的贴图坐标: 0: Off; 1: On
<Spiral_Stair>.GenerateOutsideRailing	Integer	0	是否创建外扶手
<Spiral_Stair>.GenerateStringers	Integer	0	设置是否创建侧弦。可动画
<Spiral_Stair>.radius	Float	0.0	控制螺旋的半径大小。可动画
<Spiral_Stair>.RailingHeight	Float	42.0	控制栏杆离台阶的高度。可动画
<Spiral_Stair>.RailingOffs	Float	2.0	控制栏杆离台阶端点的偏移。可动画
<Spiral_Stair>.RailingRadius	Float	1.0	控制扶手的厚度。可动画
<Spiral_Stair>.RailingSegments	Integer	3	指定扶手的分段数目。值越高，扶手越光滑。可动画
<Spiral_Stair>.Revolutions	Float	0.75	指定螺旋的转数。可动画
<Spiral_Stair>.StepCount	Integer	12	控制梯级竖板数。梯级竖板总是比台阶多一个。隐式梯级竖板位于上板和楼梯顶部台阶之间。可动画
<Spiral_Stair>.StepDepth	Float	0.0	当 StepDepth_X 设置为 On 时，控制台阶的深度。可动画
<Spiral_Stair>.StepDepth_X	Integer	0	是否需要设置台阶的深度： 0: Off; 1: On
<Spiral_Stair>.StepHeight	Float	0.0	控制梯级竖板的高度。可动画
<Spiral_Stair>.StepSegments	Integer	0	控制 3ds max 用于构建台阶的分段数。可动画
<Spiral_Stair>.StepSegments_X	Integer	0	是否设置台阶的分段数：0: Off; 1: On
<Spiral_Stair>.StepThickness	Float	2.0	控制台阶的厚度。可动画
<Spiral_Stair>.StepType	Integer	0	设置楼梯的类型： 0: Open (创建开放式的梯级竖板楼梯) 1: Closed (创建封闭式的梯级竖板楼梯) 2: Box (创建带有封闭式梯级竖板和两侧有封闭式侧弦的楼梯)
<Spiral_Stair>.StepWidth	Float	39.0	控制螺旋楼梯的宽度。可动画
<Spiral_Stair>.StringerDepth	Float	6.0	控制侧弦离地板的深度。可动画
<Spiral_Stair>.StringerOffset	Float	1.0	控制地板与侧弦的垂直距离。可动画
<Spiral_Stair>.StringerSpringFloor	Integer	1	控制侧弦是从地面开始，还是与第一个梯级竖板的开始平齐，或是否侧弦延伸到地面以下。使用偏移选项可以控制侧弦延伸到地面以下的量。可动画 0: Off; 1: On
<Spiral_Stair>.StringerWidth	Float	1.0	控制侧弦的宽度。可动画

### 8.4.8.3 Straight\_Stair: GeometryClass (直线楼梯)

构造函数

`Straight_Stair ...`

属性

属性名称	数据类型	默认值	说明
<code>&lt;Straight_Stair&gt;.CarriageContext</code>	Integer	18	包含沿路径分布对象方式的下拉列表选项 (参见 <code>&lt;L_Type_Stair&gt;.CarriageContext</code> )
<code>&lt;Straight_Stair&gt;.CarriageCount</code>	Integer	1	设置要分布的对象的数量。可动画
<code>&lt;Straight_Stair&gt;.CarriageExtOffs</code>	Float	3.0	指定距路径始端偏移的单位数量。可动画
<code>&lt;Straight_Stair&gt;.CarriageHeight</code>	Float	8.0	控制支撑梁离地面的深度。可动画
<code>&lt;Straight_Stair&gt;.CarriageIntOffs</code>	Float	3.0	指定距路径末端偏移的单位数量。可动画
<code>&lt;Straight_Stair&gt;.CarriageSpace</code>	Float	10.0	指定对象之间的间距(以单位计)。可动画
<code>&lt;Straight_Stair&gt;.CarriageSpacingType</code>	Integer	0	设置间距计算类型: 0: 边至边的间距, 通过各对象边界框的相对边确定间隔 1: 中心至中心的间距, 将分布对象的轴点与样条曲线的切线对齐
<code>&lt;Straight_Stair&gt;.CarriageSpringFloor</code>	Integer	1	控制支撑梁是从地面开始, 还是与第一个梯级竖板的开始平齐, 或是否支撑梁延伸到地面以下。可动画 0: Off; 1: On
<code>&lt;Straight_Stair&gt;.CarriageWidth</code>	Float	3.0	控制支撑梁的宽度。可动画
<code>&lt;Straight_Stair&gt;.GenerateCarriage</code>	Integer	1	是否设置支撑梁。可动画
<code>&lt;Straight_Stair&gt;.GenerateInsideRailing</code>	Integer	0	是否创建内扶手
<code>&lt;Straight_Stair&gt;.GenerateMapping</code>	Integer	0	设置是否对楼梯应用默认的贴图坐标: 0: Off; 1: On
<code>&lt;Straight_Stair&gt;.GenerateOutsideRailing</code>	Integer	0	是否创建外扶手
<code>&lt;Straight_Stair&gt;.GenerateStringers</code>	Integer	0	设置是否创建侧弦。可动画
<code>&lt;Straight_Stair&gt;.length</code>	Float	0.0	控制楼梯的长度。可动画
<code>&lt;Straight_Stair&gt;.RailingHeight</code>	Float	42.0	控制栏杆离台阶的高度。可动画
<code>&lt;Straight_Stair&gt;.RailingOffs</code>	Float	2.0	控制栏杆离台阶端点的偏移。可动画
<code>&lt;Straight_Stair&gt;.RailingRadius</code>	Float	1.0	控制扶手的厚度。可动画
<code>&lt;Straight_Stair&gt;.RailingSegments</code>	Integer	3	指定扶手的分段数目。值越高, 扶手显得越光滑。可动画
<code>&lt;Straight_Stair&gt;.StepCount</code>	Integer	12	控制梯级竖板数。梯级竖板总是比台阶多一个。隐式梯级竖板位于上板和楼梯顶部台阶之间。可动画

(续表)

属性名称	数据类型	默认值	说明
<Straight_Stair>.StepDepth	Float	0.0	当StepDepth_X设置为On时，控制台阶的深度。可动画
<Straight_Stair>.StepDepth_X	Integer	0	是否需要设置台阶的深度： 0: Off; 1: On
<Straight_Stair>.StepHeight	Float	0.0	控制梯级竖板的高度。可动画
<Straight_Stair>.StepThickness	Float	2.0	控制台阶的厚度。可动画
<Straight_Stair>.StepType	Integer	0	设置楼梯的类型： 0: Open (创建开放式的梯级竖板楼梯) 1: Closed (创建封闭式的梯级竖板楼梯) 2: Box (创建带有封闭式梯级竖板和两侧有封闭式侧弦的楼梯)
<Straight_Stair>.Stepwidth	Float	1.0	控制台阶的厚度。可动画
<Straight_Stair>.StringerDepth	Float	6.0	控制侧弦离地板的深度。可动画
<Straight_Stair>.StringerOffset	Float	1.0	控制地板与侧弦的垂直距离。可动画
<Straight_Stair>.StringerSpringFloor	Integer	1	控制侧弦是从地面开始，还是与第一个梯级竖板的开始平齐，或是否侧弦延伸到地面以下。使用偏移选项可以控制侧弦延伸到地面以下的量。可动画 0: Off; 1: On
<Straight_Stair>.StringerWidth	Float	1.0	控制侧弦的宽度。可动画

#### 8.4.8.4 U\_Type\_Stair: GeometryClass (U形楼梯)

构造函数

U\_Type\_Stair...

UTypeStair...

属性

属性名称	数据类型	默认值	说明
<U_Type_Stair>.CarriageContext	Integer	18	包含沿路径分布对象方式的下拉列表选项 (参见<L_Type_Stair>.CarriageContext)
<U_Type_Stair>.CarriageCount	Integer	1	设置要分布的对象的数量。可动画
<U_Type_Stair>.CarriageExtOffs	Float	3.0	指定距路径始端偏移的单位数量。可动画
<U_Type_Stair>.CarriageHeight	Float	8.0	控制支撑梁离地面的深度。可动画
<U_Type_Stair>.CarriageIntOffs	Float	3.0	指定距路径末端偏移的单位数量。可动画
<U_Type_Stair>.CarriageSpace	Float	10.0	指定对象之间的间距(以单位计)。可动画
<U_Type_Stair>.CarriageSpacingType	Integer	0	设置间距计算类型： 0: 边至边的间距，各对象边界框的相对边确定间隔 1: 中心至中心的间距，将分布对象的轴点与样条曲线的切线对齐

(续表)

属性名称	数据类型	默认值	说明
<U_Type_Stair>.CarriageSpringFloor	Integer	1	控制支撑梁是从地面开始,还是与第一个梯级竖板的开始平齐,或是否支撑梁延伸到地面以下。可动画 0: Off; 1: On
<U_Type_Stair>.CarriageWidth	Float	3.0	控制支撑梁的宽度。可动画
<U_Type_Stair>.direction	Integer	1	设置楼梯的方向。可动画
<U_Type_Stair>.GenerateCarriage	Integer	1	是否设置支撑梁。可动画
<U_Type_Stair>.GenerateInsideRailing	Integer	0	是否创建内扶手
<U_Type_Stair>.GenerateMapping	Integer	0	设置是否对楼梯应用默认的贴图坐标: 0: Off; 1: On
<U_Type_Stair>.GenerateOutsideRailing	Integer	0	是否创建外扶手
<U_Type_Stair>.GenerateStringers	Integer	0	设置是否创建侧弦。可动画
<U_Type_Stair>.length	Float	0.0	控制第一段楼梯的长度。可动画
<U_Type_Stair>.length2	Float	0.0	控制第二段楼梯的长度。可动画
<U_Type_Stair>.RailingHeight	Float	42.0	控制栏杆离台阶的高度。可动画
<U_Type_Stair>.RailingOffs	Float	2.0	控制栏杆离台阶端点的偏移。可动画
<U_Type_Stair>.RailingRadius	Float	1.0	控制栏杆的厚度。可动画
<U_Type_Stair>.RailingSegments	Integer	3	指定栏杆的分段数目。值越高,栏杆显示得越光滑。可动画
<U_Type_Stair>.StepCount	Integer	12	控制梯级竖板数。梯级竖板总是比台阶多一个。隐式梯级竖板位于上板和楼梯顶部台阶之间。可动画
<U_Type_Stair>.StepDepth	Float	0.0	当 StepDepth_X 设置为 On 时,控制台阶的深度。可动画
<U_Type_Stair>.StepDepth_X	Integer	0	是否需要设置台阶的深度: 0: Off; 1: On
<U_Type_Stair>.StepHeight	Float	0.0	控制梯级竖板的高度。可动画
<U_Type_Stair>.StepThickness	Float	2.0	控制台阶的厚度。可动画
<U_Type_Stair>.StepType	Integer	0	设置楼梯的类型: 0: Open (创建开放式的梯级竖板楼梯) 1: Closed (创建封闭式的梯级竖板楼梯) 2: Box (创建带有封闭式梯级竖板和两侧有封闭式侧弦的楼梯)
<U_Type_Stair>.Stepwidth	Float	39.0	控制楼梯的宽度,包括台阶和平台。可动画
<U_Type_Stair>.StringerDepth	Float	6.0	控制侧弦离地板的深度。可动画
<U_Type_Stair>.StringerOffset	Float	1.0	控制地板与侧弦的垂直距离。可动画
<U_Type_Stair>.StringerSpringFloor	Integer	1	控制侧弦是从地面开始,还是与第一个梯级竖板的开始平齐,或是否侧弦延伸到地面以下。使用偏移选项可以控制侧弦延伸到地面以下的量。可动画 0: Off; 1: On
<U_Type_Stair>.StringerWidth	Float	1.0	控制侧弦的宽度。可动画
<U_Type_Stair>.UpperOffset	Float	0.0	控制平台与第二段楼梯的距离,相应调整平台的长度。可动画

### 8.4.9 Geometry-Patch Objects (面片栅格对象)

下面列出了所有的 Patch 类对象：

QuadPatch (四边形面片)

TriPatch (三角形面片)

#### 8.4.9.1 QuadPatch: GeometryClass (四边形面片)

构造函数

QuadPatch ...

属性

属性名称	数据类型	默认值	说明
<QuadPatch>.length	Float	25.0	指定四边形面片的长度。可动画
<QuadPatch>.width	Float	25.0	指定四边形面片的宽度。可动画
<QuadPatch>.lengthsegs <QuadPatch>.Length_Segments	Integer	1	指定四边形面片长度方向的分段精度。可动画
<QuadPatch>.mapCoords	Boolean	False	当设置为 On 时, 为四边形面片对象指定默认的贴图坐标
<QuadPatch>.widthsegs <QuadPatch>.Width_Segments	Integer	1	指定四边形面片宽度方向的分段精度。可动画

#### 8.4.9.2 TriPatch: GeometryClass (三角形面片)

构造函数

TriPatch ...

属性

属性名称	数据类型	默认值	说明
<TriPatch>.length	Float	25.0	指定三角形面片的长度。可动画
<TriPatch>.width	Float	25.0	指定三角形面片的宽度。可动画
<TriPatch>.mapCoords	Boolean	False	当设置为 On 时, 为三角面片对象指定默认的贴图坐标

### 8.4.10 Geometry-Particle Systems (粒子系统)

下面列出了所有粒子系统对象：

Blizzard (暴风雪粒子系统)

PArray (粒子阵列粒子系统)

PCloud (粒子云粒子系统)

Snow (雪粒粒子系统)

Spray (喷射粒子系统)

### SuperSpray (超级喷射粒子系统)

#### 8.4.10.1 粒子系统通用属性、操作符和方法

在 MAXScript 里共有五个函数用于粒子系统，用来获取或设置单独的粒子信息。这些函数返回或设置当前 at time 关联语句下或当前 Time Slider 显示的特定时间的粒子信息。下面是一个简单的关于 3ds max 系统粒子系统工作原理的描述，以帮助读者理解这些函数。

在任一时间，都会有一个包含了当前时间下所有活动粒子的数组：Particles。在这个数组里，每个粒子都由序号来标识。当某一粒子“死”了时，也许过了一段时间，它又“活”了，但它在 Particles 数组里的序号可能与原先不一样了，看起来像是一个新的粒子。在一个动画过程中，同一粒子可能“死”、“活”好几次，同时它们的序号也会不断变化。在下面的函数中，有的利用序号来标识粒子，在动画过程中，同一个序号可能代表不同的粒子。我们可以通过查看粒子的属性.age 来判别这一点：当.age 的值变为 0 时，我们就知道这一序号下的粒子已经变成另一个粒子了。如果在某一时间粒子处于非活动状态，利用下面的函数来获取这些粒子的属性.position、.velocity 和.age 都会返回值 undefined。这也告诉用户粒子正处在非活动状态。

下面是粒子系统函数：

##### 1. particleCount <particlesys\_Node>

返回指定粒子<particlesys\_Node>当前的粒子总数。数据类型为 Integer。如果场景目前没有被渲染，这个数值就是要在视窗里绘制的粒子数量；而如果场景目前正被渲染，返回值可能是渲染粒子数量或视窗粒子数量，取决于指定粒子此时是否正在被渲染。

##### 2. particleSize <particlesys\_Node>

返回一个 Float 值，包含指定粒子的大小。本参数是在粒子系统的 Parameter 卷展栏里设置的。粒子系统里的所有粒子只能有同一个大小。但如果将别的对象与粒子连接，当然可以将连接对象的大小设置成任意大小。

##### 3. particlePos <particlesys\_Node> <particle\_index\_Integer>

返回一个 Point3 值，包含指定粒子系统里指定序号粒子的当前坐标值。粒子序号从 1 开始。返回的坐标值为在当前工作坐标系下。如果粒子正处在非活动状态，本函数返回 undefined。

##### 4. particleVelocity <particlesys\_Node> <particle\_index\_Integer>

返回一个 Point3 值，包含指定粒子系统里指定序号粒子的当前速度值。粒子序号从 1 开始。返回的矢量为在当前工作坐标系下。如果粒子正处在非活动状态，本函数返回 undefined。

##### 5. particleAge <particlesys\_Node> <particle\_index\_Integer>

返回一个 Time 值，包含指定粒子系统里指定序号粒子的当前年龄值，从该粒子开始存在的那一帧开始计算。粒子序号从 1 开始。如果粒子正处在非活动状态，本函数返回 undefined。

#### 8.4.10.2 Blizzard: GeometryClass (暴风雪粒子系统)

##### 构造函数

Blizzard ...

## 属性

属性名称	数据类型	默认值	说明
Basic Parameters (基本参数)			
<Blizzard>.Emitter_Width	Float	0.0	设置发射器图标在视图中的显示尺寸。可动画
<Blizzard>.Emitter_Length	Boolean	False	设置是否在视图中隐藏发射器的图标
<Blizzard>.emitterHidden	Integer	0	设置粒子图在视图中的显示类型: 0: Dots (视图中粒子显示为小点) 1: Ticks (在视图中将粒子将显示为十字标记) 2: Mesh (视图中粒子将显示为网格对象, 该选项会降低视图的更新显示速度) 3: Bbox (视图中将实例几何体类型的粒子对象显示为边界盒)
<Blizzard>.viewType	Float	10.0	设置粒子显示百分比
<Blizzard>.viewPercent	Float	10.0	设置粒子显示百分比
<Blizzard>.Percentage_Displayed	Boolean	False	
Particle Generation (粒子生成)			
<Blizzard>.quantityMethod	Integer	0	设置粒子数量的方式: 0: Use rate (在动画的每一帧中发射相同数量的粒子) 1: Use total (指定在粒子系统寿命周期中发射粒子的总量) 提示: 对一个连续的粒子流使用 Use Rate 比较好, 而对短时间内的粒子爆炸, 使用 Use Total 比较好
<Blizzard>.Birth_Rate	Integer	10	设置发射器每帧喷射的粒子数量。可动画
<Blizzard>.Total_Number	Integer	100	设置发射器喷射的粒子总量
<Blizzard>.speed	Float	10.0	设置在粒子寿命开始时的速度, 单位为每一帧粒子的移动距离。可动画
<Blizzard>.Speed_Variation	Float	0.0	设置每个粒子初始速度的变化量。可动画, 百分比
<Blizzard>.tumble	Float	0.0	设置粒子随机旋转量。可动画
<Blizzard>.Tumble_Rate	Float	0.0	设置粒子旋转时的速度。可动画
<Blizzard>.Emitter_Start	Time	0f	设置粒子开始出现的动画帧
<Blizzard>.Emitter_Stop	Time	30f	设置粒子被发射完的动画帧
<Blizzard>.Display_Until	Time	100f	设置在哪个动画帧粒子全部消亡
<Blizzard>.life	Time	30f	设置每个粒子从被发射到消亡所持续的帧数。可动画
<Blizzard>.Life_Variation	Time	0f	设置粒子寿命随机增加或减少的帧数。可动画

(续表)

属性名称	数据类型	默认值	说明
<Blizzard>.subsampleCreationTime	Boolean	False	当设置为 On 时, 通过为运动附加时间偏移的方式, 避免粒子在一般帧处理时产生的粒子团堆积不流畅的效果
<Blizzard>.subsampleEmitterTranslation	Boolean	True	如果当前发射器被指定了空间平移动画, 将该属性设置为 True 可以避免粒子团堆积不流畅的效果, 以创建平滑螺旋发射粒子的效果
<Blizzard>.subsampleEmitterRotation	Boolean	True	如果当前发射器被指定了旋转动画, 将该属性设置为 True 可以避免粒子团堆积不流畅的效果, 以创建平滑螺旋发射粒子的效果
<Blizzard>.size	Float	1.0	设置所有粒子的目标尺寸。可动画
<Blizzard>.Size_Variation	Float	0.0	设置粒子尺寸变化的百分比。可动画, 百分数
<Blizzard>.Growth_Time	Time	10f	设置粒子从最小尺寸增长到目标尺寸所需要的帧数
<Blizzard>.Fade_Time	Time	10f	设置粒子从目标尺寸萎缩到消亡所持续的帧数
<Blizzard>.seed <Blizzard>.Random_Seed	Integer	0	不同的种子数可以使粒子产生随机的变化, 避免一模一样的形态
Particle Type (粒子类型)			
<Blizzard>.particleType	Integer	0	设置粒子类型: 0: Standard particle (标准粒子, 如三角形、立方体等) 1: Metaparticles (变形球粒子, 粒子在喷射或流动过程中可以互相碰撞融合, 用于模拟真实的液体粒流效果) 2: Instanced Geometry (实例几何体, 在场景中选择一个现有的对象(或链接层级对象, 成组对象等)作为粒子)
<Blizzard>.standardParticle	Integer	0	设置标准粒子类型: 0: Triangle (将每个粒子渲染输出为三角形, 如果为粒子指定了噪波不透明贴图, 可用于模拟水蒸气或烟尘) 1: Cube (将每个粒子渲染输出为立方体) 2: Special (将每个粒子渲染输出为交叉面, 每个粒子由三个垂直交叉的二维平面构成, 可以为这些面指定材质) 3: Facing (将每个粒子渲染输出为正四边形面片, 这些面片总是朝向于当前的渲染视图, 可以为面片指定不透明贴图, 模拟雪花或气泡的效果) 4: Constant (将每个粒子渲染输出为等大的圆形面片, 不论面片与摄影机相距多远。这些面片保持相同的渲染像素尺寸) 5: Tetra (将每个粒子渲染输出为贴图的四面体, 可用于模拟水滴或火花的效果) 6: Sixpoint (将每个粒子渲染输出为平面六角星) 7: Sphere (将每个粒子渲染输出为球体)

(续表)

属性名称	数据类型	默认值	说明
<Blizzard>.Metaparticle_Tension	Float	1.0	设置变形球粒子的张力。指定粒子的致密程度，粒子的张力越大，粒子本身越致密，也越容易与其他粒子融合在一起。可动画
<Blizzard>.Metaparticle_Tension_Variation	Float	1.0	设置张力效果变化的百分比。可动画，百分数
<Blizzard>.metaballRenderCoarseness <Blizzard>.Metaparticle_Coarseness	Float	0.5	设置变形球粒子计算的粗糙度，粗糙度数值越高，融合的计算量越小；如果粗糙度数值设置得过高，不产生粒子融合效果；如果粗糙度数值设置得过低，会耗费大量的计算时间。可动画，百分数
<Blizzard>.metaballViewCoarseness	Float	1.0	设置变形球粒子在视图中显示的粗糙度。可动画
<Blizzard>.metaballAutoCoarseness	Boolean	True	当设置为 On 时，会自动打开 Rendering Coarseness，粗糙程度基于粒子的大小，而 Viewport Coarseness 被设为 Rendering Coarseness 的约两倍大小
<Blizzard>.Bubble_Amplitude	Integer	0	设置是否启用一个相连的水滴模式：0: Off; 1: On
<Blizzard>.instancingObject	Node	undefined	设置实例对象
<Blizzard>.instanceSubTree	Boolean	False	设置是否使用层级树。当设置为 On 时，如果当前选定对象是一个层级链接系统中的一部份，所有层级链接系统中的其他对象都作为粒子对象；如果当前选定的对象是一个组对象，整个群组中的子级对象都作为粒子的子对象
<Blizzard>.instanceKeyOffsetType	Integer	0	设置动画偏移关键帧： 0: None (每个粒子复制原始对象的动画关键帧，所以在动画的任何一帧中所有粒子对象都保持与原始对象相同的动画进程) 1: Birth (第一个出生的粒子是粒子出生时源对象当前动画的实例。每个后续粒子将使用相同的开始时间设置动画) 2: Random (如果帧偏移设置为 0，此选项相当于“无”。每个粒子使用与源对象出生时相同的动画出生，但是帧根据帧偏移微调器中的值进行随机偏移)
<Blizzard>.instanceFrameOffset <Blizzard>.Animation_Offset_Amount	Integer	0	设置相对于原始对象当前时间点动画进行帧偏移的数量
<Blizzard>.mappingType	Integer	0	为粒子对象指定材质和贴图设置方式： 0: Time (时间); 1: Distance (距离)
<Blizzard>.Mapping_Time_Base	Time	30f	当属性.mappingType=0 时，设置粒子对象从诞生开始到完成一个贴图所需要的帧数。可动画
<Blizzard>.Mapping_Distance_Base	Float	100.0	当属性.mappingType=1 时，设置粒子对象从诞生开始到完成一个贴图所经过的距离。可动画

(续表)

属性名称	数据类型	默认值	说明
<Blizzard>.MaterialSource	Integer	0	设置更新粒子材质的来源类型: 0: Icon (粒子使用当前为粒子系统图标指定的材质) 1: Instanced Geometry (粒子使用为实例几何体指定的材质)
<b>Rotation and Collision (旋转和碰撞)</b>			
<Blizzard>.Spin_Time	Time	30f	设置粒子对象旋转一周所持续的帧数。如果设置为0，则粒子对象不旋转。可动画
<Blizzard>.Spin_Time_Variation	Float	0.0	设置旋转速度变化的百分比。可动画，百分数
<Blizzard>.Spin_Phase	Float	0.0	设置粒子对象的初始旋转相位。可动画，Angle
<Blizzard>.Spin_Phase_Variation	Float	0.0	设置初始相位变化的百分比。可动画，百分数
<Blizzard>.spinAxisType	Integer	0	设置旋转轴控制方式: 0: Random (每个粒子的旋转轴向是随机指定的) 1: User defined (可以自定义一个旋转的轴向)
<Blizzard>.X_Spin_Vector	Float	1.0	设置 X 轴向方向的旋转角度。可动画
<Blizzard>.Y_Spin_Vector	Float	0.0	设置 Y 轴向方向的旋转角度。可动画
<Blizzard>.Z_Spin_Vector	Float	0.0	设置 Z 轴向方向的旋转角度。可动画
<Blizzard>.Spin_Axis_Variation	Float	0.0	设置在每个旋转轴向上，旋转角度的变化量。可动画，Angle
<Blizzard>.Interparticle_Collisions_On	Integer	0	设置粒子运动过程中的相交碰撞计算是否有效: 0: Off; 1: On
<Blizzard>.Interparticle_Collision_Steps	Integer	2	设置渲染时每帧计算间隔。较高的数值可以使碰撞计算更为精确，碰撞效果更为真实，但是会减慢计算的速度
<Blizzard>.Interparticle_Collision_Bounce	Float	100.0	设置碰撞后粒子反弹的速度百分比(与原速度相比)。可动画，百分数
<Blizzard>.Interparticle_Collision_Bounce_Variation	Float	0.0	设置碰撞后粒子反弹速度变化量百分比，适用于所有粒子。可动画，百分数
<b>Object Motion Inheritance (对象运动继承)</b>			
<Blizzard>.motionInfluence	Float	100.0	设置在粒子生成时，继承基于对象的发射器运动的粒子百分比。可动画
<Blizzard>.Object_Motion_Inheritance			
<Blizzard>.motionMultiplier	Float	1.0	设置增加或减少发射器运动对粒子运动的影响，可为正数或负数。可动画
<Blizzard>.Object_Motion_Multiplier			
<Blizzard>.motionVariation	Float	0.0	设置倍增器数值变化的百分比。可动画，百分数
<Blizzard>.Object_Motion_Multiplier_Variation			

(续表)

属性名称	数据类型	默认值	说明
<b>Particle Spawn (粒子繁殖)</b>			
<Blizzard>.spawnType	Integer	0	设置当粒子碰撞到导向面后所发生的情况: 0: None (反弹或者粘连到导向面; 当粒子消亡后就自然消失) 1: Die after collision (碰撞后消亡) 2: Spawn on collision (碰撞后产生繁殖效果) 3: Spawn on death (消亡时产生繁殖效果) 4: Spawn trails (在现有粒子寿命的每个帧, 从相应粒子繁殖粒子, 产生的子级粒子基础方向与父级粒子的速率方向相反)
<Blizzard>.Die_X_frames_after_collision	Time	0f	设置粒子碰撞到导向面后持续存在的时间。如果设置为0, 当粒子碰撞到导向面之后立即消失。可动画
<Blizzard>.Die_X_frames_after_collision_variation	Float	0.0	设置每个粒子持续时间变化的百分比。可动画, 百分数
<Blizzard>.Spawn_Affects	Integer	100	设置在粒子系统中繁殖粒子的百分比
<Blizzard>.Spawn_Multiplier_Variation	Float	0.0	设置倍增数值在每一帧中变化的百分比。百分数
<Blizzard>.Spawn_Direction_Chaos	Float	0.0	设置子级粒子运动方向沿父级粒子运动方向变化的混乱度。如果设置为0, 不发生方向变化; 如果设置为100, 子级粒子可以随机沿各个方向变化; 如果设置为50, 子级粒子随机运动, 但与父级对象运动方向的偏移角度不超过90度。可动画
<Blizzard>.Spawn_Speed_Chaos	Float	0.0	设置子级粒子相对于父级粒子运动速度变化的百分比, 如果设置为0, 表示运动速度不发生变化; 可动画
<Blizzard>.spawnSpeedType	Integer	0	设置速度混乱度类型: 0: Slow (随机应用速度因子, 减慢繁殖的粒子的速度) 1: Fast (根据速度因子随机加快粒子的速度) 2: Both (根据速度因子, 有些粒子加快速度, 有些粒子减慢速度)
<Blizzard>.spawnInheritVelocity	Boolean	False	当设置为On时, 指定子级粒子继承父级粒子的运动速度, 并在该速度基础上进行随机的速度变化
<Blizzard>.spawnSpeedFixed	Boolean	False	设置是否指定固定的速度变化量
<Blizzard>.lifespanValueQueue	Array	#()	设置寿命值序列。在该数组中可以为繁殖创建的子级粒子指定新的寿命数值, 而不是继承其父级粒子的寿命数值
<Blizzard>.objectMutationQueue	Array	#()	设置实例对象类型粒子系统的实例对象数组

#### 8.4.10.3 PArray: GeometryClass (粒子阵列粒子系统)

构造函数

Parray ...

## 属性

属性名称	数据类型	默认值	说明
Basic Parameters (基本参数)			
<PArray>.emitter	Node	undefined	设置发射器物体
<PArray>.formation <PArray>.Emitter_Distribution	Integer	0	<p>设置粒子在对象发射器表面的分布方式</p> <p>0: Over Entire Surface (在对象基础发射器的全部表面随机发射粒子)</p> <p>1: Along Visible (沿发射器对象的所有可见边随机发射粒子)</p> <p>2: At All Vertices (从发射器对象的所有顶点发射粒子)</p> <p>3: At Distinct Points (在发射器表面随机指定的一些点上发射粒子)</p> <p>4: At face centers (从发射器对象表面每三个结构面的中心发射粒子)</p>
<PArray>.numDistinctPoints <PArray>.Number_of_Emitters	Integer	20	当.Emitter_Distribution=3 时, 设置发射器表面发射点的总数
<PArray>.Use_Selected_Subobjects	Integer	0	对 Mesh-based 发射器, 本参数设置是否将粒子源限制为子对象选集。可动画 0: Off; 1: On
<PArray>.iconsize <PArray>.Emitter_Width	Float	20.0	设置发射器图标在视图中的显示尺寸。可动画
<PArray>.iconHidden	Boolean	False	设置是否在视图中隐藏发射器的图标
<PArray>.viewType	Integer	0	设置粒子图在视图中的显示类型: 0: Dots; 1: Ticks; 2: Mesh; 3: Bbox
<PArray>.viewPercent <PArray>.Percentage_Displayed	Float	10.0	设置粒子显示百分比。百分数
Particle Generation (粒子生成)			
<PArray>.quantityMethod	Integer	0	<p>设置粒子数量的方式:</p> <p>0: Use rate (在动画的每一帧中发射相同数量的粒子)</p> <p>1: Use total (在粒子系统生命周期中发射粒子的总量)</p>
<PArray>.Birth_Rate	Integer	10	设置发射器每帧喷射的粒子数量。可动画
<PArray>.Total_Number	Integer	100	设置发射器喷射的粒子总量
<PArray>.speed	Float	10.0	设置在粒子生命周期中, 粒子在每一帧中的移动距离。可动画
<PArray>.Speed_Variation	Float	0.0	设置粒子初始速度的变化量。可动画, 百分数
<PArray>.Divergence_Angle	Float	10.0	设置粒子运动方向与发射器法线方向的角度变化量。可动画, Angle

(续表)

属性名称	数据类型	默认值	说明
<PArray>.Emitter_Start	Time	0f	设置粒子开始出现的动画帧
<PArray>.Emitter_Stop	Time	30f	设置粒子被发射完的动画帧
<PArray>.Display_Until	Time	100f	设置在哪个动画帧粒子全部消亡
<PArray>.life	Time	30f	设置每个粒子从被发射到消亡所持续的帧数。可动画
<PArray>.Life_Variation	Time	0f	设置粒子寿命随机增加或减少的帧数。可动画
<PArray>.subSampleCreationTime	Boolean	False	当设置为On时，通过为运动方程附加时间偏移的方式，避免粒子在一般帧处理时产生的粒子团堆积不流畅的效果
<PArray>.subSampleEmitterTranslation	Boolean	True	如果当前发射器被指定了空间平移动画，设置为True时可以避免粒子团堆积不流畅的效果，以创建平滑螺旋发射粒子的效果
<PArray>.subSampleEmitterRotation	Boolean	True	如果当前发射器被指定了旋转动画，设置为True时可以避免粒子团堆积不流畅的效果，以创建平滑螺旋发射粒子的效果
<PArray>.size	Float	1.0	设置所有粒子的目标尺寸。可动画
<PArray>.Size_Variation	Float	0.0	设置粒子尺寸变化的百分比。可动画，百分数
<PArray>.Growth_Time	Time	10f	设置粒子从最小尺寸增长到目标尺寸所需要的帧数
<PArray>.Fade_Time	Time	10f	设置粒子从目标尺寸萎缩到消亡所持续的帧数
<PArray>.seed <PArray>.Random_Seed	Integer	0	重新设置随机数根值
Particle Type (粒子类型)			
<PArray>.particleType	Integer	0	设置粒子类型： 0: Standard particle (标准粒子) 1: Metaparticles (变形球粒子) 2: Object fragments (对象碎片) 3: Instanced geometry (实例几何体)
<PArray>.standardParticle	Integer	0	设置标准粒子类型： 0: Triangle (三角形) 1: Cube (立方体) 2: Special (特殊) 3: Facing (面) 4: Constant (恒定) 5: Tetra (四面体) 6: Sixpoint (六角星) 7: Sphere (球体)
<PArray>.Metaparticle_Tension	Float	1.0	设置变形球粒子的张力。指定粒子的致密程度，粒子的张力越大，粒子本身越致密，也越容易与其他粒子融合在一起。可动画

(续表)

属性名称	数据类型	默认值	说明
<PArray>.Metaparticle_Tension_Variation	Float	0.0	设置张力效果变化的百分比。可动画，百分数
<PArray>.metaballRenderCoarseness <PArray>.Metaparticle_Coarseness	Float	0.5	设置变形球粒子计算的粗糙度，粗糙度数值越高，融合的计算量越小；如果粗糙度数值设置得过高，不产生粒子融合效果；如果粗糙度数值设置得过低，会耗费大量的计算时间。可动画，百分数
<PArray>.metaballViewCoarseness	Float	1.0	设置变形球粒子在视图中显示的粗糙度
<PArray>.metaballAutoCoarseness	Boolean	True	设置变形球粒子渲染输出是否计算粗糙度。如果设为 True，渲染时计算粗糙度，大小为粒子大小，视图粗糙度为渲染粗糙度的两倍
<PArray>.fragmentMethod	Integer	0	设置对象碎片粒子的方式： 0: All Faces (对象的每一个面都变成一个三角形粒子) 1: Number of Chunks(对象破裂成不规则的碎片) 2: Smoothing Angle (碎片按属性.FragSmoothing Angle 规定的面法线之间的角度生成，一般而言，角度值越大，碎片数越少)
<PArray>.Fragment_Thickness	Float	1.0	设置碎片的厚度。厚度为 0 时，碎片为单面没有厚度；厚度大于 0 时，碎片在破碎时按指定值进行挤出
<PArray>.fragChunkMinimum <PArray>.Fragment_Count	Integer	100	设置对象被分解的碎片数量
<PArray>.fragSmoothingAngle	Float	0.0	设置光滑角度的数值
<PArray>.instancingObject	Node	undefined	指定实例对象
<PArray>.instanceSubTree	Boolean	False	设置是否使用层级树。 当设为 True 时，如果当前选定对象是一个层级链接系统中的一部份，所有层级链接系统中的其他对象都作为粒子对象；如果当前选定的对象是一个对象组，整个群组中的子级对象都作为粒子的子对象
<PArray>.instanceKeyOffsetType	Integer	0	设置动画偏移关键帧： 0: None (无) 1: Birth (出生) 2: Random (随机)
<PArray>.instanceFrameOffset <PArray>.Animation_Offset_Amount	Integer	0	设置相对于原始对象当前时间点动画进行帧偏移的数量

(续表)

属性名称	数据类型	默认值	说明
<PArray>.mappingType	Integer	0	为粒子对象指定材质和贴图设置方式: 0: Time (时间); 1: Distance (距离)
<PArray>.Mapping_Time_Base	Time	30f	当属性.mappingType=0时, 设置粒子对象从诞生开始到完成一个贴图所持续的帧数。可动画
<PArray>.Mapping_Distance_Base	Float	100.0	当属性.mappingType=0时, 设置粒子对象从诞生开始到完成一个贴图所经过的距离。可动画
<PArray>.MaterialSource	Integer	0	设置更新粒子材质的来源类型: 0: Icon (粒子使用当前为粒子系统图标指定的材质) 1: Picked emitter (粒子使用为分布对象指定的材质) 2: Instanced geometry (粒子使用为实例几何体指定的材质)
<PArray>.fragOutsideMatID	Integer	0	设置碎片的表面材质 ID 号码。默认值为 0, 可以使碎片外面材质与原始对象的材质保持一致
<PArray>.fragEdgeMatID	Integer	2	设置碎片的边缘材质 ID 号码
<PArray>.fragBackMatID	Integer	3	设置碎片的背面材质 ID 号码
Rotation and Collision (旋转和碰撞)			
<PArray>.Spin_Time	Time	0f	设置粒子对象旋转一周所持续的帧数, 如果设置为 0, 则粒子对象不旋转。可动画
<PArray>.Spin_Time_Variation	Float	0.0	设置旋转速度变化的百分比。可动画, 百分数
<PArray>.Spin_Phase	Float	0.0	设置粒子对象的初始旋转相位。可动画, Angle
<PArray>.Spin_Phase_Variation	Float	0.0	设置初始相位变化的百分比。可动画, 百分数
<PArray>.spinAxisType	Integer	0	设置旋转轴控制方式: 0: Random (每个粒子的旋转轴向是随机指定的) 1: User defined (可以自定义一个旋转的轴向)
<PArray>.Blur_Stretch	Integer	0	设置沿粒子运动的方向对粒子对象进行的挤压变形。最终挤压效果还受到粒子运动的影响。本属性仅在选择Direction of Travel/Mblur选项时才有用。可动画
<PArray>.X_Spin_Vector	Float	1.0	设置 X 轴向方向的旋转角度。可动画
<PArray>.Y_Spin_Vector	Float	0.0	设置 Y 轴向方向的旋转角度。可动画
<PArray>.Z_Spin_Vector	Float	0.0	设置 Z 轴向方向的旋转角度。可动画
<PArray>.Spin_Axis_Variation	Float	0.0	设置在每个旋转轴向上, 旋转角度的变化量。可动画, Angle
<PArray>.Interparticle_Collisions_On	Integer	0	设置粒子运动过程中的相交碰撞计算是否有效

(续表)

属性名称	数据类型	默认值	说明
<PArray>.Interparticle_Collision_Steps	Integer	2	设置每帧计算间隔。较高的数值可以使碰撞计算更为精确，碰撞效果更为真实，但是会减慢计算的速度
<PArray>.Interparticle_Collision_Bounce	Float	100.0	设置碰撞后粒子反弹的速度。可动画，百分数
Object Motion Inheritance (对象运动继承)			
<PArray>.motionInfluence	Float	100.0	设置粒子受发射器运动影响的百分比，如果指定为 100，则所有粒子对象都随同发射器一起运动，如果指定为 0，则所有粒子对象都不受发射器运动的影响。可动画
<PArray>.Object_Motion_Inheritance			
<PArray>.motionMultiplier	Float	1.0	设置增加或减少发射器运动对粒子对象的影响。可动画
<PArray>.Object_Motion_Multiplier			
<PArray>.motionVariation	Float	0.0	设置倍增器数值变化的百分比。可动画，百分数
<PArray>.Object_Motion_Multiplier_Variation			
Bubble Motion (气泡运动)			
<PArray>.Bubble_Amplitude	Float	0.0	设置粒子在晃动过程中，相对于运动路径的偏移距离。可动画
<PArray>.Bubble_Amplitude_Variation	Float	0.0	设置每个粒子振幅变化的百分比。可动画，百分数
<PArray>.Bubble_Period	Time	100000f	设置粒子完成一次完整晃动(一个波峰和一个波谷)所持续的帧数。一般设置为 20~30 可以产生较好的气泡运动效果。可动画
<PArray>.Bubble_Period_Variation	Float	0.0	设置每个粒子周期变化的百分比。可动画，百分数
<PArray>.Bubble_Phase	Float	0.0	设置粒子对象开始晃动之前，相对于运动路径的偏移距离。可动画，Angle
<PArray>.Bubble_Phase_Variation	Float	0.0	设置每个粒子相位变化的百分比。可动画，百分数
Particle Spawn (粒子繁殖)			
<PArray>.spawnType	Integer	0	设置当粒子碰撞到导向面后所发生的情况： 0: None (无) 1: Die after collision (碰撞后消亡) 2: Spawn on collision (碰撞后繁殖) 3: Spawn on death (碰撞后消亡) 4: Spawn trails (繁殖拖尾)
<PArray>.Die_X_frames_after_collision	Time	0f	设置粒子碰撞到导向面后持续存在的时间。如果设置为 0，当粒子碰撞到导向面之后立即消失。可动画
<PArray>.Die_X_frames_after_collision_variation	Float	0.0	设置每个粒子持续时间变化的百分比。可动画，百分数

(续表)

属性名称	数据类型	默认值	说明
<PArray>.Spawn_Generations	Integer	1	设置在生成粒子之后的繁殖次数。如：将属性.Spawn_Generations 设为 1，在每个粒子的生存期之外会繁殖 1 次
<PArray>.Spawn_Affects	Integer	100	设置在粒子系统中繁殖粒子的百分比
<PArray>.Spawn_Multiplier	Integer	1	设置倍增每个粒子的繁殖个数
<PArray>.Spawn_Multiplier_Variation	Float	0.0	设置倍增数值在每一帧中变化的百分比。可动画，百分数
<PArray>.Spawn_Direction_Chaos	Float	0.0	设置子级粒子运动方向沿父级粒子运动方向变化的混乱度。如果设置为 0，不发生方向变化；如果设置为 100，子级粒子可以随机沿各个方向变化；如果设置为 50，子级粒子随机运动，但与父级对象运动方向的偏移角度不超过 90 度。可动画，百分数
<PArray>.Spawn_Speed_Chaos	Float	0.0	设置子级粒子相对于父级粒子运动速度变化的百分比，如果设置为 0，表示运动速度不发生变化；可动画
<PArray>.spawnSpeedType	Integer	0	设置速度混乱度类型：0: Slow (慢); 1: Fast (快); 2: Both (二者均可)
<PArray>.spawnInheritVelocity	Boolean	False	当设置为 On 时，指定子级粒子继承父级粒子的运动速度，并在该速度基础上进行随机的速度变化
<PArray>.spawnSpeedFixed	Boolean	False	设置是否指定固定的速度变化量
<PArray>.Spawn_Scale_Chaos	Float	0.0	设置子级粒子相对于父级粒子运动速度变化量变化的百分比。如果设置为 0，表示运动速度不发生变化；可动画
<PArray>.spawnScaleType	Integer	0	设置放缩变化类型： 0: Slow (依据因数的设置，子级粒子相对于父级粒子运动速度随机变慢) 1: Fast (依据因数的设置，子级粒子相对于父级粒子运动速度随机变快) 2: Both (依据因数的设置，一些子级粒子相对于父级粒子运动速度随机变慢，一些子级粒子相对于父级粒子运动速度随机变快)
<PArray>.spawnScaleFixed	Boolean	False	当设置为 On 时，指定一个固定的放缩变化量，不采用 factor 参数指定的尺寸随机变化范围
<PArray>.lifespanValueQueue	Array	#()	子级粒子寿命值数组 (元素为整数)。在该数组中可以为繁殖创建的子级粒子指定新的寿命数值，而不是继承其父级粒子的寿命数值
<PArray>.Spawn_Lifespan	Integer	0	设置子级粒子初始寿命数值
<PArray>.Interparticle_Collision_Bounce_Variation	Float	0.0	设置寿命数值变化的百分比。可动画，百分数
<PArray>.objectMutationQueue	Array	#()	设置实例对象类型粒子系统的实例对象数组

## 8.4.10.4 PCloud: GeometryClass (粒子云粒子系统)

构造函数

Pcloud ...

属性

属性名称	数据类型	默认值	说明
<b>Basic Parameters (基本参数)</b>			
<PCloud>.emitter	Node	undefined	设置发射器物体
<PCloud>.formation	Integer	0	设置粒子构成方式: 0: Box Emitter (长方体发射器) 1: Sphere Emitter (球体发射器) 2: Cylinder Emitter (圆柱体发射器) 3: Object-based Emitter (对象基础发射器)
<PCloud>.Emitter_Rad_Len	Float	0.0	设置球体或圆柱体发射器图标的半径, 或长方体发射器图标的高度。可动画
<PCloud>.Emitter_Width	Float	0.0	设置长方体发射器图标在视图中显示的宽度。可动画
<PCloud>.Emitter_Height	Float	0.0	设置长方体或圆柱体发射器图标的高度。可动画
<PCloud>.emitterHidden	Boolean	False	设置在视图中是否隐藏发射器的图标。注意: 发射器的图标不会被着色输出
<PCloud>.viewType	Integer	0	设置粒子图在视图中的显示类型: 0: Dots; 1: Ticks; 2: Mesh; 3: Bbox
<PCloud>.viewPercent <PCloud>.Percentage_Displayed	Float	0.0	设置出现在视图中的粒子百分比, 减少该数值可以加快视图更新显示的速度, 不会对最终的渲染输出产生影响。百分数
<b>Particle Generation (粒子生成)</b>			
<PCloud>.quantityMethod	Integer	0	设置粒子数量的方式: 0: Use rate (在动画的每一帧中发射相同数量的粒子) 1: Use total (指定在粒子系统寿命周期中发射粒子的总量)
<PCloud>.Birth_Rate	Integer	10	设置发射器每帧喷射的粒子数量。可动画
<PCloud>.Total_Number	Integer	100	设置发射器喷射的粒子总量
<PCloud>.speed	Float	0.0	设置在粒子寿命周期中, 粒子在每一帧中的移动距离。可动画
<PCloud>.Speed_Variation	Float	0.0	设置粒子初始速度的变化量。可动画, 百分数
<PCloud>.motionType	Integer	0	设置粒子的运动类型: 0: Random direction (随机指定发射粒子的方向) 1: Enter vector (依据输入的 X、Y、Z 轴向角度发射粒子) 2: Reference object (依据选定参考对象的局部坐标 Z 轴方向发射粒子)

(续表)

属性名称	数据类型	默认值	说明
<PCloud>.Direction_Vector_X	Float	1.0	设置X轴向角度发射的粒子。可动画
<PCloud>.Direction_Vector_Y	Float	0.0	设置Y轴向角度发射的粒子。可动画
<PCloud>.Direction_Vector_Z	Float	0.0	设置Z轴向角度发射的粒子。可动画
<PCloud>.motionReferenceObject	Node	undefined	设置粒子发射的实例对象
<PCloud>.directionVariation <PCloud>.Speed_Direction_Variation	Float	0.0	当选择Direction Vector或Reference Object选项时设置粒子发射方向的变化量。可动画，百分数
<PCloud>.Emitter_Start	Time	0f	设置粒子开始出现的动画帧
<PCloud>.Emitter_Stop	Time	0f	设置粒子被发射完的动画帧
<PCloud>.Display_Until	Time	100f	设置粒子全部消亡的动画帧
<PCloud>.life	Time	101f	设置每个粒子从被发射到消亡所持续的帧数。可动画
<PCloud>.Life_Variation	Time	0f	设置粒子寿命随机增加或减少的帧数。可动画
<PCloud>.size	Float	1.0	设置所有粒子的目标尺寸。可动画
<PCloud>.Size_Variation	Float	0.0	设置粒子尺寸变化的百分比。可动画，百分数
<PCloud>.Growth_Time	Time	0f	设置粒子从最小尺寸增长到目标尺寸所需要的帧数
<PCloud>.Fade_Time	Time	0f	设置粒子从目标尺寸萎缩到消亡所持续的帧数
<PCloud>.seed <PCloud>.Random_Seed	Integer	0	不同的种子数可以使其产生随机的变化，避免一模一样的形态
Particle Type(粒子类型)			
<PCloud>.particleType	Integer	0	设置粒子类型： 0: Standard particle(标准粒子) 1: Metaparticles(变形球粒子) 2: Instanced geometry(实例几何体)
<PCloud>.standardParticle	Integer	0	设置标准粒子类型： 0: Triangle(三角形) 1: Cube(立方体) 2: Special(特殊) 3: Facing(面) 4: Constant(恒定) 5: Tetra(四面体) 6: Sixpoint(六角星) 7: Sphere(球体)
<PCloud>.Metaparticle_Tension	Float	1.0	设置变形球粒子的张力。指定粒子的致密程度，粒子的张力越大，粒子本身越致密，也越容易与其他粒子融合在一起。可动画
<PCloud>.Metaparticle_Tension_Variation	Float	0.0	设置张力效果变化的百分比。可动画，百分数
<PCloud>.MetaballRenderCoarseness <PCloud>.Metaparticle_Coarseness	Float	0.5	设置变形球粒子计算的粗糙度，粗糙度数值越高，融合的计算量越小；如果粗糙度数值设置得过高，不产生粒子融合效果；如果粗糙度数值设置得过低，会耗费大量的计算时间

(续表)

属性名称	数据类型	默认值	说明
<PCloud>.metaballViewCoarsness	Float	1.0	设置变形球粒子在视图中显示的粗糙度
<PCloud>.metaballAutoCoarsness	Boolean	True	设置变形球粒子渲染输出是否计算粗糙度
<PCloud>.instancingObject	Node	undefined	设置实例对象
<PCloud>.instanceSubTree	Boolean	False	设置是否使用层级树。当设置为 On 时，如果当前选定对象是一个层级链接系统中的一部分，所有层级链接系统中的其他对象都作为粒子对象；如果当前选定的对象是一个成组对象，整个群组中的子级对象都作为粒子的子对象
<PCloud>.instanceKeyOffsetType	Integer	0	设置动画偏移关键帧： 0: None (无) 1: Birth (出生) 2: Random (随机)
<PCloud>.InstanceFrameOffset<PCloud>.Animation_Offset_Amount	Integer	0	设置相对于原始对象当前时间点动画进行帧偏移的数量
<PCloud>.mappingType	Integer	0	为粒子对象指定材质和贴图设置方式： 0: Time (时间); 1: Distance (距离)
<PCloud>.Mapping_Time_Base	Time	30f	当属性.mappingType=1 时，设置粒子对象从诞生开始到完成一个贴图所持续的帧数。可动画
<PCloud>.Mapping_Distance_Base	Float	100.0	当属性.mappingType=1 时，设置粒子对象从诞生开始到完成一个贴图所经过的距离
<PCloud>.MaterialSource	Integer	0	设置更新粒子材质的来源类型。可动画 0: Icon (粒子对象使用当前在场景中指定到粒子系统图标的材质) 1: Instanced geometry (使用实例几何体的材质作为粒子对象材质)
<b>Rotation and Collision (旋转和碰撞)</b>			
<PCloud>.Spin_Time	Time	0f	设置粒子对象旋转一周所持续的帧数。如果设置为 0，则粒子对象不旋转。可动画
<PCloud>.Mapping_Distance_Base	Float	100.0	设置粒子对象从诞生开始到完成一个贴图所经过的距离。可动画
<PCloud>.Spin_Time_Variation	Float	0.0	设置旋转速度变化的百分比。可动画，百分数
<PCloud>.Spin_Phase	Float	0.0	设置粒子对象的初始旋转相位。可动画，Angle
<PCloud>.Spin_Phase_Variation	Float	0.0	设置初始相位变化的百分比。可动画，百分数
<PCloud>.spinAxisType	Integer	0	设置旋转轴控制方式： 0: random(每个粒子的旋转轴向是随机指定的) 1: user defined (自定义旋转的轴向)
<PCloud>.Blur_Stretch	Integer	0	设置沿粒子运动的方向对粒子对象进行的挤压变形。当大于 0 时，粒子沿运动轴挤压，挤压大小取决于粒子速度。本属性仅在 Travel/Mblur 选项有效时才可用。可动画

(续表)

属性名称	数据类型	默认值	说明
<PCloud>.X_Spin_Vector	Float	1.0	设置 X 轴向方向的旋转角度。可动画
<PCloud>.Y_Spin_Vector	Float	0.0	设置 Y 轴向方向的旋转角度。可动画
<PCloud>.Z_Spin_Vector	Float	0.0	设置 Z 轴向方向的旋转角度
<PCloud>.Spin_Axis_Variation	Float	0.0	设置在每个旋转轴向上旋转角度的变化量。可动画, Angle
<PCloud>.Interparticle_Collisions_On	Integer	0	设置粒子运动过程中的相交碰撞计算是否有效: 0: Off; 1: On
<PCloud>.Interparticle_Collision_Steps	Integer	2	设置渲染时每帧之间碰撞计算间隔。较高的数值可以使碰撞计算更为精确, 碰撞效果更为真实, 但是会减慢计算的速度
<PCloud>.Interparticle_Collision_Bounce	Float	100.0	设置碰撞后粒子反弹速度与原速度的百分比。可动画, 百分数
<PCloud>.Interparticle_Collision_Bounce_Variation	Float	0.0	设置碰撞后粒子反弹速度的变化量。可动画, 百分数
Object Motion Inheritance (对象运动继承)			
<PCloud>.motionInfluence <PCloud>.Object_Motion_Inheritance	Float	100.0	设置粒子生成时受基于对象的发射器运动影响的百分比, 如果指定为 100, 则所有粒子对象都随同发射器一起运动, 如果指定为 0, 则所有粒子对象都不受发射器运动的影响。可动画
<PCloud>.motionMultiplier <PCloud>.Object_Motion_Multiplier	Float	1.0	设置增加或减少发射器运动对粒子运动的影响。可为正数或负数。可动画
<PCloud>.motionVariation <PCloud>.Object_Motion_Multiplier_Variation	Float	0.0	设置倍增器数值变化的百分比。可动画, 百分数
Bubble Motion (气泡运动)			
<PCloud>.Bubble_Amplitude	Float	0.0	设置粒子在晃动过程中, 相对于运动路径的偏移距离。可动画
<PCloud>.Bubble_Amplitude_Variation	Float	0.0	设置每个粒子振幅变化的百分比。可动画, 百分数
<PCloud>.Bubble_Period	Time	100000f	设置粒子完成一次完整晃动(一个波峰和一个波谷)所持续的帧数。一般设置为 20~30 可以产生较好的气泡运动效果。可动画
<PCloud>.Bubble_Period_Variation	Float	0.0	设置每个粒子周期变化的百分比。可动画, 百分数
<PCloud>.Bubble_Phase	Float	0.0	设置粒子对象开始晃动之前, 相对于运动路径的偏移角度。可动画, Angle
<PCloud>.Bubble_Phase_Variation	Float	0.0	设置每个粒子相位变化的百分比。可动画, 百分数
Particle Spawn (粒子繁殖)			
<PCloud>.spawnType	Integer	0	设置当粒子碰撞到导向面后所发生的情况: 0: None (无) 1: Die after collision (碰撞后消亡) 2: Spawn on collision (碰撞后繁殖) 3: Spawn on death (碰撞后消亡) 4: Spawn trails (繁殖拖尾)

(续表)

属性名称	数据类型	默认值	说明
<PCloud>.Die_X_frames_after_collision	Time	0f	设置粒子碰撞到导向面后持续存在的时间。如果设置为 0 (默认值), 粒子碰撞到导向面之后立即消失。可动画
<PCloud>.Die_X_frames_after_collision_variation	Float	0.0	当属性.Die_X_frames_after_collision>0 时, 设置每个粒子持续时间的变化百分比。可动画, 百分数
<PCloud>.Spawn_Generations	Integer	1	设置在生成粒子之后的繁殖次数。 如: 将.Spawn_Generations 设为 1, 在每个粒子的生存期之外会繁殖 1 次
<PCloud>.Spawn_Multiplier	Integer	1	设置粒子每次繁殖时繁殖个数的倍增系数
<PCloud>.Spawn_Direction_Chaos	Float	0.0	设置子级粒子运动方向沿父级粒子运动方向变化的混乱度。如果设置为 0, 不发生方向变化; 如果设置为 100, 子级粒子可以随机沿各个方向变化; 如果设置为 50, 子级粒子随机运动, 但与父级对象运动方向的偏移角度不超过 90 度。可动画, 百分数
<PCloud>.Spawn_Speed_Chaos	Float	0.0	设置子级粒子相对于父级粒子运动速度变化的百分比, 如果设置为 0, 表示运动速度不发生变化。可动画
<PCloud>.spawnSpeedType	Integer	0	设置速度混乱度类型: 0: Slow (慢) 1: Fast (快) 2: Both (二者均可)
<PCloud>.spawnInheritVelocity	Boolean	False	当设置为 On 时, 子级粒子继承父级粒子的运动速度, 并在该速度基础上进行随机的速度变化
<PCloud>.spawnSpeedFixed	Boolean	False	设置是否指定固定的速度变化量
<PCloud>.Spawn_Scale_Chaos	Float	0.0	设置子级粒子相对于父级粒子运动放缩变化量变化的百分比, 如果设置为 0, 表示运动放缩变化量不发生变化。可动画
<PCloud>.spawnScaleType	Integer	0	设置放缩变化类型: 0: Slow (依据因数的设置, 子级粒子相对于父级粒子运动速度随机变慢) 1: Fast (依据因数的设置, 子级粒子相对于父级粒子运动速度随机变快) 2: Both (依据因数的设置, 一些子级粒子相对于父级粒子运动速度随机变慢, 一些子级粒子相对于父级粒子运动速度随机变快)
<PCloud>.spawnScaleFixed	Boolean	False	当设置为 On 时, 指定一个固定的放缩变化量, 不采用 factor 参数指定的尺寸随机变化范围
<PCloud>.lifespanValueQueue	Array	#()	设置寿命值序列。在该项目中可以为繁殖创建的子级粒子指定新的寿命数值, 而不是继承其父级粒子的寿命数值
<PCloud>.Spawn_Lifespan	Integer	0	设置繁殖粒子的生存期初始值
<PCloud>.objectMutationQueue	Array	#()	设置实例对象类型粒子系统的实例对象数组

#### 8.4.10.5 Snow: GeometryClass (雪粒粒子系统)

构造函数

Snow ...

属性

属性名称	数据类型	默认值	说明
<Snow>.viewportcount	Integer	100	设置在场景视图中显示粒子的数量
<Snow>.rendercount	Integer	100	设置在渲染输出的一帧中包含的最大粒子数量
<Snow>.flakesize <Snow>.Flake_Size	Float	2.0	设置每个粒子渲染输出的尺寸。可动画
<Snow>.speed	Float	10.0	设置每个粒子离开发射器时的初始速度。粒子将依据该速度恒定不变地运动，除非其受到粒子空间扭曲的影响。可动画
<Snow>.variation	Float	2.0	设置粒子初始速度和运动方向的变化量。该数值设置得越大，粒子喷射得越强烈，粒子喷射分布的范围也越广。可动画
<Snow>.tumble	Float	0.0	设置雪粒粒子的随机翻滚量。取值范围：0 到 1 之间。如果指定为 0，雪粒粒子不翻滚；如果指定为 1，雪粒粒子强烈翻滚。每个粒子的旋转轴向是随机指定的。可动画
<Snow>.tumblescale <Snow>.Tumble_Rate	Float	1.0	设置雪粒粒子的随机旋转速度，该数值设置得越高，粒子翻滚旋转的速度越快。可动画
<Snow>.display	Integer	0	设置粒子在视图中显示的形状，与粒子最终被渲染输出的形状无关： 0: Flakes (水滴) 1: Dots (圆点) 2: Ticks (十字标记)
<Snow>.render	Integer	0	设置粒子渲染输出的方式： 0: Six Point (以六角形的方式渲染输出粒子，每个星角都是一个面，可以指定不同的材质) 1: Triangle (以三角形的方式渲染输出粒子，这种粒子只能指定一种材质) 2: Facing (以正四边形面片的方式渲染输出粒子，面片总是正面朝向视图，特别适合于指定材质贴图，例如可以为每个粒子指定雪花黑白图案的不透明贴图)
<Snow>.start <Snow>.starttime	Time	0f	设置粒子开始喷射的动画帧
<Snow>.lifetime <Snow>.life	Time	30f	设置粒子存在的寿命，即粒子持续多少帧后消亡
<Snow>.birthrate <Snow>.Birth_Rate	Time	0f	设置每帧产生新粒子的数量。如果属性.birth rate 小于或等于属性.viewportcount 的数值，粒子是连续不断发射的；如果属性.birth rate 大于属性.viewportcount 数值，粒子流是间歇发射的。可动画

(续表)

属性名称	数据类型	默认值	说明
<Snow>.constant	Boolean	True	当设置为 On 时, 属性.birth rate 参数失效, 再生速度使用属性.viewportcount 数值; 反之, 依据属性.birthrate 确定再生速率
<Snow>.emitterwidth <Snow>.width	Float	25.0	指定发射器的宽度。可动画
<Snow>.emitterheight <Snow>.length	Float	25.0	指定发射器长度。可动画
<Snow>.hideemitter	Boolean	False	设置是否在视图中隐藏发射器

#### 8.4.10.6 Spray: GeometryClass (喷射粒子系统)

##### 构造函数

Spray ...

##### 属性

属性名称	数据类型	默认值	说明
<Spray>.viewportcount	Integer	100	设置在指定动画帧的场景视图中显示粒子的数量。一般将粒子系统的视图显示数量设置得低一些, 以提高视图显示更新的速度, 粒子被最终渲染输出的数量由.rendercount 参数指定
<Spray>.rendercount	Integer	100	在渲染输出的一帧中包含的粒子数量, 当渲染数量达到指定数值时, 粒子产生的速度被延缓, 直到当前存在的粒子消亡时才产生新的粒子
<Spray>.dropsize <Spray>.Drop_Size	Float	2.0	设置每个粒子渲染输出时的尺寸。可动画
<Spray>.speed	Float	10.0	指定每个粒子离开发射器时的初始速度, 粒子将依据该速度恒定不变地运动, 除非其受到粒子空间扭曲的影响。可动画
<Spray>.variation	Float	2.0	设置粒子初始速度和运动方向的变化量, 该数值设置得越大, 粒子喷射得越强烈, 粒子喷射分布的范围也越广。可动画
<Spray>.display	Integer	0	设置粒子在视图中显示的形状, 与粒子最终被渲染输出的形状无关: 0: Flakes (水滴) 1: Dots (圆点) 2: Ticks (十字标记)
<Spray>.render	Integer	0	设置粒子的渲染方式: 0: Tetrahedron (以细长四面体的方式渲染输出粒子, 可用于模拟水滴的效果, 长度由属性.dropsize 决定) 1: Facing (以正四边形面片的方式渲染输出粒子, 面片总是正面朝向视图, 特别适合于模拟水滴的效果, 长度由属性.dropsize 决定)

(续表)

属性名称	数据类型	默认值	说明
<Spray>.start <Spray>.starttime	Time	0f	指定粒子开始喷射的动画帧
<Spray>.lifetime <Spray>.life	Time	30f	指定粒子存在的寿命, 即持续多少帧后消亡
<Spray>.birthrate <Spray>.Birth_Rate	Time	0f	指定每帧产生新粒子的数量, 可动画
<Spray>.constant	Boolean	True	当设置为 On 时, 属性.birth rate 失效, 再生速度使用属性.viewportcount 的数值; 否则, 依据属性.birthrate 参数确定再生速率
<Spray>.emitterheight <Spray>.length	Float	25.0	指定发射器长度。可动画
<Spray>.emitterwidth <Spray>.width	Float	25.0	指定发射器的宽度。可动画
<Spray>.hideemitter	Boolean	False	如果设置为 True 时, 在视图中隐藏发射器

#### 8.4.10.7 SuperSpray: GeometryClass (超级喷射粒子系统)

构造函数

superSpray ...

属性

属性名称	数据类型	默认值	说明
Basic Parameters (基本参数)			
<SuperSpray>.Off_Axis	Float	0.0	指定粒子流与发射器 Z 轴方向的偏移角度。可动画, Angle
<SuperSpray>.Axis_Spread	Float	0.0	指定粒子被发射后在 Z 轴方向的分散角度。可动画, Angle
<SuperSpray>.Off_Plane	Float	0.0	指定粒子流与发射器平面的偏移角度。可动画, Angle
<SuperSpray>.Plane_Spread	Float	0.0	指定粒子被发射后在发射器平面方向的分散角度。可动画, Angle
<SuperSpray>.iconsize <SuperSpray>.Emitter_Width	Float	20.0	设置发射器图标在视图中的显示尺寸
<SuperSpray>.iconHidden	Boolean	False	当设置为 On 时, 在视图中隐藏发射器的图标
<SuperSpray>.viewType	Integer	0	设置粒子图在视图中的显示类型: 0: Dots; 1: Ticks; 2: Mesh; 3: Bbox
<SuperSpray>.viewPercent <SuperSpray>.Percentage_Displayed	Float	10.0	设置出现在视图中的粒子百分比, 减少该数值可以加快视图更新显示的速度, 不会对最终的渲染输出产生影响。百分数

(续表)

属性名称	数据类型	默认值	说明
<b>Particle Generation (粒子生成)</b>			
<SuperSpray>.quantityMethod	Integer	0	设置粒子数量的方式: 0: Use rate (在动画的每一帧中发射相同数量的粒子) 1: Use total (指定在粒子系统寿命周期中发射粒子的总量)
<SuperSpray>.Birth_Rate	Integer	10	设置发射器每帧喷射的粒子数量。可动画
<SuperSpray>.Total_Number	Integer	100	设置发射器喷射的粒子总量
<SuperSpray>.speed	Float	10.0	指定在粒子寿命周期中，粒子在每一帧中的移动距离。可动画
<SuperSpray>.Speed_Variation	Float	0.0	设置粒子初始速度的变化量。可动画，百分数
<SuperSpray>.Emitter_Start	Time	0f	设置粒子开始出现的动画帧。可动画
<SuperSpray>.Emitter_Stop	Time	30f	设置粒子被发射完的动画帧
<SuperSpray>.Display_Until	Time	100f	设置在哪个动画帧粒子全部消亡
<SuperSpray>.life	Time	30f	设置每个粒子从被发射到消亡所持续的帧数。可动画
<SuperSpray>.Life_Variation	Time	0f	设置粒子寿命随机增加或减少的帧数。可动画
<SuperSpray>.subsampleCreationTime	Boolean	False	当设置为 On 时，通过为运动方程附加时间偏移的方式，避免粒子在一般帧处理时产生的粒子团堆积不流畅的效果
<SuperSpray>.subsampleEmitterTranslation	Boolean	True	当设置为 On 时，若当前基于对象的发射器被指定了空间平移动画，仅在整数帧时生成粒子，以避免粒子团堆积不流畅的效果
<SuperSpray>.subsampleEmitterRotation	Boolean	True	如果当前发射器被指定了旋转动画，若打开该选项，可以避免粒子团堆积不流畅的效果，并创建平滑螺旋发射粒子的效果
<SuperSpray>.size	Float	1.0	设置所有粒子的目标尺寸。可动画
<SuperSpray>.Size_Variation	Float	0.0	设置粒子尺寸变化的百分比。可动画，百分数
<SuperSpray>.Growth_Time	Time	10f	设置粒子从最小尺寸增长到目标尺寸所需要的帧数
<SuperSpray>.Fade_Time	Time	10f	设置粒子从目标尺寸萎缩到消亡所持续的帧数
<SuperSpray>.seed <SuperSpray>.Random_Seed	Integer	0	不同的种子数可以使它们产生随机的变化，避免一模一样的粒子形态
<b>Particle Type (粒子类型)</b>			
<SuperSpray>.particleType	Integer	0	设置粒子类型： 0: Standard particle (标准粒子) 1: Metaparticles (变形球粒子) 2: Object fragments (对象碎片) 3: Instanced Geometry (实例几何体)

(续表)

属性名称	数据类型	默认值	说明
<SuperSpray>.standardParticle	Integer	0	设置标准粒子类型: 0: Triangle (三角形) 1: Cube (立方体) 2: Special (特殊) 3: Facing (面) 4: Constant (恒定) 5: Tetra (四面体) 6: Sixpoint (六角星) 7: Sphere (球体)
<SuperSpray>.Metaparticle_Tension	Float	0.0	设置变形球粒子的张力。指定粒子的致密程度，粒子的张力越大，粒子本身越致密，也越容易与其他粒子融合在一起。可动画
<SuperSpray>.Metaparticle_Tension_Variation	Float	0.0	设置张力效果变化的百分比。可动画，百分数
<SuperSpray>.MetaballRenderCoarseness <SuperSpray>.Metaparticle_Coarseness	Float	0.5	设置变形球粒子计算的粗糙度，粗糙度数值越高，融合的计算量越小；如果粗糙度数值设置得过高，不产生粒子融合效果；如果粗糙度数值设置得过低，会耗费大量的计算时间
<SuperSpray>.metaballViewCoarseness	Float	1.0	设置变形球粒子在视图中显示的粗糙度
<SuperSpray>.metaballAutoCoarseness	Boolean	True	设置变形球粒子渲染输出是否计算粗糙度
<SuperSpray>.instancingObject	Node	undefined	设置实例对象
<SuperSpray>.instanceSubTree	Boolean	False	设置是否使用层级树。当设置为On时，如果当前选定对象是一个层级链接系统的一部份，所有层级链接系统中的其他对象都作为粒子对象；如果当前选定的对象是一个成组对象，整个群组中的子级对象都作为粒子的子对象
<SuperSpray>.instanceKeyOffsetType	Integer	0	设置动画偏移关键帧： 0: None (无) 1: Birth (出生) 2: Random (随机)
<SuperSpray>.instanceFrameOffset <SuperSpray>.Animation_Offset_Amount	Integer	0	设置相对于原始对象当前时间点动画进行帧偏移的数量
<SuperSpray>.mappingType	Integer	0	为粒子对象指定材质和贴图设置方式： 0: Time (时间); 1: Distance (距离)
<SuperSpray>.Mapping_Time_Base	Time	30f	当属性.mappingType=0时，设置粒子对象从诞生开始到完成一个贴图所持续的帧数。可动画

(续表)

属性名称	数据类型	默认值	说明
<SuperSpray>.Mapping_Distance_Base	Float	100.0	当属性.mappingType=1 时, 设置粒子对象从诞生开始到完成一个贴图所经过的距离。可动画
<b>Rotation and Collision (旋转和碰撞)</b>			
<SuperSpray>.Spin_Time	Time	30f	设置粒子对象旋转一周所持续的帧数, 如果设置为 0, 则粒子对象不旋转。可动画
<SuperSpray>.Spin_Time_Variation	Float	0.0	设置旋转速度变化的百分比。可动画, 百分数
<SuperSpray>.Spin_Phase	Float	0.0	设置粒子对象的初始旋转相位。可动画, Angle
<SuperSpray>.Spin_Phase_Variation	Float	0.0	设置初始相位变化的百分比。可动画, 百分数
<SuperSpray>.spinAxisType	Integer	0	设置旋转轴控制方式: 0: random (每个粒子的旋转轴向是随机指定的) 1: user defined (可以自定义一个旋转的轴向)
<SuperSpray>.Blur_Stretch	Integer	0	设置沿粒子运动的方向对粒子对象进行的挤出变形。最终挤出效果还受到粒子速度的影响。本属性仅在选择 Direction of Travel/Mblur 选项时才有用。可动画
<SuperSpray>.X_Spin_Vector	Float	1.0	设置 X 轴向方向的旋转角度。可动画
<SuperSpray>.Y_Spin_Vector	Float	0.0	设置 Y 轴向方向的旋转角度。可动画
<SuperSpray>.Z_Spin_Vector	Float	0.0	设置 Z 轴向方向的旋转角度。可动画
<SuperSpray>.Spin_Axis_Variation	Float	0.0	设置在每个旋转轴向上, 旋转角度的变化量。可动画, Angle
<SuperSpray>.Interparticle_Collisions_On	Integer	0	设置粒子运动过程中的相交碰撞计算是否有效: 0: Off; 1: On
<SuperSpray>.Interparticle_Collision_Steps	Integer	2	设置每帧计算间隔。较高的数值可以使碰撞计算更为精确, 碰撞效果更为真实, 但是会减慢计算的速度
<SuperSpray>.Interparticle_Collision_Bounce	Float	100.0	设置碰撞后粒子反弹的速度。可动画, 百分数
<SuperSpray>.Interparticle_Collision_Bounce_Variation	Float	0.0	设置粒子受发射器运动影响的百分比, 如果指定为 100, 则所有粒子对象都随同发射器一起运动; 如果指定为 0, 则所有粒子对象都不受发射器运动的影响。可动画, 百分数
<b>Object Motion Inheritance (对象运动继承)</b>			
<SuperSpray>.motionInfluence <SuperSpray>.Object_Motion_Inheritance	Float	100.0	设置在粒子产生时, 继承基于对象的发射器运动的粒子所占的百分比。可动画
<SuperSpray>.motionMultiplier <SuperSpray>.Object_Motion_Multiplier	Float	1.0	设置增加或减少基于对象的发射器运动对粒子运动的影响。可以为正数或负数。可动画

(续表)

属性名称	数据类型	默认值	说明
<SuperSpray>.motionVariation <SuperSpray>.Object_Motion_Multiplier_Variation	Float	0.0	设置属性.motionMultiplier 的变化量百分比。可动画, 百分数
<b>Bubble Motion (气泡运动)</b>			
<SuperSpray>.Bubble_Amplitude	Float	0.0	设置粒子在运动过程中, 相对于运动路径的偏移距离。可动画
<SuperSpray>.Bubble_Amplitude_Variation	Float	0.0	设置属性.Bubble_Amplitude 的变化量。单位为百分比, 适用于每一粒子。可动画, 百分数
<SuperSpray>.Bubble_Period	Time	100000f	设置粒子完成一次完整晃动(一个波峰和一个波谷)所持续的帧数。一般设置为20~30可以产生较好的气泡运动效果
<SuperSpray>.Bubble_Period_Variation	Float	0.0	设置每个粒子周期变化的百分比。可动画, Angle
<SuperSpray>.Bubble_Phase	Float	0.0	设置粒子对象开始晃动之前, 相对于运动路径的偏移距离。可动画, 百分数
<SuperSpray>.Bubble_Phase_Variation	Float	0.0	设置每个粒子相位变化的百分比。可动画, 百分数
<b>Particle Spawn (粒子繁殖)</b>			
<SuperSpray>.spawnType	Integer	0	设置当粒子碰撞到导向面后所发生的情况: 0: None (无) 1: Die after collision (碰撞后消亡) 2: Spawn on collision (碰撞后繁殖) 3: Spawn on death (碰撞后消亡) 4: Spawn trails (繁殖拖尾)
<SuperSpray>.Die_X_frames_after_collision	Time	0f	设置粒子碰撞到导向面后持续存在的时间。如果设置为0, 当粒子碰撞到导向面之后立即消失。可动画
<SuperSpray>.Die_X_frames_after_collision_variation	Float	0.0	设置每个粒子持续时间变化的百分比。可动画, 百分数
<SuperSpray>.Spawn_Affects	Integer	100	设置在粒子系统中繁殖粒子的百分比
<SuperSpray>.Spawn_Multiplier_Variation	Float	0.0	设置倍增数值在每一帧中变化的百分比。可动画, 百分数
<SuperSpray>.Spawn_Direction_Chaos	Float	0.0	设置子级粒子运动方向沿父级粒子运动方向变化的混乱度。如果设置为0, 不发生方向变化; 如果设置为100, 子级粒子可以随机沿各个方向变化; 如果设置为50, 子级粒子随机运动, 但与父级对象运动方向的偏移角度不超过90度。可动画, 百分数
<SuperSpray>.Spawn_Speed_Chaos	Float	0.0	设置子级粒子相对于父级粒子运动速度随机变化的百分比。可动画

(续表)

属性名称	数据类型	默认值	说明
<SuperSpray>.spawnSpeedType	Integer	0	设置速度混乱度类型: 0: Slow (慢) 1: Fast (快) 2: Both (二者均可)
<SuperSpray>.spawnInheritVelocity	Boolean	False	当设置为 On 时, 指定子级粒子继承父级粒子的运动速度, 并在该速度基础上进行随机的速度变化
<SuperSpray>.spawnSpeedFixed	Boolean	False	设置是否指定固定的速度变化量
<SuperSpray>.Spawn_Scale_Chaos	Float	0.0	设置子级粒子相对于父级粒子运动放缩变化量变化的百分比。如果设置为 0, 表示运动放缩变化量不发生变化。可动画
<SuperSpray>.spawnScaleType	Integer	0	设置放缩变化类型: 0: Slow (依据因数的设置, 子级粒子相对于父级粒子运动速度随机变慢) 1: Fast (依据因数的设置, 子级粒子相对于父级粒子运动速度随机变快) 2: Both (依据因数的设置, 一些子级粒子相对于父级粒子运动速度随机变慢, 一些子级粒子相对于父级粒子运动速度随机变快)
<SuperSpray>.spawnScaleFixed	Boolean	False	当设置为 On 时, 指定一个固定的放缩变化量, 不采用 factor 参数指定的尺寸随机变化范围
<SuperSpray>.lifespanValueQueue	Array	#()	设置寿命值序列。在该项中可以为繁殖创建的子级粒子指定新的寿命数值, 而不是继承其父级粒子的寿命数值
<SuperSpray>.objectMutationQueue	Array	#()	设置实例对象类型粒子系统的实例对象数组

#### 8.4.11 Geometry-NURBS Objects (NURBS 对象)

下面列出了所有 NURBS 类对象:

- ◆ NURBSSurf (CV 曲面)
- ◆ Point\_Surf (点曲面)

##### 8.4.11.1 NURBSSurf: GeometryClass (CV 曲面)

NURBSSurfs 是由 CV (控制点) 定义的 NURBS 表面对象。在 3ds max 用户界面里创建的 NURBSSurf 对象的类都为 NURBSSurf。

构造函数

NURBSNode <nurbs\_set> ...

属性

NURBSSurf 对象的属性是通过 NURBSSet 描述对象所存取的。

#### 8.4.11.2 Point\_Surf: GeometryClass (点曲面)

Point\_Surf 是由插入点 (Interpolated Point) 定义的 NURBS 表面对象。

构造函数

NURBSNode <nurbs\_set> ...

属性

属性名称	数据类型	默认值	说明
<Point_Surf>.angle	Float	0.0	设置交叉剖面图形在渲染中旋转的角度。可动画
<Point_Surf>.thickness	Float	1.0	设置可渲染 NURBS 点曲线剖面的精度。可动画
<Point_Surf>.sides	Float	12.0	设置可渲染 NURBS 点曲线剖面的边数。可动画

## 8.5 Shape: Node

下面列出了 3ds max 里所有 Shape 类对象：

- ◆ Spline (样条曲线)
  - Arc (弧形样条曲线)
  - Circle (圆形样条曲线)
  - Donut (圆环样条曲线)
  - Ellipse (椭圆样条曲线)
  - Helix (螺旋线样条曲线)
  - Line (线形样条曲线)
  - NGon (多边形样条曲线)
  - Rectangle (矩形样条曲线)
  - Section (截面样条曲线)
  - Star (星形样条曲线)
  - Text (文本样条曲线)
- ◆ NURBS 曲线
  - CV\_Curve (CV 曲线)
  - Point\_Curve (点曲线)

### 8.5.1 Shape 类方法

通用方法

numKnots <shape> [ <spline\_index\_Integer> ]

返回一个整数，表示指定形对象<shape>里指定序号 spline 里的结点 (knot, 也叫顶点或控制点) 数量。如果参数<spline\_index\_Integer>没有指定，返回整个<shape>对象上的顶点数。

例如：

```
y = donut()
numKnots y
```

### 插入方法

3ds max 里有好几个插入函数用于 Shape 类场景对象。它们提供了操作 Shape 类对象的一些功能，如获取曲线上某点的坐标、获取曲线的长度或根据给定的坐标查找曲线上最靠近的顶点等。这些函数还可用来将一些克隆对象沿曲线排列，或当曲线运动时，使一个对象集保持沿曲线排列。

插入函数的基本格式如下：

```
Func_Name_String <shape> [<curve_num>] \
    <parameter> [steps:<Integer>]
```

其中：

- ◆ Func\_Name\_String 函数名；
- ◆ <shape> 一个 Shape 类场景对象，如 Line；
- ◆ <curve\_num> 可选参数，如果指定<shape>对象包含几条曲线，<curve\_num>用来指定第几条，默认值为 1；
- ◆ <parameter> 可选参数，浮点数，取值范围 0.0~1.0，表示沿曲线长度方向指定比例例外的点；

参数<parameter>有两类计算方法：

第一类与 3ds max 的 Path 控制器相匹配，沿路径上每一 segment 表示的百分比是不同的，而与它所处的位置有关。如果一个样条曲线 spline 由 4 段 segment 组成，第一个 segment 的<parameter>取值为[0 0.25]，第二个 segment 的<parameter>取值为[0.25 0.50]，第三个 segment 的<parameter>取值为[0.50 0.75]，第四个 segment 的<parameter>取值为[0.75 1.0]，而不管每一 segment 的长度。

第二类沿曲线上每一线段表示的百分比是一样的。0.0 表示曲线起点，0.5 表示曲线中点，而 1.0 表示曲线终点。

- ◆ steps:<Integer> 可选关键字参数，用于指定插入步长，默认值为曲线长度的 5 倍。

例如：如果一个曲线长度为 100 个 3ds max 单位长度，那么默认步长为 500。

下面函数中所有坐标均在当前坐标系下。

1. pathInterp <shape> [ <curve\_num> ] <parameter>

返回一个 Point3 值，表示指定图形<shape>的指定序号曲线（默认值为 1）上与参数<parameter>对应的插入点坐标，<parameter>值按上述第一类计算方法。

2. lengthInterp <shape> [ <curve\_num> ] <parameter> [ steps:<Integer> ]

返回一个 Point3 值，表示指定图形<shape>的指定序号曲线（默认值为 1）上与参数<parameter>对应的插入点坐标，<parameter>值按上述第二类计算方法。

3. resetLengthInterp()

清除插入长度缓存。如果正在进行长度插入运算的曲线在两次插入操作之间可能有其他的编辑要进行时，要调用本函数。

#### 4. curveLength <shape> [ <curve\_num> ]

返回指定图形<shape>的指定序号曲线的弧长。这个长度不包含任何针对图形的转换级缩放。

#### 5. pathTangent <shape> [ <curve\_num> ] <parameter>

返回一个矢量，表示指定路径<shape>的指定序号曲线（默认值为1）上与参数<parameter>对应的插入点的切线方向。

#### 6. lengthTangent <shape> [ <curve\_num> ] <parameter> [ steps:<Integer> ]

返回一个矢量，表示指定路径<shape>的指定序号曲线（默认值为1）上与参数<parameter>对应的弧长插入点的切线方向。

#### 7. nearestPathParam <shape> [ <curve\_num> ] <point3> [ steps:<Integer> ]

返回指定路径<shape>的指定序号曲线上与指定点<point3>距离最近的插入点的插入参数<parameter>，<parameter>的计算按第一种方法。之后可以用函数pathInterp()来获取该点的坐标值。

#### 8. pathToLengthParam <shape> [ <curve\_num> ] <parameter> [ steps:<Integer> ]

将指定插入点参数<parameter>从第一种计算方法转换为按第二种计算方法。

#### 9. lengthToPathParam <shape> [ <curve\_num> ] <parameter> [ steps:<Integer> ]

将指定插入点参数<parameter>从第二种计算方法转换为按第一种计算方法。

### 8.5.2 Shape-Spline (样条曲线)

下面列出了所有3ds max里Spline类Shape对象：

- ◆ Arc (弧形样条曲线)
- ◆ Circle (圆形样条曲线)
- ◆ Donut (圆环样条曲线)
- ◆ Ellipse (椭圆样条曲线)
- ◆ Helix (螺旋线样条曲线)
- ◆ Line (线形样条曲线)
- ◆ NGon (多边形样条曲线)
- ◆ Rectangle (矩形样条曲线)
- ◆ Section (截面样条曲线)
- ◆ Star (星形样条曲线)
- ◆ Text (文本样条曲线)

### 8.5.3 Spline类Shape对象通用属性和方法

下面属性适用于所有Spline类Shape对象。

## 属性

属性名称	数据类型	默认值	说明
<spline>.renderable	Boolean	False	当设置为 On 时，样条曲线曲线对象可以被渲染
<spline>.mapCoords	Boolean	False	当设置为 On 时，为可渲染的样条曲线曲线对象自动指定贴图坐标，默认的贴图坐标指定沿周长方向为贴图 U 轴；沿路径方向为贴图 V 轴
<spline>.thickness	Float	1.0	设置样条曲线曲线直径

**注意** Spline 类属性.renderable 与 Node 级属性.renderable 名字冲突。

例如：

```
$ .renderable=False           --表示关掉 Node 级属性.renderable
$ .baseobject.renderable=False --表示 Spline 对象的属性.renderable
```

## 方法

### 1. applyOffset <spline\_shape\_Node> <offset\_Float>

将指定 Spline 类 Shape 对象复制一份，并向右偏移指定距离。如果指定的<spline\_shape\_Node>为非封闭曲线，偏移结果和偏移轮廓会组成一个封闭曲线。如果<offset\_Float>为负数，表示向左偏移指定距离；如果指定 Spline 类图形< spline\_shape\_Node>不是 SplineShape（或 Editable\_Spline），系统会先将它转化为 SplineShape。

### 2. measureOffset <spline\_shape\_Node> <point3>

返回一个 Float 值，其绝对值表示从指定点<point3>到图形<spline\_shape\_Node>的最近距离，其符号规则为：如果点在图形对象的左边，返回值为正数。该返回值可以用于函数applyOffset()的参数<offset\_Float>，来生成一个通过该点的偏移曲线。但是如果最近点为一个非封闭图形的一个端点，用上面方法生成的曲线将不会通过点<point3>。

### 3. trimExtend <fixed\_Node\_array> <alterable\_Node\_array> \ [trim: <boolean> ] [extend: <boolean>] \ [infinite: <boolean>] [project: #view | #3D | #grid]

裁剪或伸展指定对象集合。

### 8.5.3.1 Arc: Shape (弧形样条曲线)

#### 构造函数

arc ...

#### 属性

属性名称	数据类型	默认值	说明
<Arc>.radius	Float	25.0	设置弧形的半径。可动画
<Arc>.from	Float	45.0	设置弧形的起始角度。可动画
<Arc>.to	Float	135.0	设置弧形的终止角度。可动画
<Arc>.pie	Boolean	False	当设置为 On 时，生成封闭的扇形

(续表)

属性名称	数据类型	默认值	说明
<Arc>.reverse	Boolean	False	当设置为 On 时, 反转弧形
<Arc>.steps	Integer	6	设置每两个顶点之间短直线的数量
<Arc>.optimize	Boolean	True	当设置为 On 时, 自动检查并移除样条曲线上的多余步数设置, 以减小样条曲线的复杂程度
<Arc>.adaptive	Boolean	False	当设置为 On 时, 将依据样条曲线的复杂程度自动指定步数, 对于直线自动将步数设定为 0; 对于复杂的弯曲曲线, 自动加大步数使其光滑
<Arc>.angle	Float	0.0	设置剖面图形沿路径轴向旋转的角度。可动画
<Arc>.thickness	Float	1.0	设置可渲染二维图形剖面的直径。可动画
<Arc>.sides	Float	12.0	设置可渲染二维图形剖面的边数。可动画
<Arc>.viewport_thickness	Float	1.0	设置二维图形在视窗中的剖面直径
<Arc>.viewport_sides	Integer	12	设置二维图形在视窗中的剖面边数
<Arc>.viewport_angle	Float	0.0	设置剖面图形在视窗中的沿路径轴向旋转的角度。可动画
<Arc>.DisplayRenderMesh	Boolean	False	当设置为 On 时, 将可渲染的二维图形对象显示为网格对象
<Arc>.UseViewportSettings	Boolean	False	当设置为 On 时, 依据视图设置将可渲染的二维图形对象显示为网格对象
<Arc>.DisplayRenderSettings	Boolean	True	当设置为 On 时, 依据渲染设置将可渲染的二维图形对象显示为网格对象
<Arc>.renderable	Boolean	True	当设置为 On 时, 二维图形对象可以被渲染
<Arc>.mapCoords	Boolean	False	当设置为 On 时, 为可渲染的二维图形对象自动指定贴图坐标, 默认的贴图坐标指定沿周长方向为贴图 U 轴; 沿路径方向为贴图 V 轴

### 8.5.3.2 Circle: Shape (圆形样条曲线)

构造函数

circle ...

属性

属性名称	数据类型	默认值	说明
<Circle>.radius	Float	25.0	设置圆形的半径。可动画
<Circle>.steps	Integer	6	设置每两个顶点之间短直线的数量
<Circle>.optimize	Boolean	True	当设置为 On 时, 自动检查并移除样条曲线上的多余步数设置, 以减小样条曲线的复杂程度
<Circle>.adaptive	Boolean	False	当设置为 On 时, 将依据样条曲线的复杂程度自动指定步数, 对于直线自动将步数设定为 0; 对于复杂的弯曲曲线, 自动加大步数使其光滑
<Circle>.angle	Float	0.0	设置剖面图形沿路径轴向旋转的角度。可动画

(续表)

属性名称	数据类型	默认值	说明
<Circle>.thickness	Float	1.0	设置可渲染二维图形剖面的直径。可动画
<Circle>.sides	Float	12.0	设置可渲染二维图形剖面的边数。可动画
<Circle>.viewport_thickness	Float	1.0	设置二维图形在视窗中的剖面直径
<Circle>.viewport_sides	Integer	12	设置二维图形在视窗中的剖面边数
<Circle>.viewport_angle	Float	0.0	设置剖面图形在视窗中的沿路径轴向旋转的角度
<Circle>.DisplayRenderMesh	Boolean	False	当设置为 On 时, 将可渲染的二维图形对象显示为网格对象
<Circle>.UseViewportSettings	Boolean	False	当设置为 On 时, 依据视图设置将可渲染的二维图形对象显示为网格对象
<Circle>.DisplayRenderSettings	Boolean	True	当设置为 On 时, 依据渲染设置将可渲染的二维图形对象显示为网格对象
<Circle>.renderable	Boolean	True	当设置为 On 时, 二维图形对象可以被渲染
<Circle>.mapCoords	Boolean	False	当设置为 On 时, 为可渲染的二维图形对象自动指定贴图坐标, 默认的贴图坐标指定沿周长方向为贴图 U 轴; 沿路径方向为贴图 V 轴

### 8.5.3.3 Donut: Shape (圆环样条曲线)

#### 构造函数

```
donut ...
```

#### 属性

属性名称	数据类型	默认值	说明
<Donut>.radius1	Float	35.0	设置第一个圆环的半径。可动画
<Donut>.radius2	Float	25.0	设置第二个圆环的半径。可动画
<Donut>.steps	Integer	6	设置每两个顶点之间短直线的数量
<Donut>.optimize	Boolean	True	当设置为 On 时, 自动检查并移除样条曲线上的多余步数设置, 以减小样条曲线的复杂程度
<Donut>.adaptive	Boolean	False	当设置为 On 时, 将依据样条曲线的复杂程度自动指定步数, 对于直线自动将步数设定为 0; 对于复杂的弯曲曲线, 自动加大步数使其光滑
<Donut>.angle	Float	0.0	设置剖面图形沿路径轴向旋转的角度。可动画
<Donut>.thickness	Float	1.0	设置可渲染二维图形剖面的直径。可动画
<Donut>.sides	Float	12.0	设置可渲染二维图形剖面的边数。可动画
<Donut>.viewport_thickness	Float	1.0	设置二维图形在视窗中的剖面直径。可动画
<Donut>.viewport_sides	Integer	12	设置二维图形在视窗中的剖面边数

(续表)

属性名称	数据类型	默认值	说明
<Donut>.viewport_angle	Float	0.0	设置剖面图形在视窗中的沿路径轴向旋转的角度。可动画
<Donut>.DisplayRenderMesh	Boolean	False	当设置为On时,将可渲染的二维图形对象显示为网格对象
<Donut>.UseViewportSettings	Boolean	False	当设置为On时,依据视图设置将可渲染的二维图形对象显示为网格对象
<Donut>.DisplayRenderSettings	Boolean	True	当设置为On时,依据渲染设置将可渲染的二维图形对象显示为网格对象
<Donut>.renderable	Boolean	True	当设置为On时,二维图形对象可以被渲染
<Donut>.mapCoords	Boolean	False	当设置为On时,为可渲染的二维图形对象自动指定贴图坐标,默认的贴图坐标指定沿周长方向为贴图U轴;沿路径方向为贴图V轴

#### 8.5.3.4 Ellipse: Shape (椭圆样条曲线)

构造函数

ellipse ...

属性

属性名称	数据类型	默认值	说明
<Ellipse>.length	Float	25.0	指定椭圆在局部坐标系Y轴方向的长轴长度。可动画
<Ellipse>.width	Float	35.0	指定椭圆在局部坐标系X轴方向的短轴宽度。可动画
<Ellipse>.steps	Integer	6	设置每两个顶点之间短直线的数量
<Ellipse>.optimize	Boolean	True	当设置为On时,自动检查并移除样条曲线上的多余步数设置,以减小样条曲线的复杂程度
<Ellipse>.adaptive	Boolean	False	当设置为On时,将依据样条曲线的复杂程度自动指定步数,对于直线自动将步数设定为0;对于复杂的弯曲曲线,自动加大步数使其光滑
<Ellipse>.angle	Float	0.0	设置剖面图形沿路径轴向旋转的角度。可动画
<Ellipse>.thickness	Float	1.0	设置可渲染二维图形剖面的直径。可动画
<Ellipse>.sides	Float	12.0	设置可渲染二维图形剖面的边数。可动画
<Ellipse>.viewport_thickness	Float	1.0	设置二维图形在视窗中的剖面直径
<Ellipse>.viewport_sides	Integer	12	设置二维图形在视窗中的剖面边数
<Ellipse>.viewport_angle	Float	0.0	设置剖面图形在视窗中的沿路径轴向旋转的角度
<Ellipse>.DisplayRenderMesh	Boolean	False	当设置为On时,将可渲染的二维图形对象显示为网格对象
<Ellipse>.UseViewportSettings	Boolean	False	当设置为On时,依据视图设置将可渲染的二维图形对象显示为网格对象

(续表)

属性名称	数据类型	默认值	说明
<Ellipse>.DisplayRenderSettings	Boolean	True	当设置为 On 时，依据视图设置将可渲染的二维图形对象显示为网格对象
<Ellipse>.renderable	Boolean	True	当设置为 On 时，二维图形对象可以被渲染
<Ellipse>.mapCoords	Boolean	False	当设置为 On 时，为可渲染的二维图形对象自动指定贴图坐标，默认的贴图坐标指定沿周长方向为贴图 U 轴；沿路径方向为贴图 V 轴

### 8.5.3.5 Helix: Shape (螺旋线样条曲线)

#### 构造函数

helix ...

#### 属性

属性名称	数据类型	默认值	说明
<Helix>.radius1	Float	35.0	指定螺旋线开始圆环的半径。可动画
<Helix>.radius2	Float	15.0	指定螺旋线结束圆环的半径。可动画
<Helix>.height	Float	25.0	指定螺旋线的高度。可动画
<Helix>.turns	Float	2.0	指定螺旋线在开始圆环与结束圆环之间旋转的圈数。可动画
<Helix>.bias	Float	0.0	指定螺旋的偏向。如果设置为-1.0 螺旋偏向开始端，即开始端螺旋比较密集；如果设置为 1.0 融合偏向结束端，即结束端螺旋比较密集，可动画
<Helix>.direction	Integer	0	指定螺旋的旋转方向： 0: Clockwise (顺时针) 1: Counterclockwise (逆时针)
<Helix>.angle	Float	0.0	设置剖面图形沿路径轴向旋转的角度。可动画
<Helix>.thickness	Float	1.0	设置可渲染二维图形剖面的直径。可动画
<Helix>.sides	Float	12.0	设置可渲染二维图形剖面的边数。可动画
<Helix>.viewport_thickness	Float	1.0	设置二维图形在视窗中的剖面直径
<Helix>.viewport_sides	Integer	12	设置二维图形在视窗中的剖面边数
<Helix>.viewport_angle	Float	0.0	设置剖面图形在视窗中的沿路径轴向旋转的角度
<Helix>.DisplayRenderMesh	Boolean	False	当设置为 On 时，将可渲染的二维图形对象显示为网格对象
<Helix>.UseViewportSettings	Boolean	False	当设置为 On 时，依据视图设置将可渲染的二维图形对象显示为网格对象
<Helix>.DisplayRenderSettings	Boolean	True	当设置为 On 时，依据渲染设置将可渲染的二维图形对象显示为网格对象
<Helix>.renderable	Boolean	True	当设置为 On 时，二维图形对象可以被渲染
<Helix>.mapCoords	Boolean	False	当设置为 On 时，为可渲染的二维图形对象自动指定贴图坐标，默认的贴图坐标指定沿周长方向为贴图 U 轴；沿路径方向为贴图 V 轴

### 8.5.3.6 Line: Shape (线形样条曲线)

构造函数

line ...

属性

属性名称	数据类型	默认值	说明
<Line>.steps	Integer	6	设置每两个顶点之间短直线的数量
<Line>.optimize	Boolean	True	当设置为 On 时，自动检查并移除样条曲线上的多余步数设置，以减小样条曲线的复杂程度
<Line>.adaptive	Boolean	False	当设置为 On 时，将依据样条曲线的复杂程度自动指定步数，对于直线自动将步数设定为 0；对于复杂的弯曲曲线，自动加大步数使其光滑
<Line>.angle	Float	0.0	设置剖面图形沿路径轴向旋转的角度。可动画
<Line>.thickness	Float	1.0	设置可渲染二维图形剖面的直径。可动画
<Line>.sides	Float	12.0	设置可渲染二维图形剖面的边数。可动画
<Line>.viewport_thickness	Float	1.0	设置二维图形在视窗中的剖面直径
<Line>.viewport_sides	Integer	12	设置二维图形在视窗中的剖面边数
<Line>.viewport_angle	Float	1.0	设置剖面图形在视窗中的沿路径轴向旋转的角度
<Line>.DisplayRenderMesh	Boolean	False	当设置为 On 时，将可渲染的二维图形对象显示为网格对象
<Line>.UseViewportSettings	Boolean	False	当设置为 On 时，依据视图设置将可渲染的二维图形对象显示为网格对象
<Line>.DisplayRenderSettings	Boolean	True	当设置为 On 时，依据渲染设置将可渲染的二维图形对象显示为网格对象
<Line>.renderable	Boolean	True	当设置为 On 时，二维图形对象可以被渲染
<Line>.mapCoords	Boolean	False	当设置为 On 时，为可渲染的二维图形对象自动指定贴图坐标，默认的贴图坐标指定沿周长方向为贴图 U 轴；沿路径方向为贴图 V 轴

**注意** Line 类 Shape 与 SplineShape 类 Shape 是一样的，所有与 SplineShape 相关的方法同样适用于 Line，这些方法请读者参见 SplineShape: Shape。

### 8.5.3.7 NGon: Shape (多边形样条曲线)

构造函数

ngon ...

属性

属性名称	数据类型	默认值	说明
<NGon>.radius	Float	25.0	设置基准圆形的半径长度。可动画
<NGon>.scribe	Integer	0	设置多边形的创建形式: 0: Circumscribed (创建基准圆形的外切多边形) 1: Inscribed (创建基准圆形的内切多边形)
<NGon>.sides	Integer	6	指定多边形的顶点数量。可动画
<NGon>.corner_Radius	Float	0.0	指定多边形顶角的倒圆半径。如果设定为 0，则生成不倒圆的尖角多边形。可动画
<NGon>.circular	Boolean	True	当设置为 On 时，指定多边形为圆形。可动画
<NGon>.steps	Integer	6	设置每两个顶点之间短直线的数量
<NGon>.optimize	Boolean	True	当设置为 On 时，自动检查并移除样条曲线上的多余步数设置，以减小样条曲线的复杂程度
<NGon>.adaptive	Boolean	False	当设置为 On 时，将依据样条曲线的复杂程度自动指定步数，对于直线自动将步数设定为 0；对于复杂的弯曲曲线，自动加大步数使其光滑
<NGon>.angle	Float	0.0	设置剖面图形沿路径轴向旋转的角度。可动画
<NGon>.thickness	Float	1.0	设置可渲染二维图形剖面的直径。可动画
<NGon>.sides	Float	12.0	设置可渲染二维图形剖面的边数。可动画
<NGon>.viewport_thickness	Float	1.0	设置二维图形在视窗中的剖面直径
<NGon>.viewport_sides	Integer	12	设置二维图形在视窗中的剖面边数
<NGon>.viewport_angle	Float	1.0	设置剖面图形在视窗中的沿路径轴向旋转的角度
<NGon>.DisplayRenderMesh	Boolean	False	当设置为 On 时，将可渲染的二维图形对象显示为网格对象
<NGon>.UseViewportSettings	Boolean	False	当设置为 On 时，依据视图设置将可渲染的二维图形对象显示为网格对象
<NGon>.DisplayRenderSettings	Boolean	True	当设置为 On 时，依据渲染设置将可渲染的二维图形对象显示为网格对象
<NGon>.mapCoords	Boolean	False	当设置为 On 时，为可渲染的二维图形对象自动指定贴图坐标，默认的贴图坐标指定沿周长方向为贴图 U 轴；沿路径方向为贴图 V 轴
<NGon>.renderable	Boolean	True	当设置为 On 时，二维图形对象可以被渲染

### 8.5.3.8 Rectangle: Shape (矩形样条曲线)

构造函数

rectangle ...

属性

属性名称	数据类型	默认值	说明
<Rectangle>.length	Float	10.0	设置矩形沿Y轴方向的长度。可动画
<Rectangle>.width	Float	25.0	设置矩形沿X轴方向的宽度。可动画
<Rectangle>.Corner_Radius	Float	0.0	指定矩形的倒圆半径。如果设定为0，则矩形为90度的直角。可动画
<Rectangle>.steps	Integer	6	设置每两个顶点之间短直线的数量
<Rectangle>.optimize	Boolean	True	当设置为On时，自动检查并移除样条曲线上的多余步数设置，以减小样条曲线的复杂程度
<Rectangle>.adaptive	Boolean	False	当设置为On时，将依据样条曲线的复杂程度自动指定步数，对于直线自动将步数设定为0；对于复杂的弯曲曲线，自动加大步数使其光滑
<Rectangle>.angle	Float	0.0	设置剖面图形沿路径轴向旋转的角度。可动画
<Rectangle>.thickness	Float	1.0	设置可渲染二维图形剖面的直径。可动画
<Rectangle>.sides	Float	12.0	设置可渲染二维图形剖面的边数。可动画
<Rectangle>.viewport_thickness	Float	1.0	设置二维图形在视窗中的剖面直径
<Rectangle>.viewport_sides	Integer	12	设置二维图形在视窗中的剖面边数
<Rectangle>.viewport_angle	Float	0.0	设置剖面图形在视窗中沿路径轴向旋转的角度
<Rectangle>.DisplayRenderMesh	Boolean	False	当设置为On时，将可渲染的二维图形对象显示为网格对象
<Rectangle>.UseViewportSettings	Boolean	False	当设置为On时，依据视图设置将可渲染的二维图形对象显示为网格对象
<Rectangle>.DisplayRenderSettings	Boolean	True	当设置为On时，依据渲染设置将可渲染的二维图形对象显示为网格对象
<Rectangle>.renderable	Boolean	True	当设置为On时，二维图形对象可以被渲染
<Rectangle>.mapCoords	Boolean	False	当设置为On时，为可渲染的二维图形对象自动指定贴图坐标，默认的贴图坐标指定沿周长方向为贴图U轴；沿路径方向为贴图V轴

### 8.5.3.9 Section: Shape (截面样条曲线)

构造函数

section...

属性

属性名称	数据类型	默认值	说明
<Section>.width	Float	0.0	指定截面矩形的宽度。可动画
<Section>.length	Float	0.0	指定截面矩形的长度。可动画
<section>.angle	Float	0.0	当截面对象可以被渲染时，设置其沿路径轴向旋转的角度
<section>.renderable	Boolean	False	当设置为 On 时，二维图形对象产生一个面，并可以被渲染
<section>.mapCoords	Boolean	False	设置可渲染的二维图形对象是否指定贴图坐标
<section>.thickness	Float	1.0	设置可渲染的二维图形的线段数
<section>.sides	Integer	12	设置可渲染的二维图形对象的边数
<section>.viewport_angle	Float	0.0	设置可渲染的二维图形对象在视窗中沿路径轴向旋转的角度
<section>.DisplayRenderMesh	Boolean	False	当设置为 On 时，将可渲染的二维图形对象显示为网格对象
<section>.UseViewportSettings	Boolean	False	当设置为 On 时，二维图形对象可以被渲染
<section>.DisplayRenderSettings	Boolean	True	当设置为 On 时，依据视图设置将可渲染的二维图形对象显示为网格对象

**注意** 为了正确存取属性.renderable，必须使用<section>.baseobject.renderable 替代，因为属性.renderable 同时也是所有 Node 对象的一个属性。

由于 3ds max 内部的错误，如果在循环里创建一个 Section 对象，然后将其转换成 SplineShape，Section 对象不能正确地塌陷成 SplineShape 类对象。为了正确地进行转换，必须在 Section 对象创建后执行一个视窗刷新命令。请参见下面的例子：

```

meshSelected = sphere()      -- 创建要生成等高线的对象
minZ = meshSelected.min.z   -- 获取 z 坐标的最大、最小值
maxZ = meshSelected.max.z
numLevels = 10               -- 等高线数量
delta =(maxZ-minZ)/(numLevels + 1) -- 计算步长
for currentZ = minZ to maxZ by delta do -- 开始循环
( s = section pos:[0, 0, currentZ]      -- 创建 Section 对象
    max views redraw                  -- 本行是为避免 bug 而加的视窗刷新命令
    convertToSplineShape s            -- 将 Section 转换成 SplineShape
    s.renderable = True              -- 设置.renderable 属性
)

```

### 8.5.3.10 Star: Shape (星形样条曲线)

构造函数

star ...

属性

属性名称	数据类型	默认值	说明
<Star>.radius1 <Star>.Radius_1	Float	25.0	指定星形的内圆环半径
<Star>.radius2 <Star>.Radius_2	Float	12.0	指定星形的外圆环半径
<Star>.points <Star>.numPoints	Integer	6	指定星形的顶点数目
<Star>.distort <Star>.distortion	Float	0.0	围绕中心旋转外圆环上的顶点，产生类似于锯齿的形态
<Star>.fillet1	Float	0.0	设置星形内圆环上的倒圆半径
<Star>.fillet2	Float	0.0	设置星形外圆环上的倒圆半径
<Star>.steps	Integer	6	设置每两个顶点之间短直线的数量
<Star>.optimize	Boolean	True	当设置为On时，自动检查并移除样条曲线上的多余步数设置，以减小样条曲线的复杂程度
<Star>.adaptive	Boolean	False	当设置为On时，将依据样条曲线的复杂程度自动指定步数，对于直线自动将步数设定为0；对于复杂的弯曲曲线，自动加大步数使其光滑
<Star>.angle	Float	0.0	设置剖面图形沿路径轴向旋转的角度。可动画
<Star>.thickness	Float	1.0	设置可渲染二维图形剖面的直径。可动画
<Star>.sides	Float	12.0	设置可渲染二维图形剖面的边数。可动画
<Star>.viewport_thickness	Float	1.0	设置二维图形在视窗中的剖面直径
<Star>.viewport_sides	Integer	12	设置二维图形在视窗中的剖面边数
<Star>.viewport_angle	Float	0.0	设置剖面图形在视窗中的沿路径轴向旋转的角度
<Star>.DisplayRenderMesh	Boolean	False	当设置为On时，将可渲染的二维图形对象显示为网格对象
<Star>.UseViewportSettings	Boolean	False	当设置为On时，二维图形对象可以被渲染
<Star>.DisplayRenderSettings	Boolean	True	当设置为On时，依据视图设置将可渲染的二维图形对象显示为网格对象
<Star>.renderable	Boolean	True	当设置为On时，二维图形对象可以被渲染
<Star>.mapCoords	Boolean	False	当设置为On时，为可渲染的二维图形对象自动指定贴图坐标，默认的贴图坐标指定沿周长方向为贴图U轴；沿路径方向为贴图V轴

**注意** 为了正确存取.points属性，必须通过.baseobject属性，因为属性.points同时也是所有Node对象的一个属性。如：

```
MyStar.baseobject.points=10
```

### 8.5.3.11 Text: Shape(文本样条曲线)

构造函数

text ...

## 属性

属性名称	数据类型	默认值	说明
<text>.font	String	“Arial”	从字体列表中选择 3ds max 可用的字体
<text>.italic	Boolean	False	开/关字体的斜体形式
<text>.underline	Boolean	False	开/关字体的下划线形式
<text>.alignment	Integer	1	设置字体的对齐方式： 1: Align Left (左对齐) 2: Align Center (中对齐) 3: Align Right (右对齐) 4: Justify (用空行将绑定框填满)
<text>.size	Float	100.0	设置文本的字号大小。可动画
<text>.kerning	Float	0.0	设置文本字符的间距。可动画
<text>.leading	Float	0.0	设置多行文本的行间距。可动画
<text>.text	String	“3ds max Text” 指定视窗中的文本	
<text>.steps	Integer	6	设置每两个顶点之间短直线的数量
<text>.optimize	Boolean	True	当设置为 On 时，自动检查并移除样条曲线上的多余步数设置，以减小样条曲线的复杂程度
<text>.adaptive	Boolean	False	当设置为 On 时，将依据样条曲线的复杂程度自动指定步数，对于直线自动将步数设定为 0；对于复杂的弯曲曲线，自动加大步数使其光滑
<text>.angle	Float	0.0	设置剖面图形沿路径轴向旋转的角度。可动画
<text>.thickness	Float	1.0	设置可渲染二维图形剖面的直径。可动画
<text>.sides	Float	12.0	设置可渲染二维图形剖面的边数。可动画
<text>.viewport_thickness	Float	1.0	设置二维图形在视窗中的剖面直径
<text>.viewport_sides	Integer	12	设置二维图形在视窗中的剖面边数
<text>.viewport_angle	Float	0.0	设置剖面图形在视窗中沿路径轴向旋转的角度
<text>.DisplayRenderMesh	Boolean	False	当设置为 On 时，将可渲染的二维图形对象显示为网格对象
<text>.UseViewportSettings	Boolean	False	当设置为 On 时，二维图形对象可以被渲染
<text>.DisplayRenderSettings	Boolean	True	当设置为 On 时，依据视图设置将可渲染的二维图形对象显示为网格对象
<text>.renderable	Boolean	True	当设置为 On 时，二维图形对象可以被渲染
<text>.mapCoords	Boolean	False	当设置为 On 时，为可渲染的二维图形对象自动指定贴图坐标，默认的贴图坐标指定沿周长方向为贴图 U 轴；沿路径方向为贴图 V 轴

**注意** 在 3DS VIZ 里，属性.text 的默认值为 “VIZ Text”。

### 8.5.4 NURBS 曲线

#### 8.5.4.1 CV\_Curve: Shape (CV 曲线)

构造函数

NURBSNode <nurbs\_set> ...

属性

属性名称	数据类型	默认值	说明
<CV_Curve>.angle	Float	0.0	设置剖面图形沿 CV 曲线旋转的角度。可动画
<CV_Curve>.thickness	Float	1.0	设置可渲染 CV 曲线对象剖面的直径。可动画
<CV_Curve>.sides	Float	12.0	设置可渲染 CV 曲线对象剖面的边数。可动画
<CV_Curve>.renderable	Boolean	True	当设置为 On 时，可以将 CV 曲线对象设定为可渲染的 NURBS 曲线
<CV_Curve>.mapCoords	Boolean	False	当设置为 On 时，为可渲染的 CV 曲线对象自动指定贴图坐标，默认的贴图坐标指定沿剖面周长方向为 U 轴，沿点曲线方向为 V 轴

#### 8.5.4.2 Point\_Curve: Shape (点曲线)

构造函数

NURBSNode <nurbs\_set> ...

属性

属性名称	数据类型	默认值	说明
<Point_Curve>.angle	Float	0.0	设置剖面图形沿点曲线对象旋转的角度。可动画
<Point_Curve>.thickness	Float	1.0	设置可渲染点曲线对象剖面的直径。可动画
<Point_Curve>.sides	Float	12.0	设置可渲染点曲线对象剖面的边数。可动画
<Point_Curve>.renderable	Boolean	True	当设置为 On 时，可以将点曲线对象设定为可渲染的 NURBS 曲线
<Point_Curve>.mapCoords	Boolean	False	当设置为 On 时，为可渲染的点曲线对象自动指定贴图坐标，默认的贴图坐标指定沿剖面周长方向为 U 轴向；沿点曲线方向为 V 轴

## 8.6 Light: Node

下面列出了 3ds max 里所有 Light 类对象：

- ◆ DirectionalLight (自由平行光)
- ◆ FreeSpot (自由聚光灯)
- ◆ OmniLight (泛光灯)

- ◆ TargetSpot (目标平行光)
- ◆ TargetDirectionalLight (目标聚光灯)
- ◆ Free\_Point (自由点灯光)
- ◆ Target\_Point (目标点灯光)
- ◆ Free\_Area (自由区域灯光)
- ◆ Target\_Area (目标区域灯光)
- ◆ Free\_Linear (自由线性光)
- ◆ Target\_Linear (目标线性光)
- ◆ IES\_Sky (IES 天光)
- ◆ IES\_Sun (IES 太阳光)
- ◆ Skylight (天光)

下面的类仅在 Autodesk VIZ 里可用，与 3ds max 里的 Skylight 类执行同样的功能：

- ◆ Texture\_Sky

### 8.6.1 Light 通用属性、操作符和方法

#### 属性

属性名称	数据类型	默认值	说明
<light>.type	Name	#freeDirect	设置灯光类型： #omni: 泛光灯 #freeSpot: 自由聚光灯 #targetSpot: 目标聚光灯 #freeDirect: 自由平行光灯 #targetDirect: 目标平行光灯
<light>.enabled <light>.on	Boolean	True	控制开/关灯光
<light>.excludeList	Array	#()	设置场景中不受影响的对象序列，排除效果只在场景渲染输出时有效
<light>.includeList	Array	undefined	设置场景中受影响的对象序列
<light>.inclExclType	Integer	3	设置物体在灯光 Include/ Exclude 列表中的类型： 1: Illumination (排除或包含灯光照明的影响) 2: Shadow casting (排除或包含灯光阴影投射的影响) 3: Both (排除或包含灯光阴影投射和照明的影响)
<light>.castShadows	Boolean	False	设置灯光对象在场景中是否投射阴影
<light>.rgb <light>.color	Color	(color 180 180 180)	设置红、绿、蓝三原色光模式的红、绿、蓝颜色值。 可动画
<light>.hsv	Point3	[0,0,180]	调整色调、饱和度和值。色调设置颜色；饱和度设置颜色纯度；而值设置颜色亮度或强度
<light>.hue	Integer	0	设置纯色
<light>.saturation	Integer	0	设置颜色的纯度和强度

(续表)

属性名称	数据类型	默认值	说明
<light>.value	Integer	180	设置亮度
<light>.multiplier	Float	1.0	将灯光的功率放大一个正或负的量。使用该参数增加强度可以使颜色看起来有“烧坏”的效果。通常，将倍增设置为默认值 1.0，特殊效果和特殊情况除外。高倍增值会冲蚀颜色。例如，如果将聚光灯设置为红色，之后将其倍增增加到 10，则在聚光区中的灯光为白色并且只有在衰减区域的灯光为红色，其中并没有应用倍增。负的倍增值导致黑色灯光。即灯光使对象变暗而不是使对象变亮
<light>.contrast	Float	0.0	调整曲面的漫反射区域和环境光区域之间的对比度。增加该值可增加特殊效果的对比度。例如，太空中刺眼的灯光。可动画
<light>.softenDiffuseEdge <light>.Diffuse_Soften	Float	0.0	增加柔化漫反射边的值可以柔化曲面的漫反射部分与环境光部分之间的边缘。这样有助于消除在某些情况下曲面上出现的边缘。可动画
<light>.affectDiffuse	Boolean	True	设置灯光是否将影响对象曲面的漫反射属性
<light>.affectSpecular	Boolean	True	设置灯光是否影响对象曲面的高光属性
<light>.ambientOnly	Boolean	False	设置灯光是否仅影响照明的环境光组件
<light>.projector	Boolean	False	设置是否投射选定的贴图
<light>.projectorMap	Texture-Map	undefined	指定用于投影的贴图
<light>.useShadowProjectorMap	Boolean	False	设置是否指定阴影贴图
<light>.nearAttenStart <light>.Attenuation_Near_Start	Float	0.0	设置灯光开始淡入的距离。可动画
<light>.nearAttenEnd <light>.Attenuation_Near_End	Float	40.0	设置设置灯光达到其全值的距离。可动画
<light>.useNearAtten	Boolean	False	设置灯光是否使用近距衰减
<light>.showNearAtten	Boolean	False	设置灯光是否显示近距衰减范围线框，默认近距衰减开始线框为深蓝色，近距衰减结束线框为浅蓝色
<light>.farAttenStart <light>.Attenuation_Far_Start	Float	80.0	设置灯光开始淡出的距离。可动画
<light>.farAttenEnd <light>.Attenuation_Far_End	Float	80.0	设置灯光衰减为 0 的距离。可动画
<light>.useFarAtten	Boolean	False	设置灯光是否使用远距衰减
<light>.showFarAtten	Boolean	False	设置灯光是否显示远距衰减范围线框，默认远距衰减开始线框为深棕色，远距衰减结束线框为暗棕色

(续表)

属性名称	数据类型	默认值	说明
<light>.attenDecay	Integer	1	设置灯光衰退类型： 1: None (无, 不应用衰退。从其源到无穷大灯光仍然保持全部强度, 除非启用远距衰减) 2: Inverse (应用反向衰退。公式亮度为 $R_0/R$ , 其中 $R_0$ 为灯光的径向源 (如果不使用衰减), 为灯光的近距结束值 (如果不使用衰减)。R 为与 $R_0$ 照明曲面的径向距离) 3: Inverse square (应用平方反比衰退。该公式为 $(R_0/R)^2$ 。实际上这是灯光的真实衰退, 但在计算机图形中可能很难查找)
<light>.DecayRadius <light>.Decay_Falloff	Float	40.0	设置灯光开始淡入的距离。可动画
<light>.useGlobalShadowSettings	Boolean	False	设置是否使用灯光投射阴影的全局设置
<light>.raytracedShadows	Boolean	False	设置是否使用光线跟踪阴影
<light>.ShadowColor <light>.Shadow_Color	Color	(color 0 0 0)	设置当前灯光投射的阴影颜色。可动画
<light>.shadowMultiplier <light>.Shadow_Density	Float	1.0	设置阴影的密度。增加密度值可以增加阴影的密度 (暗度)。减少密度会减少阴影密度。强度可以有负值, 使用该值可以帮助模拟反射灯光的效果。白色阴影颜色和负密度渲染黑色阴影的质量没有黑色阴影颜色和正密度渲染的质量好。可动画
<light>.shadowProjectorMap	Texture Map	undefined	指定阴影贴图, 贴图颜色与阴影颜色相融合, 默认值为无阴影贴图
<light>.lightAffectsShadow	Boolean	False	设置是否将灯光色彩与阴影色彩互相混合
<light>.atmosShadows	Boolean	True	当设置为 On 时, 大气效果如灯光穿过它们一样投射阴影
<light>.atmosOpacity <light>.Atmosphere_Opacity	Float	100.0	调整阴影的不透明度。可动画, 百分数
<light>.atmosColorAmt <light>.Atmosphere_Color_Amount	Float	100.0	调整阴影的不透明度。此值为百分比。可动画, 百分数
<light>.mapBias <light>.map_bias	Float	1.0	设置面向或背离阴影投射对象移动阴影。可动画
<light>.mapSize <light>.map_size.	Integer	512	设置阴影贴图的尺寸, 尺寸越大阴影投射越精细。可动画
<light>.sampleRange <light>.map_range	Float	4.0	设置采样范围的大小。采样范围越大, 阴影的边缘越柔和; 采样的范围越小, 阴影的边缘越锐利, 取值范围在 0.01~50.0 之间。可动画

(续表)

属性名称	数据类型	默认值	说明
<light>.absoluteMapBias	Boolean	False	设置贴图偏移量是否采用绝对计算方式，即将灯光对象与目标对象之间的距离等分为100份。可动画
<light>.absolute_Bias <light>.absoluteMapBias	Integer	0	设置灯光对象是否使用绝对贴图偏移
<light>.raytraceBias	Float	0.2	用于面向或背离阴影投射对象移动阴影。如果偏移值太低，阴影可能在无法到达的地方“泄露”，从而生成叠纹图案或在网格上生成不合适的黑色区域。如果偏移值太高，阴影可能从对象中分离。在任何一方向上如果偏移值是极值，则阴影根本不可能被渲染。可动画
<light>.maxDepth	Integer	7	设置调整四元树的深度。增大四元树深度值可以缩短光线跟踪时间，但却以占用内存为代价。然而，使用这个深度值虽然可以改善性能，但是却要花费大量的时间才能生成四元树本身。可动画

**注意** 设置属性.includelist 或.excludelist 会自动将另一个属性设为 undefined。

### 8.6.1.1 DirectionalLight: Light (自由平行光)

构造函数

DirectionalLight ...

属性

属性名称	数据类型	默认值	说明
<DirectionalLight>.aspect <DirectionalLight>.Aspect_Ratio	Float	1.0	设置自由平行光约束矩形聚光区与衰减区的长宽比例。可动画
<DirectionalLight>.falloff	Float	45.0	设置自由平行光衰减区光锥的角度。可动画
<DirectionalLight>.showCone	Boolean	False	设置自由平行光在场景中是否显示灯光范围
<DirectionalLight>.hotspot	Float	43.0	设置自由平行光光锥的角度。可动画
<DirectionalLight>.overShoot	Boolean	False	设置是否使自由平行光具有泛光灯的特性，即能够照亮投射范围之外的其他场景对象，同时保持聚光灯投射阴影的效果，即只有在衰减范围的对象才投射阴影
<DirectionalLight>.coneShape <DirectionalLight>.lightShape	Integer	1	设置自由平行光聚光区和衰减区的形状： 1: Circle; 2: Rectangle

### 8.6.1.2 FreeSpot: Light (自由聚光灯)

构造函数

FreeSpot ...

属性

属性名称	数据类型	默认值	说明
<Freespot>.aspect <Freespot>.Aspect_Ratio	Float	1.0	设置自由聚光灯约束矩形聚光区与衰减区的长宽比例关系。可动画
<Freespot>.falloff	Float	45.0	设置自由聚光灯衰减区光锥的角度。可动画
<Freespot>.showCone	Boolean	False	设置自由聚光灯在场景中是否显示灯光范围
<Freespot>.hotspot	Float	43.0	设置自由聚光灯聚光区光锥的角度。可动画
<Freespot>.overShoot	Boolean	False	设置是否使自由聚光灯具有泛光灯的特性，即能够照亮投射范围之外的其他场景对象，同时保持聚光灯投射阴影的效果，即只有在衰减范围的对象才投射阴影
<Freespot>.coneShape <Freespot>.lightShape	Integer	1	设置聚光区和衰减区的形状: 1: Circle; 2: Rectangle

### 8.6.1.3 OmniLight: Light (泛光灯)

构造函数

omniLight ...

属性

除通用属性外，OmniLight 类没有额外的属性。

### 8.6.1.4 TargetDirectionalLight: Light (目标平行光)

构造函数

TargetDirectionalLight ...

属性

属性名称	数据类型	默认值	说明
<TargetDirectionalLight>.aspect <TargetDirectionalLight>.Aspect_Ratio	Float	1.0	设置目标平行光约束矩形聚光区与衰减区的长宽比例关系。可动画
<TargetDirectionalLight>.falloff	Float	45.0	设置目标平行光衰减区光锥的角度。可动画
<TargetDirectionalLight>.showCone	Boolean	False	设置目标平行光在场景中是否显示灯光范围
<TargetDirectionalLight>.hotspot	Float	43.0	设置聚光区光锥的角度。可动画
<TargetDirectionalLight>.overShoot	Boolean	False	设置是否使目标平行光具有泛光灯的特性，即能够照亮投射范围之外的其他场景对象，同时保持聚光灯投射阴影的效果，即只有在衰减范围的对象才投射阴影
<TargetDirectionalLight>.coneShape <TargetDirectionalLight>.lightShape	Integer	1	设置目标平行光聚光区和衰减区的形状: 1: Circle; 2: Rectangle

**注意**

在MAXScript里，必须为target类对象显式地指定一个目标对象。

例如：

```
c = TargetDirectionallight pos:[x,y,z] \
target: (targetObject pos:[xt, yt, zt])
```

### 8.6.1.5 TargetSpot: Light (目标聚光灯)

构造函数

targetSpot ...

属性

属性名称	数据类型	默认值	说明
<Targetspot>.aspect <Targetspot>.Aspect_Ratio	Float	1.0	设置目标聚光灯约束矩形聚光区与衰减区的长宽比例关系。可动画
<Targetspot>.falloff	Float	45.0	设置目标聚光灯衰减区光锥的角度。可动画
<Targetspot>.showCone	Boolean	False	设置目标聚光灯在场景中是否显示灯光范围
<Targetspot>.hotspot	Float	43.0	设置目标聚光灯衰减区光锥的角度。可动画
<Targetspot>.overShoot	Boolean	False	设置是否使目标聚光灯具有泛光灯的特性，即能够照亮投射范围之外的其他场景对象，同时保持聚光灯投射阴影的效果，即只有在衰减范围的对象才投射阴影
<Targetspot>.coneShape <Targetspot>.lightShape	Integer	1	设置目标聚光灯聚光区和衰减区的形状： 1: Circle; 2: Rectangle

### 8.6.1.6 Free\_Point: Light (自由点灯光)

构造函数

Free\_Point ...

属性

属性名称	数据类型	默认值	说明
<Free_Point>.distribution	Integer	0	描述光源发射的灯光的方向分布： 0: Isotropic (等向) 1: SpotLight (聚光灯) 2: Not supported by Point lights 3: Web

### 8.6.1.7 Target\_Point: Light (目标点灯光)

构造函数

Target\_Point ...

属性

属性名称	数据类型	默认值	说明
<Target_Point>.distribution	Integer	0	描述光源发射的灯光的方向分布: 0: Isotropic (等向) 1: SpotLight (聚光灯) 2: Not supported by Point lights 3: Web

#### 8.6.1.8 Free\_Area: Light (自由区域灯光)

构造函数

Free\_Area ...

属性

属性名称	数据类型	默认值	说明
<Free_Area>.distribution	Integer	2	描述光源发射的灯光的方向分布: 0: Isotropic (等向) 1: SpotLight (聚光灯) 2: Not supported by Point lights (不被点光源支持) 3: Web

#### 8.6.1.9 Target\_Area: Light (目标区域灯光)

构造函数

Target\_Area ...

属性

属性名称	数据类型	默认值	说明
<Target_Area>.distribution	Integer	2	描述光源发射的灯光的方向分布: 0: Not supported by Area lights (不被区域光源支持) 1: Not supported by Area lights (不被区域光源支持) 2: Diffuse 3: Web

#### 8.6.1.10 Free\_Linear: Light (自由线性光)

构造函数

Free\_Linear ...

属性

属性名称	数据类型	默认值	说明
<Free_Linear>.distribution	Integer	0	描述光源发射的灯光的方向分布: 0: Not supported by Area lights (不被区域光源支持) 1: Not supported by Area lights (不被区域光源支持) 2: Diffuse 3: Web

### 8.6.1.11 Target\_Linear: Light (目标线性光)

构造函数

Target\_Linear ...

属性

属性名称	数据类型	默认值	说明
<Target_Linear>.distribution	Integer	2	描述光源发射的灯光的方向分布: 0: Not supported by Area lights (不被区域光源支持) 1: Not supported by Area lights (不被区域光源支持) 2: Diffuse 3: Web

### 8.6.1.12 IES\_Sky: Light (IES 天光)

构造函数

IES\_Sky...

IesSkyLight...

属性

属性名称	数据类型	默认值	说明
<IES_Sky>.multiplier	Float	1.0	控制天光的强度。可动画
<IES_Sky>.sky_cover	Float	0.0	设置灯光在整个天空中散射的程度。0.0 表示晴朗，1.0 表示多云。可动画
<IES_Sky>.enabled <IES_Sky>.on	Boolean	True	控制开/关天空光
<IES_Sky>.rgb <IES_Sky>.color	Color	(color 255 255 255)	控制天空的颜色。可动画
<IES_Sky>.rays_per_sample	Integer	20	用于计算落在场景中指定点上天空光的光线数。可动画
<IES_Sky>.ray_bias	Float	0.005	设置对象在场景中指定点上投射阴影的最短距离。将该值设置为 0 可以使该点在自身上投射阴影，并且将该值设置为大的值可以防止点附近的对象在该点上投射阴影。可动画
<IES_Sky>.castShadows	Boolean	False	设置天光是否投射阴影

方法

1. isSun() <boolean>

如果显示的是 Sun Light 将返回 True；如果是 IES\_Sky 将返回 False。

2. isSky() <boolean>

如果显示的是 Sky Light 将返回 True；如果是 IES\_Sky 将返回 True。

### 8.6.1.13 IES\_Sun: Light (IES 太阳光)

构造函数

IES\_Sun...

freeIesSun...

属性

属性名称	数据类型	默认值	说明
<IES_Sun>.absoluteMapBias	Boolean	False	设置 IES 太阳光贴图偏移量是否采用绝对计算方式，即将灯光对象与目标对象之间的距离等分为 100 份。可动画
<IES_Sun>.Absolute_Bias	Integer	0	
<IES_Sun>.affectDiffuse <IES_Sun>.Affect_Diffuse	Boolean	True	设置灯光是否将影响对象曲面的漫反射属性
<IES_Sun>.affectSpecular	Boolean	True	设置灯光是否影响对象曲面的高光属性
<IES_Sun>.atmosColorAmt	Float	100.0	调整阴影的不透明度。可动画，百分数
<IES_Sun>.atmosOpacity	Float	100.0	调整阴影的不透明度。可动画，百分数
<IES_Sun>.atmosShadows	Boolean	False	当设置为 On 时，大气效果如灯光穿过他们一样投射阴影
<IES_Sun>.castShadows	Boolean	False	设置 IES 太阳光在场景中是否投射阴影
<IES_Sun>.contrast	Float	0.0	调整 IES 太阳光的曲面的漫反射区域和环境光区域之间的对比度。增加该值可增加特殊效果的对比度：例如，太空中刺眼的灯光。可动画
<IES_Sun>.hasTarget <IES_Sun>.Has_Target	Boolean	True	当设置为 On 时，IES 太阳把日光系统创建的罗盘的中心作为目标；否则，则手动设置太阳的位置
<IES_Sun>.mapbias <IES_Sun>.Map_Bias	Float	1.0	设置面向或背离阴影投射对象移动阴影。可动画
<IES_Sun>.mapsize <IES_Sun>.Map_Size	Integer	512	设置阴影贴图的尺寸，尺寸越大阴影投射越精细。可动画
<IES_Sun>.multiplier	Float	80000.0	设置灯光的强度值。可动画
<IES_Sun>.on	Boolean	True	控制开/关 IES 太阳光
<IES_Sun>.on <IES_Sun>.rgb	Color	默认值：(color 255 242.25 229.5)	
		设置 IES 太阳光的颜色值。可动画	
<IES_Sun>.samplerrange <IES_Sun>.Map_Range	Float	4.0	设置采样范围的大小
<IES_Sun>.shadowMultiplier <IES_Sun>.Shadow_Density	Float	1.0	设置阴影的密度。增加密度值可以增加阴影的密度（暗度）。减少密度会减少阴影密度。强度可以有负值，使用该值可以帮助模拟反射灯光的效果。白色阴影颜色和负密度渲染黑色阴影的质量没有黑色阴影颜色和正密度渲染的质量好。可动画

(续表)

属性名称	数据类型	默认值	说明
<IES_Sun>.softenDiffuseEdge <IES_Sun>.Diffuse_Soften	Float	50.0	用于柔化对象过渡区与阴影区的边缘，避免明显的明暗边界。可动画，百分数
<IES_Sun>.targetDistance <IES_Sun>.Target_Distance	Float	240.0	设置太阳光的强度。可动画
<IES_Sun>.twoSidedShadows	Boolean	False	设置是否选择双面阴影。当设置为On时，物体背面不会被忽略，但是影响着色时间
<IES_Sun>.useGlobalShadowSettings <IES_Sun>.Use_Global_Shadow_Settings	Boolean	False	设置是否使用IES太阳光投射阴影的全局设置

### 方法

#### 1. isSun() <boolean>

如果显示的是Sun Light将返回True；如果是IES\_Sun将返回True。

#### 2. isSky() <boolean>

如果显示的是Sky Light将返回True；如果是IES\_Sun将返回False。

例如：

```
sun = freeIesSun()
sun.NaturalLightClass.isSun() --返回 True
sun.isSky() --返回 False
```

### 8.6.1.14 Skylight: Light(天空光)

#### 构造函数

Skylight...

TexSkyLight...

#### 属性

属性名称	数据类型	默认值	说明
<Skylight>.multiplier	Float	1.0	设置天空光的强度值。可动画
<Skylight>.on <Skylight>.enabled	Boolean	True	控制开/关天空光
<Skylight>.sky_color_map	TextureMap	undefined	设置天空光颜色贴图
<Skylight>.sky_color_map_on	Boolean	True	设置是否使用天空光颜色贴图
<Skylight>.sky_color_map_amt	Float	100.0	设置天空光颜色贴图的透明度，当设置为0时，仅显示天空颜色；当设置为100时，仅显示贴图。可动画

(续表)

属性名称	数据类型	默认值	说明
<Skylight>.sky_mode <Skylight>.Skylight_Mode	Integer	1	设置影响天空颜色的方式： 0: Use Scene Environment(使用场景环境); 1: Sky Color(天空颜色)
<Skylight>.color <Skylight>.rgb	Color	(color 255 255 255)	设置天空光的颜色值。可动画
<Skylight>.rays_per_sample	Integer	20	设置每采样光线的数目。可动画
<Skylight>.ray_bias	Float	0.005	设置光线的偏移值。可动画
<Skylight>.castShadows	Boolean	False	设置天光在场景中是否投射阴影

### 方法

#### 1. isSun() <boolean>

如果显示的是 Sun Light 将返回 True; 如果是 Skylight 将返回 False。

#### 2. isSky() <boolean>

如果显示的是 Sky Light 将返回 True。

### 8.6.1.15 Texture\_Sky: Light (天光)

#### 构造函数

Texture\_Sky ...

#### 属性

属性名称	数据类型	默认值	说明
<Texture_Sky>.on <Texture_Sky>.enabled	Boolean	True	控制开/关灯光
<Texture_Sky>.multiplier	Float	1.0	设置天光的强度值。可动画
<Texture_Sky>.sky_mode <Texture_Sky>.Skylight_Mode	Integer	1	设置影响天空颜色的方式： 0: Use Scene Environment(使用场景环境); 1: Sky Color(天空颜色)
<Texture_Sky>.color	Color	(color 242.25 242.25 255)	当天光方式为 Sky Color 时, 为设置天空的 RGB 颜色值。可动画
<Texture_Sky>.rgb <Texture_Sky>.color	Color	(color 255 255 255)	当天光方式为 Sky Color 时, 为设置天光的 RGB 颜色值。可动画
<Texture_Sky>.affectDiffuse <Texture_Sky>.Affect_Diffuse	Boolean	True	设置灯光是否将影响对象曲面的漫反射属性
<Texture_Sky>.affectSpecular	Boolean	False	设置灯光是否影响对象曲面的高光属性
<Texture_Sky>.castShadows	Boolean	True	设置对象在场景中是否投射阴影

(续表)

属性名称	数据类型	默认值	说明
<Texture_Sky>.exponent <Texture_Sky>.Specular_Exponent	Float	30.0	设置影响对象曲面高光属性的采样指数。可动画
<Texture_Sky>.sky_color_map_on	Boolean	True	设置是否使用天光颜色贴图
<Texture_Sky>.sky_color_map_amt	Float	100.0	设置天光颜色贴图的透明度,当设置为0时,仅显示天空颜色;当设置为100时,仅显示贴图。可动画
<Texture_Sky>.sky_color_map	undefined	undefined	设置是否使用天光颜色贴图
<Texture_Sky>.rays_per_sample	Integer	4	设置每采样光线的数目。可动画
<Texture_Sky>.ray_bias	Float	0.01	设置光线的偏移值。可动画
<Texture_Sky>.rayengine	Ray _Engine	默认值: ReferenceTarget:Ray_Engine	
		设置光线的引擎	

## 8.7 Camera: Node

下面列出了所有 Camera 类对象:

- ◆ FreeCamera (自由摄影机)
- ◆ TargetCamera (目标摄影机)

### 8.7.1 Camera 通用属性

#### 属性

下面属性适用于所有 Camera 类对象:

属性名称	数据类型	默认值	说明
<camera>.fov	Float	45.0	设置摄影机查看区域的宽度(视野)。可动画, Angle
<camera>.orthoProjection	Boolean	False	当设置为 On 时, 摄影机视图如同用户视图一样; 否则, 摄影机视图如同透视图一样
<camera>.type	Name	#free	设置摄影机类型: #free-Free Camera: (目标摄影机) #target-Target: (自由摄影机)
<camera>.showCone	Boolean	False	设置是否显示摄影机视野定义的锥形光线(实际上是一个四棱锥)。锥形光线出现在其他视口但是不出现在摄影机视口中
<camera>.showHorizon	Boolean	False	设置在摄影机视图中是否显示一条深灰色的地平线
<camera>.nearrange <camera>.Near_Env_Range	Float	0.0	设置大气效果的近距范围。可动画

(续表)

属性名称	数据类型	默认值	说明
<camera>.farrange <camera>.Far_Env_Range	Float	1000.0	设置大气效果的远距范围，对象在两个范围之间依据距离百分比进行淡化处理。可动画
<camera>.clipManually	Boolean	False	当设置为 On 时，可以手动定义截面矩形；反之，则场景中的对象与摄影机的距离小于三个单位时，该对象在摄影机视图中不显示
<camera>.nearclip <camera>.near_clip	Float	1.0	设置近距截面矩形。可动画
<camera>.farclip <camera>.far_clip	Float	1000.0	设置远距截面矩形。注意：过高的远距剪切设置会导致浮点计算错误，导致视图中的 Z-buffer(Z 缓冲)通道出现问题。可动画
<camera>.showRanges	Boolean	False	设置在摄影机光锥中是否显示一个黄色的矩形，以标明近距距离与远距距离的参数设置
<camera>.targetDistance <camera>.Target_Distance	Float	160.0	设置一个不可见的拍摄目标点，自由摄影机可以围绕该点盘旋拍摄。可动画

**注意** 属性.fov 为水平 fov，单位为度。垂直 fov 和对角 fov 及透镜值均基于水平 fov 和当前渲染光圈宽度。

下面脚本可以从水平 fov 计算垂直 fov：

```
fn GetCamVFOV theCamera =
(local r_aspect=(renderWidth as Float)/renderHeight
2.0*atan (tan(theCamera.fov/2.0)/r_aspect)
)
```

下面例子通过改变指定 Camera 来模拟 Zoom Extents All 命令的效果。

```
fn ZE_Cam cam objs =
--声明局部变量
(local max2, fov, asp, v
--定义一个函数，返回一个数组里数值集的最大值
fn maxof vals =(local v=vals[1]
for v1 in vals do(if v1 > v do v=v1)
v) --函数 maxof 定义结束
--初始化 fov 值
fov=0
--计算 Renderer 的图像 Aspect Ratio
asp=(renderWidth as Float)/renderHeight
--设置在 Camera 坐标系里进行工作
in coordsys cam
--对除 Camera 以外的所有对象循环
(for obj in objs where obj != cam do
(if obj.min.z >=0 do continue --如果对象在 Camera 后面，跳过它
--获取对象边界框极限值以修正图像纵横比 (aspect ratio)
v = maxof #((abs obj.max.x),(abs obj.min.x),\
(abs(obj.max.y*asp)),(abs(obj.min.y*asp)))
--如有需要，加大 fov
fov = maxof #(fov,(2*atan(-v/obj.min.z)))
```

```

) --for 循环结束
) --in coordsys 关联语句结束
cam.fov=fov      --将 Camera 的属性.fov 设为新值
)
--测试 --
cam=$camera01    --指定进行 zoom extent all 模拟操作的 Camera
ZE_Cam cam $*    --函数将调用 Camera 和由所有对象组成的对象集

```

### 8.7.1.1 FreeCamera: Camera (自由摄影机)

构造函数

FreeCamera ...

属性

除通用属性外, FreeCamera 类没有额外的属性。

### 8.7.1.2 TargetCamera: Camera (目标摄影机)

构造函数

TargetCamera ...

属性

除通用属性外, TargetCamera 类没有额外的属性。

## 8.8 Helper: Node

下面列出了所有的 Helper 类对象:

- ◆ Standard (标准辅助对象)
  - Bone (骨骼系统)
  - Compass (指南针辅助对象)
  - Dummy (虚拟辅助对象)
  - Grid (栅格辅助对象)
  - Point (点辅助对象)
  - Protractor (量角器辅助对象)
  - Tape (卷尺辅助对象)
  - Atmospheric (大气装置):
  - BoxGizmo (长方体 Gizmo)
  - CylGizmo (圆柱体 Gizmo)
  - SphereGizmo (球体 Gizmo)
- ◆ Camera Match (摄影机匹配辅助对象)
  - CamPoint (摄影机点辅助对象)
- ◆ VRML 1.0/VRBL
  - Inline (内嵌辅助对象)
  - LOD (LOD 辅助对象)
  - VRML\_VRBL

- ◆ VRML97 (VRML97 辅助对象)
  - Anchor (锚定 VRML97 辅助对象)
  - AudioClip (声音 VRML97 辅助对象)
  - Background (背景 VRML97 辅助对象)
  - Billboard (布告牌 VRML97 辅助对象)
  - FogHelper (雾 VRML97 辅助对象)
  - InlineHelper (内嵌 VRML97 辅助对象)
  - LODHelper (LOD VRML97 辅助对象)
  - NavInfo (漫游信息 VRML97 辅助对象)
  - ProxSensor (范围感应器 VRML97 辅助对象)
  - Sound (音频剪辑 VRML97 辅助对象)
  - TouchSensor (触动感应器 VRML97 辅助对象)
  - TimeSensor (时间感应器 VRML97 辅助对象)
- ◆ Miscellaneous (操纵器辅助对象)
  - LinkOriginPtHelper

### 8.8.1 Bone: Helper (骨骼系统)

#### 构造函数

Bone ...

#### 属性

除通用属性外，Bone 类没有额外的属性。

**注意** Bone 类辅助对象在 3ds max 里用来代表一个 Bones system 里的 Node。在 3ds max 4 和更高版本里，还不能在 Bone 对象的 Node 上用 IK Controller 创建一个 Bones system。

可以用 PRS 转换控制器为每一 Node 创建一个 Bone 类辅助对象的层级。如：

```
b0 = bone pos:[x,y,z]
in b0 b1 = bone pos:[x1,y1,z1]
in b1 b2 = bone pos:[x2,y2,z2]
in b2 b3 = bone pos:[x3,y3,z3]
```

可通过属性.Parent 来重新安排 Bone 对象的层级。

如果建立了一个 Bone 对象层级，属性.showLinks 和.showLinksOnly 被自动设为 True。

### 8.8.2 Compass: Helper (指南针辅助对象)

#### 构造函数

Compass ...

#### 属性

除通用属性外，Compass 类没有额外的属性。

### 8.8.3 Dummy: Helper (虚拟辅助对象)

#### 构造函数

Dummy ...

#### 属性

<Dummy>.boxsize Point3, 默认值: [10,10,10]

设置图标大小。

### 8.8.4 Grid: Helper (栅格辅助对象)

#### 构造函数

Grid ...

#### 属性

属性名称	数据类型	默认值	说明
<grid>.length	Float	50.0	指定栅格平面的长度。可动画
<grid>.width	Float	160.0	指定栅格平面的宽度。可动画
<grid>.grid	Float	10.0	指定栅格线的间距。可动画

### 8.8.5 Point: Helper (点辅助对象)

#### 构造函数

Point ...

#### 属性

属性名称	数据类型	默认值	说明
<Point>.axisLength	Float	20.0	指定点对象坐标三角架的长度
<Point>.size	Float	20.0	指定点对象的显示尺寸。可动画
<Point>.centermarker	Boolean	False	当设置为 On 时, 在点对象的几何中心显示一个小 X 标记。可动画
<Point>.axistripod	Boolean	False	当设置为 On 时, 显示一个小的坐标三角架, 标定点对象的位置和方向。可动画
<Point>.cross	Boolean	True	当设置为 On 时, 显示轴的交叉十字。可动画
<Point>.box	Boolean	False	当设置为 On 时, 在点对象的几何中心显示一个方块。可动画
<Point>.constantscreensize	Boolean	False	当设置为 On 时, 不论如何放缩显示视图, 点对象都保持原有的显示尺寸。可动画
<Point>.drawontop	Boolean	False	当设置为 On 时, 将点对象显示在场景中所有对象的上方。可动画

**注意** 有关色彩和显示的属性不能用 MAXScript 进行存取。

### 8.8.6 Protractor: Helper (量角器辅助对象)

构造函数

protractor ...

属性

除通用属性外，Protractor 类没有额外的属性。

**注意** 在 3ds max 的 MAXScript 里没有办法设置 Protractor 对象的参考对象。

### 8.8.7 Tape: Helper (卷尺辅助对象)

构造函数

tape ...

属性

<Tape>.length: Float, 默认值: 50.0, 可动画

Tape 对象的长度。

**注意** 为了构造一个 Tape 对象，必须先用构造函数 targetObject() 创建一个目标对象，然后用这个目标对象来构造 Tape 对象：

```
tape pos:p target: (targetObject pos:q)
```

属性.length 为在 3ds max 用户界面里设置的 Tape 长度，而不是从 Tape 对象到其 Target 对象之间的距离。下面的函数可以用来返回 Tape 对象到其 Target 对象之间的距离：

```
fn getTapeDist TapeObj = distance TapeObj TapeObj.target  
getTapeDist $tape01
```

### 8.8.8 Helper-Atmospheric (大气装置)

下面列出了所有 Atmospheric 类辅助对象：

- ◆ BoxGizmo (长方体 Gizmo)
- ◆ CylGizmo (圆柱体 Gizmo)
- ◆ SphereGizmo (球体 Gizmo)

#### 8.8.8.1 BoxGizmo: Helper (长方体 Gizmo)

构造函数

BoxGizmo ...

属性

属性名称	数据类型	默认值	说明
<BoxGizmo>.length	Float	0.0	指定长方体 Gizmo 底面的长度。可动画
<BoxGizmo>.width	Float	0.0	指定长方体 Gizmo 底面的宽度。可动画
<BoxGizmo>.height	Float	0.0	指定长方体 Gizmo 的高度。可动画
<BoxGizmo>.seed <BoxGizmo>.parameter	Integer	0	如果场景中存在多个大气装置，不同的种子数可以使它们产生随机的变化，避免一模一样的形态

### 8.8.8.2 CylGizmo: Helper (圆柱体 Gizmo)

构造函数

cylGizmo ...

属性

属性名称	数据类型	默认值	说明
<CylGizmo>.radius	Float	0.0	设置圆柱体 Gizmo 底面的半径。可动画
<CylGizmo>.height	Float	0.0	设置圆柱体 Gizmo 的高度。可动画
<CylGizmo>.seed <CylGizmo>.parameter	Integer	0	如果场景中存在多个大气装置，不同的种子数可以使它们产生随机的变化，避免一模一样的形态

### 8.8.8.3 SphereGizmo: Helper (球体 Gizmo)

构造函数

sphereGizmo ...

属性

属性名称	数据类型	默认值	说明
<SphereGizmo>.radius	Float	0.0	设置球体 Gizmo 的半径。可动画
<SphereGizmo>.hemisphere	Integer	0	设置是否创建一个半球 Gizmo 大气装置： 0: Off; 1: On
<SphereGizmo>.seed	Integer	0	如果场景中存在多个大气装置，不同的种子数可以使它们产生随机的变化，避免一模一样的形态

## 8.8.9 Helper-Camera Match (摄影机匹配)

Camera match (摄影机匹配) 类辅助对象仅有一种：

◆ CamPoint

构造函数

CamPoint ...

属性

属性名称	数据类型	默认值	说明
<CamPoint>.showAxis	Boolean	True	控制三轴架是否与摄影机点对象一起显示
<CamPoint>.axisLength	Float	20.0	控制三轴架的长度。当显示三轴架处于启用状态时，可以在调整微调器箭头时在视口中观察三轴架更改长度；当显示三轴架处于禁用状态时，仍然可以调整轴长度值，但不显示三轴架

### 8.8.10 Helper-VRML 1.0/VRBL

下面列出了所有 VRML 1.0/VRBL 类辅助对象：

- ◆ **Inline**（内嵌辅助对象）
- ◆ **LOD**（LOD 辅助对象）
- ◆ **VRML\_VRBL**

#### 8.8.10.1 Inline: Helper（内嵌辅助对象）

构造函数

Inline ...

属性

除通用属性外，Inline 类对象没有额外的属性。

#### 8.8.10.2 LOD: Helper（LOD 辅助对象）

构造函数

LOD ...

属性

除通用属性外，LOD 类对象没有额外的属性。

#### 8.8.10.3 VRML\_VRBL: Helper

构造函数

VRML\_VRBL...

属性

除通用属性外，VRML\_VRBL 类对象没有额外的属性。

### 8.8.11 Anchor: Helper

构造函数

Anchor ...

### 属性

除通用属性外，Anchor类对象没有额外的属性。

## 8.8.12 AudioClip: Helper (锚定VRML97辅助对象)

### 构造函数

AudioClip ...

### 属性

除通用属性外，AudioClip类对象没有额外的属性。

## 8.8.13 Background: Helper (背景VRML97辅助对象)

### 构造函数

Background ...

### 属性

除通用属性外，Background类对象没有额外的属性。

## 8.8.14 Billboard: Helper (布告牌VRML97辅助对象)

### 构造函数

billboard ...

### 属性

除通用属性外，Billboard类对象没有额外的属性。

## 8.8.15 FogHelper: Helper (雾VRML97辅助对象)

### 构造函数

FogHelper ...

### 属性

除通用属性外，FogHelper类对象没有额外的属性。

## 8.8.16 InlineHelper: Helper (内嵌VRML97辅助对象)

### 构造函数

InlineHelper ...

### 属性

除通用属性外，InlineHelper 类对象没有额外的属性。

## 8.8.17 LODHelper: Helper (LOD VRML97 辅助对象)

### 构造函数

LODHelper ...

### 属性

除通用属性外，LODHelper 类对象没有额外的属性。

## 8.8.18 NavInfo: Helper (漫游信息 VRML97 辅助对象)

### 构造函数

NavInfo ...

### 属性

除通用属性外，NavInfo 类对象没有额外的属性。

## 8.8.19 ProxSensor: Helper (范围感应器 VRML97 辅助对象)

### 构造函数

ProxSensor ...

### 属性

除通用属性外，ProxSensor 类对象没有额外的属性。

## 8.8.20 Sound: Helper (音频剪辑 VRML97 辅助对象)

### 构造函数

Sound ...

### 属性

#### 1. WAVsound.filename

存取音轨文件名，数据类型为 String。

#### 2. WAVsound.start

存取音轨开始时间，数据类型为 Time。

#### 3. WAVsound.end

只读参数，表示音轨结束时间，数据类型为 Time。

#### 4. WAVsound.mute

存取音轨当前是否活动，数据类型为 Boolean。

#### 5. WAVsound.isPlaying

只读参数，表示音轨当前是否在播放，数据类型为 Boolean。

#### 方法

**WAVsound.scrub <interval>**

将指定音轨播放指定时间长度<interval>。

### 8.8.21 TimeSensor: Helper (时间感应器 VRML97 辅助对象)

#### 构造函数

TimeSensor...

#### 属性

除通用属性外，TimeSensor类对象没有额外的属性。

### 8.8.22 TouchSensor: Helper (触动感应器 VRML97 辅助对象)

#### 构造函数

TouchSensor ...

#### 属性

除通用属性外，TouchSensor类对象没有额外的属性。

## 8.9 System: Node (系统)

在 MAXScript 里不能创建 System 类对象，但可以存取与 System 类对象有关的各种属性，下面列出了所有 3ds max 里的所有 System 类对象：

- ◆ Bones (骨骼系统)
- ◆ Sunlight (太阳光系统)
- ◆ RingArray (环形阵列系统)

### 8.9.1 Bones: System (骨骼系统)

Bones类对象不能用MAXScript来创建。

一个 Bones 对象由下面对象组成：Bone 辅助对象、一个主 IK 控制器、多个从 IK 控制器。从 IK 控制器用来作为 IK 系统里所有 Node 对象的 Transform 控制器，而主 IK 控制器用来控制单独的从 IK 控制器。与 IK 控制器有关的方法参见本书 12.10.10 节“IK\_ControllerMatrix3Controller: Matrix3Controller (反向动力学控制器)”。

### 8.9.2 Sunlight: System (太阳光系统)

Sunlight 类对象不能用 MAXScript 来创建。

一个 Sunlight 类对象由一个 Compass 辅助对象和一个 TargetDirectionalLight 对象组成, TargetDirectionalLight 对象的转换 (Transform) 由一个 Float 控制器集控制, 这些控制器都是一个不可存取的 Position 控制器的子控制器。这些子控制器可以作为 TargetDirectionalLight 对象的 subAnim 进行存取, 下面列出了这些子控制器与其对应的 subAnim 对应号:

Solar\_Time—subAnim[3]

Solar\_Date—subAnim[4]

Latitude—subAnim[5]

Longitude—subAnim[6]

Orbital\_Scale—subAnim[7]

比如: 如果对象 Sun01 是某一 Sunlight system 的灯光, 我们可以用下面方法存取 Solar\_Time 控制器:

```
solarTimeCont=$Sun01[3].object
```

### 8.9.3 RingArray: System (环形阵列系统)

RingArray 类不能用 MAXScript 来创建。

一个 RingArray 对象由一个 Dummy 辅助对象和一个数量可变的 Box 对象、一个主控制器、一些从控制器组成。从控制器用来作为所有 Box 对象的 Transform 控制器, 而主控制器用来控制单独的从控制器。与从控制器有关的方法请读者参见本书 12.10.24 节“Slave\_Control”。

## 8.10 SpacewarpObject: Node (空间扭曲)

下面列出了所有的 Spacewarp 对象:

- ◆ Geometric/Deformable (几何/可变形)
  - Bomb (爆炸空间扭曲)
  - Conform (一致空间扭曲)
  - SpaceDisplace (位移空间扭曲)
  - SpaceFFDBox (FFD 长方体空间扭曲)
  - SpaceFFDCyl (FFD 柱体空间扭曲)
  - SpaceRipple (涟漪空间扭曲)
  - SpaceWave (波浪空间扭曲)
- ◆ Particle and Dynamic (粒子系统和动力学系统)
  - Gravity (重力空间扭曲)
  - Motor (马达空间扭曲)

- PBomb (粒子爆炸空间扭曲)
- Push (推力空间扭曲)
- Wind (风力空间扭曲)
- ◆ Modifier-Based (基于修改器)
  - SpaceBend (弯曲修改器)
  - SpaceNoise (噪波修改器)
  - SpaceSkew (倾斜修改器)
  - SpaceStretch (挤出修改器)
  - SpaceTaper (锥化修改器)
  - SpaceTwist (扭曲修改器)
- ◆ Dynamics Interface (动力学导向器)
  - PDynaFlect (动力学导向板空间扭曲)
  - SDynaFlect (动力学导向球空间扭曲)
  - UDynaFlect (通用动力学导向器空间扭曲)
- ◆ Particles Only (粒子导向器)
  - Deflector (导向器空间扭曲)
  - Path\_Follow (路径跟随空间扭曲)
  - POmniFlect (泛方向导向板空间扭曲)
  - SDeflector (导向球空间扭曲)
  - SOmniFlect (泛方向导向球空间扭曲)
  - UDeflector (通用导向器空间扭曲)
  - UOmniFlect (通用泛方向导向器空间扭曲)

### 8.10.1 Bomb: SpacewarpObject (爆炸空间扭曲)

#### 构造函数

Bomb ...

#### 属性

属性名称	数据类型	默认值	说明
<Bomb>.strength	Float	1.0	设置爆炸力。较大的数值能使粒子飞得更远。对象离爆炸点越近，爆炸的效果越强烈。可动画
<Bomb>.spin	Float	0.0	指定碎片旋转的速率，以每秒转数表示。这也会受混乱度参数（使不同的碎片以不同的速度旋转）和衰减参数（使碎片离爆炸点越远时爆炸力越弱）的影响。可动画
<Bomb>.falloff	Float	100.0	指定爆炸效果距爆炸点的距离，以世界单位数表示。超过该距离的碎片不受强度和自旋设置影响，但会受重力设置影响
<Bomb>.fallOffEnable	Boolean	False	当设置为 On 时，启用衰减设定，在场景中出现黄色的球体范围线框
<Bomb>.minFragmentSize	Integer	1	指定由炸开随机生成的每个碎片的最小面数

(续表)

属性名称	数据类型	默认值	说明
<Bomb>.maxFragmentSize	Integer	1	指定由炸开随机生成的每个碎片的最大面数
<Bomb>.Gravity	Float	1.0	指定由重力产生的加速度。注意重力的方向总是指向 World 坐标系的 Z 轴方向。重力可以为负
<Bomb>.chaos	Float	0.0	随机指定爆炸后碎片的运动方向，模拟真实的方向混乱效果。当设置为 0 时，混乱度不起作用；当设置为 1.0 时，产生真实的方向混乱效果。可动画
<Bomb>.detonation	Time	5f	指定起爆的帧数，在此帧之前对象不受爆炸空间扭曲的影响
<Bomb>.seed	Integer	0	指定随机数。设置为 0 时，为完全均匀；当设置为 1.0 时，具有真实感；当设置大于 1.0 的数值时，会使爆炸效果特别混乱。取值范围为 0~10.0

### 相关方法

`bindSpaceWarp <node> <bomb_Node>`

将指定对象<node>与指定 SpaceWarp 对象<bomb\_Node>绑定。

### 有关的 Binding Modifier

`BombBinding`

本 Modifier 由方法 `bindSpaceWarp()` 自动创建，而不能由 MAXScript 直接创建。没有与本 Modifier 相关的属性。

## 8.10.2 ConformSpaceWarp: SpacewarpObject (一致空间扭曲)

### 构造函数

`ConformSpaceWarp ...`

### 属性

属性名称	数据类型	默认值	说明
<ConformSpaceWarp>.Projection_Distance	Float	0.0	绑定对象中的顶点在未与目标对象相交的情况下距离其原始位置的移动距离。可动画
<ConformSpaceWarp>.Standoff_Distance	Float	1.0	指定顶点和目标对象的表面之间所保持的距离。例如，如果将其设置为 5，就不会把顶点推动至离目标对象的表面小于 5 个单位的位置。可动画
<ConformSpaceWarp>.Selected_Verts	Integer	0	当设置为 On 时，只有包裹对象表面选定的顶点进行拟合化操作；当设置为 Off 时，包裹对象表面所有的顶点都进行拟合化操作。可动画 0: Off; 1: On
<ConformSpaceWarp>.Icon_Size	Float	0.0	指定图标的显示尺寸

## 相关方法

`bindSpaceWarp <node> <conformSpaceWarp_Node>`

将指定对象`<node>`与指定 SpaceWarp 对象绑定。

## 相关的 Binding Modifier

`SpaceConform`

本 Modifier 由方法 `bindSpaceWarp()`自动创建，而不能由 MAXScript 直接创建。没有与本 Modifier 相关的属性。

## 8.10.3 SpaceDisplace: SpacewarpObject (位移空间扭曲)

### 构造函数

`SpaceDisplace ...`

### 属性

属性名称	数据类型	默认值	说明
<code>&lt;SpaceDisplace&gt;.strength</code>	Float	0.0	设置位移空间扭曲的强度。设置为 0.0 时，位移扭曲没有任何效果；大于 0.0 的值会使对象几何体或粒子按偏离位移空间扭曲对象所在位置的方向发生位移；小于 0.0 的值会使几何体朝扭曲位移。可动画
<code>&lt;Spacedisplace&gt;.decay</code>	Float	0.0	设置位移空间扭曲作用范围的衰减量。默认情况下，位移扭曲在整个世界空间内有相同的强度。增加衰退值会导致位移强度从位移扭曲对象的所在位置开始随距离的增加而减弱。可动画
<code>&lt;Spacedisplace&gt;.lumCenterEnable</code>	Boolean	False	开/关使用亮度中心
<code>&lt;Spacedisplace&gt;.lumCenter</code>	Float	0.5	默认情况下，位移空间扭曲通过使用中等（50%）灰色作为零位移值来定义亮度中心。大于 128 的灰色值以向外的方向（背离位移扭曲对象）进行位移；而小于 128 的灰色值以向内的方向（朝向位移扭曲对象）进行位移。取值范围为 0~1.0
<code>&lt;Spacedisplace&gt;.bitmap</code>	Bitmap	undefined	指定位图
<code>&lt;Spacedisplace&gt;.map</code>	Texture-Map	undefined	指定贴图
<code>&lt;Spacedisplace&gt;.blur</code>	Float	0.0	增加该值可以模糊或柔化位图位移的效果。可动画
<code>&lt;Spacedisplace&gt;.maptype</code>	Integer	0	设置贴图选项： 0: Planar 1: Cylindrical 2: Spherical 3: Shrink Wrap
<code>&lt;Spacedisplace&gt;.length</code>	Float	1.0	指定空间扭曲 gizmo 的边界框长度。可动画
<code>&lt;Spacedisplace&gt;.width</code>	Float	1.0	指定空间扭曲 gizmo 的边界框宽度。可动画

(续表)

属性名称	数据类型	默认值	说明
<Spacedisplace>.height	Float	1.0	指定空间扭曲 gizmo 的边界框高度。可动画
<Spacedisplace>.U_Flip	Boolean	False	指定贴图是否在 U 轴方向反转
<Spacedisplace>.U_Tile	Float	1.0	指定在 U 轴方向的重复数量。可动画
<Spacedisplace>.V_Flip	Boolean	False	指定贴图是否在 V 轴方向反转
<Spacedisplace>.V_Tile	Float	1.0	指定在 V 轴方向的重复数量。可动画
<Spacedisplace>.W_Flip	Boolean	False	指定贴图是否在 W 轴方向反转
<Spacedisplace>.W_Tile	Float	1.0	指定在 W 轴方向的重复数量。可动画
<Spacedisplace>.axis	Integer	0	
<Spacedisplace>.cap	Boolean	False	
<Spacedisplace>.useMap	Boolean	False	
<Spacedisplace>.applyMap	Boolean	False	

### 相关方法

bindSpaceWarp <node> <spaceDisplace\_Node>

相关的 Binding Modifier

DisplaceBinding

**注意** 如果将一个 TextureMap 类值赋给属性.map，一个 Displacement\_Map subAnim 会被自动添加到参数里。

## 8.10.4 SpaceFFDBox：SpacewarpObject (FFD 长方体空间扭曲)

### 构造函数

SpaceFFDBox...

### 属性

属性名称	数据类型	默认值	说明
<SpaceFFDBox>.length	Float	0.0	指定长方体自由空间扭曲晶格的长度。可动画
<SpaceFFDBox>.width	Float	0.0	指定长方体自由空间扭曲晶格的宽度。可动画
<SpaceFFDBox>.height	Float	0.0	指定长方体自由空间扭曲晶格的高度。可动画
<SpaceFFDBox>.dispLattice <SpaceFFDBox>.Lattice	Boolean	True	当设置为 On 时，会绘制连接控制点的线条以形成栅格。虽然绘制这些额外的线条会使视口显得混乱，但可以使晶格形象化
<SpaceFFDBox>.dispSource <SpaceFFDBox>.Source_Volume	Boolean	False	当设置为 On 时，控制点和晶格会以未修改的状态显示。当调整源体积以影响位于其内或其外的特定顶点时，该显示很有用

(续表)

属性名称	数据类型	默认值	说明
<SpaceFFDBox>.deformType	Integer	0	设置顶点受 FFD 影响变形的方式: 0: Only In Volume (只有处于长方体自由空间扭曲中的顶点受到变形影响) 1: All Vertices (对象表面的所有顶点都受长方体自由空间扭曲的变形影响)
<SpaceFFDBox>.falloff	Float	0.0	指定了 FFD 效果减为零时离晶格的距离。当设置为 0 时，它实际处于关闭状态，不存在衰减，也就是说，所有顶点无论到晶格的距离远近都会受到影响。衰减参数的单位是相对于晶格的大小指定的：衰减值 1 表示那些到晶格的距离为晶格的宽度/长度/高度的点（具体情况取决于点位于晶格的哪一侧）所受的影响降为 0。可动画
<SpaceFFDBox>.tension	Float	25.0	调整变形样条曲线的张力。可动画
<SpaceFFDBox>.continuity	Float	25.0	调整变形样条曲线的连续性。可动画

### 相关方法

bindSpaceWarp <node> <spaceFFDBox\_Node>

### 相关的 Binding Modifier

#### FFD\_Binding

**注意** 由 MAXScript 创建的 SpaceFFDBox 的控制点数目总是  $4 \times 4 \times 4$ 。在 3ds max 的 MAXScript 里没有方法用来给 FFD 的控制点添加控制器。

## 8.10.5 SpaceFFDCyl: SpacewarpObject (FFD 柱体空间扭曲)

### 构造函数

SpaceFFDCyl ...

### 属性

属性名称	数据类型	默认值	说明
<SpaceFFDCyl>.radius	Float	0.0	指定柱体自由空间扭曲晶格的半径。可动画
<SpaceFFDCyl>.height	Float	0.0	指定柱体自由空间扭曲晶格的高度。可动画
<SpaceFFDCyl>.dispLattice <SpaceFFDCyl>.Lattice	Boolean	True	当设置为 On 时，会绘制连接控制点的线条以形成栅格。虽然绘制这些额外的线条会使视口显得混乱，但可以使晶格形象化
<SpaceFFDCyl>.dispSource <SpaceFFDCyl>.Source_Volume	Boolean	False	当设置为 On 时，控制点和晶格会以未修改的状态显示。当调整源体积以影响位于其内或其外的特定顶点时，该显示很有用
<SpaceFFDCyl>.deformType	Integer	0	设置顶点受 FFD 影响变形的方式: 0: Only In Volume (只有处于圆柱体自由空间扭曲中的对象顶点受到变形影响) 1: All Vertices (对象表面的所有顶点都受圆柱体自由空间扭曲的变形影响)

(续表)

属性名称	数据类型	默认值	说明
<SpaceFFDCyl>.falloff	Float	0.0	指定 FFD 效果减为零时离晶格的距离。可动画
<SpaceFFDCyl>.tension	Float	25.0	调整变形样条曲线的张力。可动画
<SpaceFFDCyl>.continuity	Float	25.0	调整变形样条曲线的连续性。可动画

### 相关方法

bindSpaceWarp <node> <spaceFFDCyl\_Node>

#### 相关的 Binding Modifier

##### FFD\_Binding

本 Modifier 由方法 bindSpaceWarp() 自动创建，而不能由 MAXScript 直接创建。

**注意** 由 MAXScript 创建的 SpaceFFDCyl 的控制点数目总是  $4 \times 8 \times 4$ 。在 3ds max 的 MAXScript 里没有方法用来给 FFD 的控制点添加控制器。

## 8.10.6 SpaceRipple: SpacewarpObject (涟漪空间扭曲)

### 构造函数

SpaceRipple...

### 属性

属性名称	数据类型	默认值	说明
<Spaceripple>.amplitude1 <Spaceripple>.Amplitude_1	Float	5.0	设定沿涟漪扭曲对象 Local 坐标系 X 轴的涟漪振幅。振幅用活动单位数表示。可动画
<Spaceripple>.amplitude2 <Spaceripple>.Amplitude_2	Float	5.0	设定沿涟漪扭曲对象 Local 坐标系 Y 轴的涟漪振幅。振幅用活动单位数表示。可动画
<Spaceripple>.wavelength <Spaceripple>.Wave_Length	Float	5.0	以活动单位数设定每个波的长度。可动画
<Spaceripple>.phase	Float	0.0	设定波纹距离波纹中心的偏移相位，整数值无效，只有小数值才有效。在不同的关键帧变化该数值会使涟漪看起来像是在空间中传播。可动画
<Spaceripple>.decay	Float	0.0	当其设定为 0.0 时，涟漪在整个世界空间中有着相同的一个或多个振幅。增加衰退值会导致振幅从涟漪扭曲对象的所在位置开始随距离的增加而减弱。可动画
<Spaceripple>.circles	Integer	10	设定涟漪图标中的圆圈数目
<Spaceripple>.segments	Integer	16	设定涟漪图标中的分段（扇形）数目
<Spaceripple>.divisions	Integer	4	调整涟漪图标的大小，不会像缩放操作那样改变涟漪效果

### 相关方法

bindSpaceWarp <node> <spaceRipple\_Node>

### 相关的 Binding Modifier 和属性

#### RippleBinding

本 Modifier 由方法 bindSpaceWarp() 自动创建，而不能由 MAXScript 直接创建。下面为与本方法相关的属性。

<RippleBinding>.Flexibility: Float, 默认值: 1.0

### 8.10.7 SpaceWave: SpacewarpObject (波浪空间扭曲)

#### 构造函数

SpaceWave ...

#### 属性

属性名称	数据类型	默认值	说明
<Spacewave>.amplitude1 <Spacewave>.Amplitude_1	Float	5.0	设置沿波浪扭曲对象 Local 坐标系 X 轴的波浪振幅。可动画
<Spacewave>.amplitude2 <Spacewave>.Amplitude_2	Float	5.0	设置沿波浪扭曲对象 Local 坐标系 Y 轴的波浪振幅。可动画
<Spacewave>.wavelength <Spacewave>.Wave_Length	Float	25.0	以活动单位数设置每个波浪沿其 Local 坐标系 Y 轴的长度
<Spacewave>.phase	Float	0.0	设定从其在波浪对象中央的原点开始偏移波浪的相位。整数值无效，只有小数值才有效。设置该参数的动画会使波浪看起来像是在空间中传播。可动画
<Spacewave>.decay	Float	0.0	当其设置为 0.0 时，波浪在整个世界空间中有相同的一个或多个振幅。增加衰退值会导致振幅从波浪扭曲对象的所在位置开始随距离的增加而减弱
<Spacewave>.circles	Integer	4	设置沿波浪对象的 Local 坐标系 X 方向的分段数
<Spacewave>.segments	Integer	20	设置沿波浪对象的 Local 坐标系 Y 方向的分段数
<Spacewave>.divisions	Integer	10	在不改变波浪效果(缩放则会)的情况下调整波浪图标的大小

#### 相关方法

bindSpaceWarp <node> <spaceWave\_Node>

#### 相关的 Binding Modifier 和属性

#### WaveBinding

本 Modifier 由方法 bindSpaceWarp() 自动创建，而不能由 MAXScript 直接创建。下面为与本方法相关的属性：

<WaveBinding>.Flexibility: Float, 默认值: 1.0

### 8.10.8 Gravity: SpacewarpObject (重力空间扭曲)

#### 构造函数

Gravity ...

#### 属性

属性名称	数据类型	默认值	说明
<Gravity>.strength	Float	1.0	指定重力空间扭曲的强度。如果是负值，重力空间扭曲是反方向的，即粒子系统朝向空间扭曲指定的方向；如果该值为 0，重力空间扭曲不起作用。可动画
<Gravity>.decay	Float	1.0	设置重力空间扭曲作用范围的衰减量。默认为 0，表示在整个场景空间范围内，重力空间扭曲的作用强度一致。可动画
<Gravity>.gravitytype	Integer	0	设置重力空间扭曲的作用方式： 0: Planar (重力效果垂直于贯穿场景的重力扭曲对象所在的平面) 1: Spherical (重力效果为球形，以重力扭曲对象为中心)
<Gravity>.showRange	Boolean	False	当设置为 True 且属性.decay 的数值大于 0 时，在场景中的重力空间扭曲将显示一个作用范围界限，该界限标定了重力强度减少为一半时的范围
<Gravity>.iconsize	Float	10.0	设置重力空间扭曲的图标显示尺寸，该尺寸只影响重力空间扭曲对象在视图中的显示，不会影响其实际作用效果。可动画
<Gravity>.hoopson	Boolean	True	当设置为 True 且属性.decay 的数值大于 0 时，在场景中的重力空间扭曲将显示一个作用范围界限，该界限标定了重力强度减少为一半时的范围。对于平面 (Planar) 重力空间扭曲，将显示两个平面；对于球体 (Spherical) 重力空间扭曲，将显示一个双箍球

#### 相关方法

bindSpaceWarp <node> <gravity\_Node>

#### 相关的 Binding Modifier

##### GravityBinding

本 Modifier 由方法 bindSpaceWarp() 自动创建，而不能由 MAXScript 直接创建。本 Modifier 没有相关的属性。

### 8.10.9 Motor: SpacewarpObject (马达空间扭曲)

#### 构造函数

Motor ...

## 属性

属性名称	数据类型	默认值	说明
<Motor>.On_Time	Time	0f	设置动力空间扭曲的开始作用时间
<Motor>.Off_Time	Time	30f	设置动力空间扭曲的结束作用时间
<Motor>.Basic_Torque	Float	1.0	设置空间扭曲施加的力的量。可动画
<Motor>.units	Integer	0	设置动力空间扭曲的单位: 0: Newton-meters (牛顿·米) 1: Pound-feet (磅·英尺) 2: Pound-inches (磅·英寸)
<Motor>.Feedback_On	Integer	0	当设置为 On 时, 力会根据受影响对象相对于指定目标速度的速度而变化。否则, 不管受影响对象的速度如何, 力保持不变。0: Off; 1: On
<Motor>.Reversible	Integer	0	当设置为 On 时, 如果对象的运动速度超过了指定的目标速度, 动力将转换方向。0: Off; 1: On
<Motor>.Target_Revs	Float	100.0	指定反馈生效前的最大转数。以每帧经过的单位数来指定速度。可动画
<Motor>.Revs_Units	Integer	1	指定目标转速的度量单位: 0: RPH (转/小时) 1: RPM (转/分钟) 2: RPS (转/秒)
<Motor>.Control_Gain	Float	50.0	指定以何种速度调整力以达到目标速度。如果设置为100%, 校正会立即进行; 如果设置较低的值, 发生的响应会越来越慢、越来越散。仅在打开启用反馈选项时可用。可动画
<Motor>.Enable_Variation	Integer	0	设置是否使周期变化设置有效: 0: Off; 1: On
<Motor>.Variation_Period_1	Time	0.625f	设置噪波变化完成整个循环所需的时间
<Motor>.Amplitude_1	Float	100.0	设置第一个变化周期的变化强度(百分比)。可动画
<Motor>.Variation_Phase_1	Float	0.0	设置第一个偏移变化模式。可动画, Angle
<Motor>.Variation_Period_2	Time	0.625f	设置第二个变化周期的持续时间
<Motor>.Amplitude_2	Float	100.0	设置二阶波的变化强度。可动画
<Motor>.Variation_Phase_2	Float	0.0	设置偏移二阶波的变化模式。可动画, Angle
<Motor>.Range_Enable	Integer	0	当设置为 On 时, 会将效果范围限制为一个球体, 其显示为一个带有 3 个环箍的球体。当粒子靠近球体边界时, 效果会加速衰退。0: Off; 1: On
<Motor>.Range_Value	Float	1000.0	以单位数指定效果范围的半径。可动画
<Motor>.Icon_Size	Float	0.0	设置马达图标的大小。该设置仅用于显示目的, 而不会改变马达效果

## 相关方法

`bindSpaceWarp <node> <motor_Node>`

### 相关的 Binding Modifier

#### MotorMod

本 Modifier 由方法 `bindSpaceWarp()` 自动创建，而不能由 MAXScript 直接创建。本 Modifier 没有相关的属性。

## 8.10.10 PBomb: SpacewarpObject (粒子爆炸空间扭曲)

### 构造函数

`PBomb ...`

### 属性

属性名称	数据类型	默认值	说明
<code>&lt;PBomb&gt;.symmetry</code>	Integer	0	指定爆炸效果的形状或图案。 0: Spherical (爆炸的作用范围为球体，粒子沿球体径向方向飞出) 1: Cylindrical (爆炸的作用范围为柱体，粒子沿柱面法线方向飞出) 2: Planar (爆炸的作用范围为平面，粒子沿平面的正反方向飞出)
<code>&lt;PBomb&gt;.chaos</code>	Float	10.0	指定粒子爆炸后飞出的混乱程度。百分数
<code>&lt;PBomb&gt;.Start_Time</code>	Time	30f	指定粒子开始爆炸的时间
<code>&lt;PBomb&gt;.Lasts_For</code>	Time	1f	指定粒子爆炸后飞出的持续时间，为了获得真实的爆炸效果，一般将该数值设定为 0~3
<code>&lt;PBomb&gt;.strength</code>	Float	1.0	设置沿爆炸向量的速率变化，用每帧的单位数表示。增加强度会增加粒子从爆炸图标向外爆炸的速度
<code>&lt;PBomb&gt;.Decay_Type</code>	Integer	1	设置爆炸衰减的方式： 0: Unlimited Range (爆炸图标的效果能到达整个场景中所有绑定的粒子) 1: Linear (冲击力从满强度设置到指定的范围设置处线性地衰减至 0) 2: Exponential (冲击力从满强度设置到指定的范围设置处按指数规律衰减至 0)
<code>&lt;PBomb&gt;.range</code>	Float	1000.0	指定显示一个线框球体来表示粒子爆炸影响的体积。可动画
<code>&lt;PBomb&gt;.Icon_Size</code>	Float	0.0	改变粒子爆炸图标的整体大小

### 相关方法

`bindSpaceWarp <node> <pbomb_Node>`

### 相关的Binding Modifier

#### PBombMod

本Modifier由方法bindSpaceWarp()自动创建，而不能由MAXScript直接创建。本Modifier没有相关的属性。

### 8.10.11 PushSpaceWarp: SpacewarpObject(推力空间扭曲)

#### 构造函数

PushSpaceWarp ...

#### 属性

属性名称	数据类型	默认值	说明
<PushSpaceWarp>.On_Time	Time	0f	设置推力空间扭曲的开始作用的时间
<PushSpaceWarp>.Off_Time	Time	30f	设置推力空间扭曲的结束作用的时间
<PushSpaceWarp>.Basic_Force	Float	1.0	设置空间扭曲施加推力的大小。可动画
<PushSpaceWarp>.units	Integer	0	指定基本力使用的力的单位： 0: Newtons(牛顿); 1: Pounds(磅)
<PushSpaceWarp>.Feedback_On	Integer	0	当设置为On时，扭曲力的大小由粒子系统运动速度与目标转速的接近程度决定；否则，扭曲力的大小恒定，不受粒子系统运动速度的影响： 0: Off; 1: On
<PushSpaceWarp>.Reversible	Integer	0	当设置为On时，如果对象的运动速度超过了指定的目标速度，动力将转换方向：0: Off; 1: On
<PushSpaceWarp>.Target_Speed	Float	100.0	以每帧的单位数指定反馈生效前的最大速度。可动画
<PushSpaceWarp>.Control_Gain	Float	50.0	指定以何种速度调整力以达到目标速度。如果设置为100%，校正会立即进行。如果设置较低的值，发生的响应会越来越慢、越来越散。可动画
<PushSpaceWarp>.Enable_Variation	Integer	0	设置是否使周期变化设置有效：0: Off; 1: On
<PushSpaceWarp>.Variation_Period_1	Time	0.625f	设置噪波变化完成整个循环所需的时间
<PushSpaceWarp>.Amplitude_1	Float	100.0	设置第一个变化周期的变化强度(百分比)。可动画
<PushSpaceWarp>.Variation_Phase_1	Float	0.0	设置第一个偏移变化模式。可动画，Angle
<PushSpaceWarp>.Variation_Period_2	Time	0.625f	设置第二个变化周期的持续时间
<PushSpaceWarp>.Amplitude_2	Float	100.0	设置二阶波的变化强度。可动画
<PushSpaceWarp>.Variation_Phase_2	Float	0.0	设置偏移二阶波的变化模式。可动画，Angle

(续表)

属性名称	数据类型	默认值	说明
<PushSpaceWarp>.Range_Enable	Integer	0	当设置为 On 时，会将效果范围限制为一个球体，其显示为一个带有 3 个环箍的球体。当粒子靠近球体边界时，效果会加速衰退。0: Off; 1: On
<PushSpaceWarp>.Range_Value	Float	1000.0	以单位数指定效果范围的半径。可动画
<PushSpaceWarp>.Icon_Size	Float	0.0	设置推力图标的大小

### 相关方法

bindSpaceWarp <node> <pushSpaceWarp\_Node>

### 相关的 Binding Modifier

#### PushMod

本 Modifier 由方法 bindSpaceWarp() 自动创建，而不能由 MAXScript 直接创建。本 Modifier 没有相关的属性。

## 8.10.12 Wind: SpacewarpObject (风力空间扭曲)

### 构造函数

Wind ...

### 属性

属性名称	数据类型	默认值	说明
<Wind>.strength	Float	1.0	指定风力空间扭曲的强度。如果是负值，风力空间扭曲是反方向的，即粒子系统朝向空间扭曲指定的方向；如果该值为 0，风力空间扭曲不起作用。可动画
<Wind>.decay	Float	1.0	设置风力空间扭曲作用范围的衰减量。默认为 0，表示在整个场景空间范围内，风力空间扭曲的作用强度一致。可动画，Angle
<Wind>.windtype	Integer	0	设置风力空间扭曲的作用方式：0: Planar; 1: Spherical
<Wind>.turbulence	Float	1.0	设定风的紊乱度，用于模拟粒子被风吹得随机舞动的效果。可动画
<Wind>.frequency	Float	1.0	指定混乱度的变化周期，该数值的影响效果比较小，除非当前粒子系统包含大量的粒子。可动画
<Wind>.windScale	Float	0.0	放缩紊乱的作用效果，如果该值设定得很小，紊乱运动更为平稳规则；增大该值的设定，紊乱运动更为剧烈无序。可动画
<Wind>.showRange	Boolean	False	当设置为 True 且属性.decay 的数值大于 0 时，在场景中的风力空间扭曲将显示一个作用范围界限，该界限标定了重力强度减少为一半时的范围
<Wind>.iconsize	Float	10.0	设置风力空间扭曲的图标显示尺寸，该尺寸只影响风力空间扭曲对象在视图中的显示，不会影响其实际的作用效果。可动画

(续表)

属性名称	数据类型	默认值	说明
<Wind>.hoopson	Boolean	False	当设置为True且属性.decay的数值大于0时，在场景中的风力空间扭曲将显示一个作用范围界限，该界限标定了风力强度减少为一半时的范围。对于平面(Planar)风力空间扭曲，将显示两个平面；对于球体(Spherical)风力空间扭曲，将显示一个双箍球

**相关方法**

bindSpaceWarp &lt;node&gt; &lt;wind\_Node&gt;

**相关的 Binding Modifier****WindBinding**

本Modifier由方法bindSpaceWarp()自动创建，而不能由MAXScript直接创建。本Modifier没有相关的属性。

**8.10.13 SpaceBend: SpacewarpObject(弯曲修改器)****构造函数**

SpaceBend ...

**属性**

属性名称	数据类型	默认值	说明
<SpaceBend>.length	Float	0.0	指定空间扭曲对象的长度。可动画
<SpaceBend>.width	Float	0.0	指定空间扭曲对象的宽度。可动画
<SpaceBend>.height	Float	0.0	指定空间扭曲对象的高度。可动画
<SpaceBend>.decay	Float	0.0	设置弯曲空间扭曲作用范围的衰减量。当其设置为0时没有衰退，不论到对象的距离是多少，空间扭曲都会影响它的绑定对象。增加衰退时，作用在绑定对象上的效果会成指数衰减
<SpaceBend>.angle	Float	0.0	指定弯曲的角度。可动画
<SpaceBend>.direction	Float	0.0	指定弯曲相对于水平面的方向。可动画
<SpaceBend>.Lower_Limit <SpaceBend>.Lowerlimit	Float	0.0	以世界单位设置上部边界，此边界位于弯曲中心点上方，超出此边界弯曲不再影响几何体。可动画
<SpaceBend>.Upper_Limit <SpaceBend>.Upperlimit	Float	0.0	以世界单位设置下部边界，此边界位于弯曲中心点下方，超出此边界弯曲不再影响几何体。可动画

**相关方法**

bindSpaceWarp &lt;node&gt; &lt;spaceBend\_Node&gt;

### 相关的 Binding Modifier

#### SimpleOSMToWSMMMod

本 Modifier 由方法 bindSpaceWarp() 自动创建，而不能由 MAXScript 直接创建。本 Modifier 没有相关的属性。

### 8.10.14 SpaceNoise: SpacewarpObject (噪波修改器)

#### 构造函数

SpaceNoise ...

#### 属性

属性名称	数据类型	默认值	说明
<SpaceNoise>.length	Float	0.0	指定空间扭曲对象的长度。可动画
<SpaceNoise>.width	Float	0.0	指定空间扭曲对象的宽度。可动画
<SpaceNoise>.height	Float	0.0	指定空间扭曲对象的高度。可动画
<SpaceNoise>.decay	Float	0.0	设置噪波空间扭曲作用范围的衰减量。默认为 0，表示在整个场景空间范围内，噪波空间扭曲的作用强度一致。可动画
<SpaceNoise>.seed	Integer	0	指定噪波空间扭曲的随机数，当多个对象绑定到噪波空间扭曲之上时，可以使用相同的设置产生不同的噪波效果。可动画
<SpaceNoise>.scale	Float	100.0	指定噪波效果的尺寸（不是长度），高的数值产生平滑的噪波，低的数值产生锯齿状的噪波。可动画
<SpaceNoise>.fractal	Integer	0	当设置为 On 时，对噪波进行分型处理。可动画 0: Off; 1: On
<SpaceNoise>.Rough	Float	0.0	指定分型面的大小，高的数值比低的数值产生的分型效果更为粗糙。可动画
<SpaceNoise>.iterations	Float	6.0	指定重复进行分型处理的次数，较低的设置可以产生更为平滑的效果，设定为 1.0 等同于关闭分型处理。可动画
<SpaceNoise>.phase	Integer	0	指定噪波动画的初始相位。可动画
<SpaceNoise>.strength	Point3	[0,0,0]	指定三个轴向的噪波作用强度。可动画

**注意** 因为属性.scale 同时也是一个 Node 级属性，为了存取 SpaceNoise 对象的.scale 属性，应该用下面方法：

```
sn=spaceNoise()
sn.baseObject.scale=150
```

设置动画后的.noise 属性不能在 3ds max 的 MAXScript 里进行存取。

### 相关方法

`bindSpaceWarp <node> <spaceNoise_Node>`

### 相关的 Binding Modifier

`SimpleOSMToWSMMod`

本 Modifier 由方法 `bindSpaceWarp()` 自动创建，而不能由 MAXScript 直接创建。本 Modifier 没有相关的属性。

## 8.10.15 SpaceSkew: SpacewarpObject (倾斜修改器)

### 构造函数

`SpaceSkew ...`

### 属性

属性名称	数据类型	默认值	说明
<code>&lt;SpaceSkew&gt;.length</code>	Float	0.0	指定空间扭曲对象的长度。可动画
<code>&lt;SpaceSkew&gt;.width</code>	Float	0.0	指定空间扭曲对象的宽度。可动画
<code>&lt;SpaceSkew&gt;.height</code>	Float	0.0	指定空间扭曲对象的高度。可动画
<code>&lt;SpaceSkew&gt;.decay</code>	Float	0.0	设置倾斜空间扭曲作用范围的衰减量。默认为 0，表示在整个场景空间范围内，倾斜空间扭曲的作用强度一致。可动画
<code>&lt;SpaceSkew&gt;.amount</code>	Float	0.0	设置倾斜的角度。可动画
<code>&lt;SpaceSkew&gt;.direction</code>	Float	0.0	设置倾斜相对于水平面的方向。可动画
<code>&lt;SpaceSkew&gt;.Lower_Limit</code>	Float	0.0	指定倾斜操作的下限范围。可动画
<code>&lt;SpaceSkew&gt;.Upper_Limit</code>	Float	0.0	指定倾斜操作的上限范围。可动画

### 相关方法

`bindSpaceWarp <node> <spaceSkew_Node>`

### 相关的 Binding Modifier

`SimpleOSMToWSMMod`

本 Modifier 由方法 `bindSpaceWarp()` 自动创建，而不能由 MAXScript 直接创建。本 Modifier 没有相关的属性。

## 8.10.16 SpaceStretch: SpacewarpObject (挤出修改器)

### 构造函数

`SpaceStretch ...`

## 属性

属性名称	数据类型	默认值	说明
<SpaceStretch>.length	Float	0.0	指定空间扭曲对象的长度。可动画
<SpaceStretch>.width	Float	0.0	指定空间扭曲对象的宽度。可动画
<SpaceStretch>.height	Float	0.0	指定空间扭曲对象的高度。可动画
<SpaceStretch>.decay	Float	0.0	设置延展空间扭曲作用范围的衰减量。默认为 0，表示在整个场景空间范围内，延展空间扭曲的作用强度一致。可动画
<SpaceStretch>.Stretch	Float	0.0	设置延展放缩的强度。延展放缩强度被指定到选定的延展轴上；延展放缩强度的反值被指定到选定的次轴上；正值表示 Stretch+1，例如，设置为 1.5 表示 $1.5+1=2.5$ （即延展 250%）；负值表示 -1/ (Stretch-1)，例如，设置为 -1.5 表示 $-1/(-1.5-1)=0.4$ （即延展 40%）。可动画
<SpaceStretch>.Amplify	Float	0.0	设置在次轴向上延展缩放强度的倍增量。设置为正值，倍增延展放缩强度；设置为负值，减小延展缩放强度。可动画
<SpaceStretch>.from	Float	0.0	设置延展操作的下限。可动画
<SpaceStretch>.to	Float	0.0	设置延展操作的上限。可动画

## 相关方法

bindSpaceWarp <node> <spaceStretch\_Node>

### 相关的 Binding Modifier

SimpleOSMToWSMMMod

本 Modifier 由方法 bindSpaceWarp() 自动创建，而不能由 MAXScript 直接创建。本 Modifier 没有相关的属性。

## 8.10.17 SpaceTaper: SpacewarpObject (锥化修改器)

### 构造函数

SpaceTaper ...

## 属性

属性名称	数据类型	默认值	说明
<SpaceTaper>.length	Float	0.0	指定空间扭曲对象的长度。可动画
<SpaceTaper>.width	Float	0.0	指定空间扭曲对象的宽度。可动画
<SpaceTaper>.height	Float	0.0	指定空间扭曲对象的高度。可动画
<SpaceTaper>.decay	Float	0.0	设置锥化空间扭曲作用范围的衰减量。默认为 0，表示在整个场景空间范围内，锥化空间扭曲的作用强度一致。可动画
<SpaceTaper>.amount	Float	0.0	指定端面的缩放程度，这是一个相对值，最大可设定为 10。可动画

(续表)

属性名称	数据类型	默认值	说明
<SpaceTaper>.curvature	Float	0.0	为锥化线框的侧面指定一个曲率，该数值影响锥化变形的最终形态，正值产生向外凸出的侧面；负值产生向内凹进的侧面；如果设定为0侧面是平直的。可动画
<SpaceTaper>.symmetry	Integer	0	当设置为On时，锥化的效果是对称的： 0: Off; 1: On
<SpaceTaper>.Lower_Limit	Float	0.0	指定锥化操作的下限。可动画
<SpaceTaper>.Upper_Limit	Float	0.0	指定锥化操作的上限。可动画

### 相关方法

bindSpaceWarp <node> <spaceTaper\_Node>

#### 相关的 Binding Modifier

SimpleOSMToWSMMod

本 Modifier 由方法 bindSpaceWarp() 自动创建，而不能由 MAXScript 直接创建。本 Modifier 没有相关的属性。

### 8.10.18 SpaceTwist: SpacewarpObject (扭曲修改器)

#### 构造函数

SpaceTwist...

#### 属性

属性名称	数据类型	默认值	说明
<SpaceTwist>.length	Float	0.0	指定空间扭曲对象的长度。可动画
<SpaceTwist>.width	Float	0.0	指定空间扭曲对象的宽度。可动画
<SpaceTwist>.height	Float	0.0	指定空间扭曲对象的高度。可动画
<SpaceTwist>.decay	Float	0.0	设置扭曲空间扭曲作用范围的衰减量。默认为0，表示在整个场景空间范围内，扭曲空间扭曲的作用强度一致。可动画
<SpaceTwist>.angle	Float	0.0	指定扭曲变形的角度。可动画
<SpaceTwist>.bias	Float	0.0	指定扭曲变形效果在对象表面上向上的偏向数值，该值如果是负值，扭曲效果靠近变换线框的中心；如果是正值，扭曲效果沿扭曲轴向远离中心；如果该值是0，扭曲效果在扭曲轴向上是一致的。可动画
<SpaceTwist>.Lower_Limit	Float	0.0	指定扭曲操作的下限。可动画
<SpaceTwist>.Upper_Limit	Float	0.0	指定扭曲操作的上限。可动画

## 相关方法

`bindSpaceWarp <node> <spaceTwist_Node>`

### 相关的 Binding Modifier

`SimpleOSMToWSMMod`

本 Modifier 由方法 `bindSpaceWarp()` 自动创建，而不能由 MAXScript 直接创建。本 Modifier 没有相关的属性。

## 8.10.19 PDynaFlect: SpacewarpObject (动力学导向板空间扭曲)

### 构造函数

`PDynaFlect ...`

### 属性

属性名称	数据类型	默认值	说明
<code>&lt;PDynaFlect&gt;.Time_On</code>	Time	0f	指定导向器作用的开始时间。可动画
<code>&lt;PDynaFlect&gt;.Time_Off</code>	Time	100f	指定导向器作用的结束时间。可动画
<code>&lt;PDynaFlect&gt;.Reflects</code>	Float	10000.0	指定粒子受动力学导向板反作用的百分比。可动画
<code>&lt;PDynaFlect&gt;.bounce</code>	Float	1.0	指定粒子系统撞击动力学导向板后的反弹速度。可动画
<code>&lt;PDynaFlect&gt;.Bounce_Variation</code>	Float	0.0	指定反弹设置的变化量。可动画
<code>&lt;PDynaFlect&gt;.chaos</code>	Float	0.0	随机产生方向混乱的反弹变化。可动画
<code>&lt;PDynaFlect&gt;.Velocity_Inheritance</code>	Float	1.0	指定动力学导向板的运动如何影响到与其绑定的粒子系统。可动画
<code>&lt;PDynaFlect&gt;.Particle_Mass</code>	Float	1.0	指定基于选定单位的质量。可动画
<code>&lt;PDynaFlect&gt;.Particle_Mass_Units</code>	Integer	0	指定粒子质量单位。可动画 0: Gram (克); 1: Kilogram (千克); 2: Pound (磅)
<code>&lt;PDynaFlect&gt;.width</code>	Float	0.0	指定导向器在场景中显示的宽度。该图标显示指定不会影响导向器的实际作用效果。可动画
<code>&lt;PDynaFlect&gt;.height</code>	Float	0.0	指定导向器在场景中显示的高度。该图标显示指定不会影响导向器的实际作用效果。可动画

### 相关方法

`bindSpaceWarp <node> <PDynaFlect_Node>`

### 相关的 Binding Modifier

`PDynaFlectMod`

本 Modifier 由方法 `bindSpaceWarp()` 自动创建，而不能由 MAXScript 直接创建。本 Modifier 没有相关的属性。

## 8.10.20 SDynaFlect: SpacewarpObject (动力学导向球空间扭曲)

## 构造函数

SDynaFlect ...

## 属性

属性名称	数据类型	默认值	说明
<SDynaFlect>.‘time on’ <SDynaFlect>.time_on	Integer	0	指定导向器作用的开始时间。可动画
<SDynaFlect>.‘time off’ <SDynaFlect>.time_off	Integer	16000	指定导向器作用的结束时间。可动画
<SDynaFlect>.affects <SDynaFlect>.reflects	Float	10000.0	指定粒子受动力学导向球反作用的百分比。可动画，百分数
<SDynaFlect>.bouncevar <SDynaFlect>.bounce_variation	Float	0.0	指定粒子系统撞击动力学导向球后的反弹速度。可动画，百分数
<SDynaFlect>.inheritVelocity <SDynaFlect>.velocity_inheritance	Float	1.0	指定动力学导向球的运动如何影响到与其绑定的粒子系统。可动画，百分数
<SDynaFlect>.mass <SDynaFlect>.particle_mass	Float	1.0	指定基于选定单位的质量。可动画
<SDynaFlect>.‘massunits’ <SDynaFlect>.particle_mass_units	Integer	0	指定粒子质量单位。可动画 0: Gram (克) 1: Kilogram (千克) 2: Pound (磅)
<SDynaFlect>.‘force in x’	Float	0.0	导向器力沿 X 轴的分量。可动画
<SDynaFlect>.‘force in y’	Float	0.0	导向器力沿 Y 轴的分量。可动画
<SDynaFlect>.‘force in z’	Float	0.0	导向器力沿 Z 轴的分量。可动画
<SDynaFlect>.‘apply at x’	Float	0.0	导向器力作用在 X 轴上的位置。可动画
<SDynaFlect>.‘apply at y’	Float	0.0	导向器力作用在 Y 轴上的位置。可动画
<SDynaFlect>.‘apply at z’	Float	0.0	导向器力作用在 Z 轴上的位置。可动画
<SDynaFlect>.number	Integer	0	设置粒子数量。可动画
<SDynaFlect>.friction	Float	0.0	指定粒子系统沿导向器运动时，速度受摩擦力减慢的效果。如果设为 0，粒子速度不减慢；如果设为 0.5，粒子速度减慢一半；设为 1 时，粒子在打击对象表面时静止。可动画
<SDynaFlect>.collider	undefined	Undefined	指定碰撞后使粒子发生偏转的 Node 对象
<SDynaFlect>.bounce	Float	1.0	指定粒子系统撞击动力学导向球后的反弹速度。可动画
<SDynaFlect>.chaos	Float	0.0	随机产生方向混乱的反弹变化。可动画
<SDynaFlect>.radius	Float	0.0	设置动力学导向球在视图中的显示尺寸。可动画

## 相关方法

`bindSpaceWarp <node> <SDynaFlect_Node>`

### 相关的 Binding Modifier

`SDynaFlectMod`

本 Modifier 由方法 `bindSpaceWarp()` 自动创建，而不能由 MAXScript 直接创建。本 Modifier 没有相关的属性。

## 8.10.21 UDynaFlect: SpacewarpObject (通用动力学导向器空间扭曲)

### 构造函数

`UDynaFlect ...`

### 属性

属性名称	数据类型	默认值	说明
<code>&lt;UDynaFlect&gt;.‘time on’</code> <code>&lt;UDynaFlect&gt;.time_on</code>	Integer	0	指定导向器作用的开始时间。可动画
<code>&lt;UDynaFlect&gt;.‘time off’</code> <code>&lt;UDynaFlect&gt;.time_off</code>	Integer	1600000	指定导向器作用的结束时间。可动画
<code>&lt;UDynaFlect&gt;.affects</code> <code>&lt;UDynaFlect&gt;.reflects</code>	Float	10000.0	指定粒子受通用动力学导向器反作用的百分比。可动画，百分数
<code>&lt;UDynaFlect&gt;.bounce</code>	Float	1.0	指定粒子系统撞击通用动力学导向器后的反弹速度与撞击前速度的比例。可动画
<code>&lt;UDynaFlect&gt;.bouncevar</code> <code>&lt;UDynaFlect&gt;.bounce_variation</code>	Float	10000.0	指定粒子系统撞击通用动力学导向器后的反弹速度的变化量。可动画，百分数
<code>&lt;UDynaFlect&gt;.chaos</code>	Float	0.0	随机产生方向混乱的反弹变化。可动画
<code>&lt;UDynaFlect&gt;.Friction</code>	Float	0.0	指定粒子系统沿通用导向器运动时，速度受摩擦力减慢的效果。如果设定为 0，粒子速度不减慢；如果设定为 0.5，粒子速度减慢一半；设定为 1 时，粒子在打击对象表面时静止。可动画
<code>&lt;UDynaFlect&gt;.inheritVelocity</code> <code>&lt;UDynaFlect&gt;.velocity_inheritance</code>	Float	1.0	指定通用动力学导向器的运动如何影响到与其绑定的粒子系统。可动画
<code>&lt;UDynaFlect&gt;.mass</code> <code>&lt;UDynaFlect&gt;.particle_mass</code>	Float	1.0	指定基于选定单位的粒子质量。可动画
<code>&lt;UDynaFlect&gt;.‘mass units’</code> <code>&lt;UDynaFlect&gt;.particle_mass_units</code>	Integer	0	指定粒子质量单位。可动画 0: Gram (克) 1: Kilogram (千克) 2: Pound (磅)

(续表)

属性名称	数据类型	默认值	说明
<UDynaFlect>.‘force in x’	Float	0.0	导向器力沿 X 轴的分量。可动画
<UDynaFlect>.‘force in y’	Float	0.0	导向器力沿 Y 轴的分量。可动画
<UDynaFlect>.‘force in z’	Float	0.0	导向器力沿 Z 轴的分量。可动画
<UDynaFlect>.‘apply at x’	Float	0.0	导向器力作用在 X 轴上的位置。可动画
<UDynaFlect>.‘apply at y’	Float	0.0	导向器力作用在 Y 轴上的位置。可动画
<UDynaFlect>.‘apply at z’	Float	0.0	导向器力作用在 Z 轴上的位置。可动画
<UDynaFlect>.number	Integer	20	设置粒子数量。可动画
<UDynaFlect>.collider	undefined	undefined	指定碰撞后使粒子发生偏转的 Node 对象
<UDynaFlect>.radius <UDynaFlect>.icon_size	Float	0.0	设置通用动力学导向器在视图中的显示尺寸。可动画

### 相关方法

bindSpaceWarp <node> <UDynaFlect\_Node>

### 相关的 Binding Modifier

#### UDynaFlectMod

本 Modifier 由方法 bindSpaceWarp() 自动创建，而不能由 MAXScript 直接创建。本 Modifier 没有相关的属性。

### 8.10.22 Deflector: SpacewarpObject (导向器空间扭曲)

#### 构造函数

Deflector ...

#### 属性

属性名称	数据类型	默认值	说明
<Deflector>.bounce	Float	1.0	指定粒子系统撞击导向器后的反弹速度。可动画
<Deflector>.width	Float	10.0	指定导向器的宽度。可动画
<Deflector>.length	Float	10.0	指定导向器的长度。可动画
<Deflector>.variation	Float	0.0	指定反弹设置的变化量。可动画
<Deflector>.chaos	Float	0.0	随机产生方向混乱的反弹变化。可动画
<Deflector>.friction	Float	0.0	指定粒子系统沿导向器运动时，速度受摩擦力减慢的效果。如果设定为 0，粒子速度不减慢；如果设定为 0.5，粒子速度减慢一半；设定为 1 时，粒子在打击对象表面时静止。可动画
<Deflector>.inheritVelocity	Float	0.0	当该值大于 0 时，导向器的运动会和其他设置一样对粒子产生影响。例如，如果想让一个经过粒子阵列的动画化导向器影响这些粒子，加大该值。可动画
<Deflector>.quality	Integer	20	指定一个运动的导向器与静止粒子系统撞击时的计算量，较高的数值可以得到精细的碰撞效果。可动画

## 相关方法

`bindSpaceWarp <particlesys_Node> <deflector_Node>`

### 相关的 Binding Modifier

#### DeflectorBinding

本 Modifier 由方法 `bindSpaceWarp()` 自动创建，而不能由 MAXScript 直接创建。本 Modifier 没有相关的属性。

## 8.10.23 Path\_Follow: SpacewarpObject (路径跟随空间扭曲)

### 构造函数

`Path_Follow ...`

### 属性

属性名称	数据类型	默认值	说明
<code>&lt;Path_Follow&gt;.Range_Limited</code>	Integer	1	当设置为 Off 时，路径跟随空间扭曲的作用范围受属性. <code>range_value</code> 设置的影响；否则，路径跟随空间扭曲的作用范围是无限的
<code>&lt;Path_Follow&gt;.Range_Value</code>	Float	100.0	设置路径跟随空间扭曲的作用范围，即粒子系统与路径之间的距离。可动画
<code>&lt;Path_Follow&gt;.Spline_Follow_Type</code>	Integer	1	设置粒子运动路径方式： 0: Along Offset Splines (粒子系统与路径之间的平移距离对粒子的跟随运动产生影响) 1: Along Parallel Splines (粒子系统与路径之间的平移距离对粒子的跟随运动不产生影响，但路径的方向会影响运动过程)
<code>&lt;Path_Follow&gt;.Constant_Speed</code>	Integer	0	当设置为 On 时，粒子沿路径匀速运动：0: Off; 1: On
<code>&lt;Path_Follow&gt;.Tangent_Chaos</code>	Float	0.0	指定粒子在运动过程中，朝向或偏移路径的程度。具体沿路径运动的方式由属性. <code>Tangent_Dir</code> 指定。百分数
<code>&lt;Path_Follow&gt;.Tang_Chaos_Var</code>	Float	0.0	指定属性. <code>Tangent_Chaos</code> 的变化量
<code>&lt;Path_Follow&gt;.Tangent_Dir</code>	Integer	0	设置粒子沿路径运动的方式： 0: Converge (粒子一边沿路径运动一边向路径靠近) 1: Diverge (粒子一边沿路径运动一边从路径脱离) 2: Both (一部分粒子进行聚合运动，一部分粒子进行分散运动)
<code>&lt;Path_Follow&gt;.Spiral_Chaos</code>	Float	0.0	指定粒子在沿路径运动的过程中，围绕路径旋转的圈数
<code>&lt;Path_Follow&gt;.Spiral_Chaos_Var</code>	Float	0.0	指定不同粒子旋涡流动的变化量

(续表)

属性名称	数据类型	默认值	说明
<Path_Follow>.Spiral_Dir	Integer	0	设置不同粒子旋涡流动的方式: 0: Clockwise (顺时针运动) 1: Counterclockwise (逆时针运动) 2: Bidirectional (围绕路径双向旋转)
<Path_Follow>.Start_Time	Time	0f	指定空间扭曲作用于粒子系统的开始帧
<Path_Follow>.Travel_Time	Time	30f	指定系统中每个粒子沿路径跟随所用的时间
<Path_Follow>.Travel_Var	Float	0.0	指定粒子跟随时间变化的百分比。百分数
<Path_Follow>.Stop_Time	Time	100f	指定空间扭曲作用于粒子系统的结束帧
<Path_Follow>.Icon_Size	Float	0.0	设置路径跟随空间扭曲的图标尺寸，该尺寸只影响路径跟随空间扭曲对象在视图中的显示，不会影响实际的作用效果

**注意** 目前还不能在 MAXScript 里选择 Path 对象。

### 相关方法

bindSpaceWarp <particlesys\_Node> <path\_Follow\_Node>

### 相关的 Binding Modifier

#### PathFollowMod

本 Modifier 由方法 bindSpaceWarp() 自动创建，而不能由 MAXScript 直接创建。本 Modifier 没有相关的属性。

## 8.10.24 P0mniFlect: SpacewarpObject (泛方向导向板空间扭曲)

### 构造函数

P0mniFlect ...

### 属性

属性名称	数据类型	默认值	说明
<POmniFlect>.deceleration	Float	1.0	设置粒子的减速度。可动画
<POmniFlect>.'decel var'	Float	0.0	设置.deceleration 属性的变化量。可动画，百分数
<POmniFlect>.friction	Float	0.0	指定粒子系统沿泛方向导向板运动时，速度受摩擦力减慢的效果。如果设定为 0，粒子速度不减慢；如果设定为 0.5，粒子速度减慢一半；设定为 1 时，粒子在打击对象表面时静止。可动画，百分数

(续表)

属性名称	数据类型	默认值	说明
<POmniFlect>.quality	Integer	20	指定一个运动的泛方向导向板与静止粒子系统撞击时的计算量，较高的数值可以得到精细的碰撞效果。可动画
<POmniFlect>.collider	undefined	undefined	指定碰撞后使粒子运动方向发生偏转的 Node 对象
<POmniFlect>.'time on' <POmniFlect>.time_on	Integer	0	指定导向器开始作用的时间。可动画
<POmniFlect>.'time off' <POmniFlect>.time_off	Integer	16000	指定导向器结束作用的时间。可动画
<POmniFlect>.affects <POmniFlect>.reflects	Float	10000.0	指定粒子受泛方向导向板反作用的百分比。可动画
<POmniFlect>.bounce	Float	1.0	指定粒子系统撞击泛方向导向板后的反弹速度。可动画
<POmniFlect>.bouncevar <POmniFlect>.bounce_variation	Float	0.0	指定反弹速度的变化量。可动画
<POmniFlect>.chaos	Float	0.0	随机产生方向混乱的反弹变化。可动画
<POmniFlect>.Refracts	Float	100.0	指定粒子受泛方向导向板折射作用的百分比。可动画
<POmniFlect>.'pass velocity' <POmniFlect>.pass_velocity	Float	1.0	指定粒子穿过导向平面后的速度。默认为 1.0，表示粒子保持原来的速度；如果设定为 0.5，粒子穿过导向平面后速度减小到原来的一半。可动画
<POmniFlect>.'passvel var' <POmniFlect>.Pass_Velocity_Variation	Float	0.0	指定粒子穿过导向平面后速度的变化量。可动画，百分数
<POmniFlect>.refraction <POmniFlect>.distortion	Float	50.0	指定折射的角度。如果设定为 0，表示不折射；如果设定为 100%，表示粒子系统沿导向平面平行运动。可动画，百分数
<POmniFlect>.'refraction var' <POmniFlect>.distortion_variation	Float	0.0	指定扭曲设置的变化量。可动画，百分数
<POmniFlect>.Diffusion	Float	0.0	通过随机改变粒子运行的角度，指定一个锥状散射的角度。可动画，百分数
<POmniFlect>.'diffusion var' <POmniFlect>.diffusion_variation	Float	0.0	指定散射设置的变化量。可动画，百分数
<POmniFlect>.inheritVelocity <POmniFlect>.velocity_inheritance	Float	1.0	指定导向器的运动如何影响到与其绑定的粒子系统。可动画，百分数

(续表)

属性名称	数据类型	默认值	说明
<POmniFlect>.spawn <POmniFlect>.spawn_percentage	Float	100.0	指定受粒子繁殖效果作用的百分比。可动画，百分数
<POmniFlect>.Pass_Velocity	Float	1.0	指定粒子穿过导向平面后的速度。默认为1.0，表示粒子保持原来的速度；如果设定为0.5，粒子穿过导向平面后速度减小到原来的一半。可动画，百分数
<POmniFlect>.Pass_Velocity_Variation	Float	0.0	指定粒子穿过速度的变化量。可动画，百分数
<POmniFlect>.width	Float	0.0	指定导向器在场景中显示的宽度。可动画
<POmniFlect>.height	Float	0.0	指定导向器在场景中显示的高度。可动画

### 相关方法

bindSpaceWarp <particlesys\_Node> <POmniFlect\_Node>

### 相关的 Binding Modifier

POmniFlectMod

本Modifier由方法bindSpaceWarp()自动创建，而不能由MAXScript直接创建。本Modifier没有相关的属性。

## 8.10.25 SDeflector: SpacewarpObject (导向球空间扭曲)

### 构造函数

SDeflector ...

### 属性

属性名称	数据类型	默认值	说明
<SDeflector>.friction	Float	0.0	指定粒子系统沿导向器运动时，速度受摩擦力减慢的效果。如果设定为0，粒子速度不减慢；如果设定为0.5，粒子速度减慢一半；设定为1时，粒子在打击对象表面时静止。可动画
<SDeflector>.collider	undefined	undefined	指定碰撞后使粒子运动方向发生偏转的Node对象
<SDeflector>.bounce	Float	1.0	指定粒子系统撞击导向球后的反弹速度。可动画
<SDeflector>.bouncevar	Float	0.0	指定反弹设置的变化量。可动画
<SDeflector>.chaos	Float	0.0	随机产生方向混乱的反弹变化。可动画
<SDeflector>.inheritVelocity <SDeflector>.velocity_inheritance	Float	1.0	指定导向球的运动如何影响到与其绑定的粒子系统。可动画
<SDeflector>.radius	Float	0.0	指定导向球的图标直径，由于粒子系统要与导向球撞击，所以图标的尺寸影响最终的撞击效果。可动画

## 相关方法

`bindSpaceWarp <particlesys_Node> <sdeflector_Node>`

### 相关的 Binding Modifier

#### SDeflectMod

本 Modifier 由方法 `bindSpaceWarp()` 自动创建，而不能由 MAXScript 直接创建。本 Modifier 没有相关的属性。

## 8.10.26 SOmniFlect: SpacewarpObject (泛方向导向球空间扭曲)

### 构造函数

`SOmniFlect ...`

### 属性

属性名称	数据类型	默认值	说明
<code>&lt;SOmniFlect&gt;.‘time on’</code> <code>&lt;SOmniFlect&gt;.time_on</code>	Integer	0	指定导向器开始作用的时间。可动画
<code>&lt;SOmniFlect&gt;.‘time off’</code> <code>&lt;SOmniFlect&gt;.time_off</code>	Integer	16000	指定导向器结束作用的时间。可动画
<code>&lt;SOmniFlect&gt;.affects</code> <code>&lt;SOmniFlect&gt;.reflects</code>	Float	100.0	指定粒子受泛方向导向球反作用的百分比。可动画，百分数
<code>&lt;SOmniFlect&gt;.bounce</code>	Float	1.0	指定粒子系统撞击泛方向导向球后的反弹速度。可动画
<code>&lt;SOmniFlect&gt;.bouncevar</code> <code>&lt;SOmniFlect&gt;.bounce_variation</code>	Float	0.0	指定反弹设置的变化量。可动画，百分数
<code>&lt;SOmniFlect&gt;.chaos</code>	Float	0.0	随机产生方向混乱的反弹变化。可动画
<code>&lt;SOmniFlect&gt;.Refracts</code>	Float	100.0	指定粒子受泛方向导向球折射作用的百分比。可动画，百分数
<code>&lt;SOmniFlect&gt;.‘pass velocity’</code> <code>&lt;SOmniFlect&gt;.pass_velocity</code>	Float	1.0	指定粒子穿过导向平面后的速度。默认为 1.0，表示粒子保持原来的速度；如果设定为 0.5，粒子穿过导向平面后速度减小到原来的一半。可动画
<code>&lt;SOmniFlect&gt;.‘passvel var’</code> <code>&lt;SOmniFlect&gt;.pass_velocity_variation</code>	Float	0.0	指定粒子穿过速度的变化量。可动画，百分数
<code>&lt;SOmniFlect&gt;.refraction</code> <code>&lt;SOmniFlect&gt;.distortion</code>	Float	50.0	指定折射的角度。如果设定为 0，表示不折射；如果设定为 100%，表示粒子系统沿导向平面平行运动。可动画，百分数
<code>&lt;SOmniFlect&gt;.‘refraction var’</code> <code>&lt;SOmniFlect&gt;.distortion_variation</code>	Float	0.0	指定扭曲设置的变化量。可动画，百分数

(续表)

属性名称	数据类型	默认值	说明
<SOmniFlect>.Diffusion	Float	0.0	通过随机改变粒子运行的角度，指定一个锥状散射的角度。可动画，百分数
<SOmniFlect>.'diffusion var' <SOmniFlect>. diffusion_variation	Float	0.0	指定散射设置的变化量。可动画，百分数
<SOmniFlect>.inheritVelocity <SOmniFlect>. velocity_inheritance	Float	1.0	指定导向器的运动如何影响到与其绑定的粒子系统。可动画
<SOmniFlect>.deceleration	Float	1.0	设置粒子的减速度。可动画
<SOmniFlect>.'decel var'	Float	0.0	设置属性.deceleration 的变化量。可动画，百分数
<SOmniFlect>.spawn <SOmniFlect>. spawn_percentage	Float	100.0	指定受粒子繁殖效果作用的百分比。可动画，百分数
<SOmniFlect>.friction	Float	0.0	指定粒子系统沿泛方向导向球运动时，速度受摩擦力减慢的效果。如果设定为0，粒子速度不减慢；如果设定为0.5，粒子速度减慢一半；设定为1时，粒子在打击对象表面时静止。可动画，百分数
<SOmniFlect>.collider	undefined	undefined	指定碰撞后使粒子发生偏转的Node对象
<SOmniFlect>.radius	Float	0.0	指定导向器的图标半径。可动画

### 相关方法

bindSpaceWarp <particlesys\_Node> <SOmniFlect\_Node>

### 相关的 Binding Modifier

SOmniFlectMod

本 Modifier 由方法 bindSpaceWarp() 自动创建，而不能由 MAXScript 直接创建。本 Modifier 没有相关的属性。

## 8.10.27 UDeflector: SpacewarpObject (通用导向器空间扭曲)

### 构造函数

UDeflector ...

### 属性

属性名称	数据类型	默认值	说明
<UDeflector>.bounce	Float	1.0	指定粒子系统撞击通用导向器后的反弹速度。可动画
<UDeflector>.bouncevar	Float	0.0	指定反弹设置的变化量。可动画
<UDeflector>.chaos	Float	0.0	随机产生方向混乱的反弹变化。可动画
<UDeflector>.Friction	Float	0.0	指定粒子系统沿通用导向器运动时，速度受摩擦力减慢的效果。如果设定为 0，粒子速度不减慢；如果设定为 0.5，粒子速度减慢一半；设定为 1 时，粒子在打击对象表面时静止。可动画，百分数
<UDeflector>.inheritVelocity <UDeflector>.velocity_inheritance	Float	1.0	指定导向器的运动如何影响到与其绑定的粒子系统。可动画
<UDeflector>.radius <UDeflector>.icon_size	Float	1.0	指定导向器的图标半径。可动画
<UDeflector>.collider	undefined	undefined	指定碰撞后使粒子发生偏转的 Node 对象

**注意** 在 3ds max 的 MAXScript 里还没有办法设置 Deflector 对象。

### 相关方法

bindSpaceWarp <particlesys\_Node> <UDeflector\_Node>

### 相关的 Binding Modifier

UDeflectorMod

本 Modifier 由方法 bindSpaceWarp() 自动创建，而不能由 MAXScript 直接创建。本 Modifier 没有相关的属性。

## 8.10.28 UOmniFlect: SpacewarpObject (通用泛方向导向器空间扭曲)

### 构造函数

UOmniFlect ...

### 属性

属性名称	数据类型	默认值	说明
<UOmniFlect>.'time on' <UOmniFlect>.time_on	Integer	0	指定导向器开始作用的时间。可动画
<UOmniFlect>.'time off' <UOmniFlect>.time_off	Integer	16000	指定导向器结束作用的时间。可动画
<UOmniFlect>.affects <UOmniFlect>.reflects	Float	10000.0	指定粒子受通用泛方向导向器反作用的百分比。可动画，百分数
<UOmniFlect>.bounce	Float	1.0	指定粒子系统撞击通用泛方向导向器后的反弹速度。可动画

(续表)

属性名称	数据类型	默认值	说明
<UOmniFlect>.bouncevar <UOmniFlect>.bounce_variation	Float	0.0	指定反弹设置的变化量。可动画，百分数
<UOmniFlect>.chaos	Float	0.0	随机产生方向混乱的反弹变化。可动画
<UOmniFlect>.Refracts	Float	100.0	指定粒子受通用导向器折射作用的百分比可动画，百分数
<UOmniFlect>.'pass velocity' <UOmniFlect>.pass_velocity	Float	1.0	指定粒子穿过导向平面后的速度。默认为1.0，表示粒子保持原来的速度；如果设定为0.5，粒子穿过导向平面后速度减小到原来的一半。可动画
<UOmniFlect>.'passvel var' <UOmniFlect>.pass_velocity_variation	Float	0.0	指定粒子穿过速度的变化量。可动画，百分数
<UOmniFlect>.refraction <UOmniFlect>.distortion	Float	50.0	指定折射的角度，如果设定为0，表示不折射；如果设定为100%，表示粒子系统沿导向平面平行运动。可动画，百分数
<UOmniFlect>.'refraction var' <UOmniFlect>.distortion_variation	Float	0.0	指定扭曲设置的变化量。可动画，百分数
<UOmniFlect>.Diffusion	Float	0.0	通过随机改变粒子运行的角度，指定一个锥状散射的角度。可动画，百分数
<UOmniFlect>.'diffusion var' <UOmniFlect>.diffusion_variation	Float	0.0	指定散射设置的变化量。可动画，百分数
<UOmniFlect>.Friction	Float	0.0	指定粒子系统沿导向器运动时，速度受摩擦力减慢的效果。如果设为0，粒子速度不减慢；如果设为0.5，粒子速度减慢一半；设为1时，粒子在打击对象表面时静止。可动画
<UOmniFlect>.inheritVelocity <UOmniFlect>.velocity_inheritance	Float	1.0	指定导向器的运动如何影响到与其绑定的粒子系统。可动画，百分数
<UOmniFlect>.deceleration	Float	1.0	指定粒子的减速度。可动画
<UOmniFlect>.'decel var'	Float	0.0	指定属性.deceleration 的变化量。可动画，百分数
<UOmniFlect>.spawn <UOmniFlect>.spawn_percentage	Float	100.0	指定受粒子繁殖效果作用的百分比。可动画，百分数
<UOmniFlect>.Pass_Velocity	Float	1.0	指定粒子穿过导向平面后的速度。默认为1.0，表示粒子保持原来的速度；如果设定为0.5，粒子穿过导向平面后速度减小到原来的一半。可动画

(续表)

属性名称	数据类型	默认值	说明
<UOmniFlect>.Pass_Velocity_Variation	Float	0.0	指定粒子穿过速度的变化量。可动画，百分数
<UOmniFlect>.radius <UOmniFlect>.icon_size	Float	0.0	指定通用泛方向导向器的图标尺寸。可动画
<UOmniFlect>.collider	undefined	undefined	指定碰撞后使粒子发生偏转的 Node 对象

### 相关方法

bindSpaceWarp <particlesys\_Node> <UOmniFlect\_Node>

### 相关的 Binding Modifier

#### UOmniFlectMod

本 Modifier 由方法 bindSpaceWarp() 自动创建，而不能由 MAXScript 直接创建。本 Modifier 没有相关的属性。

## 8.11 XRefObject: Node (外部参照对象)

在 3ds max 里一个 XRefObject 值代表一个 XRef 对象。当调用方法 xrefs.addNewXRefObject() 时，返回一个 XRefObject 值。在 3ds max 里有两类 XRef 对象：一类为非代理对象，另一类为代理对象。参照对象经常被用来在视窗里作为非代理对象的低精度替代物。

### 构造函数

xrefs.addNewXRefObject <filename\_string> <objectname\_string> [#proxy]

从指定场景文件里装载指定 Node 对象，并将它作为一个 XRef 对象。参数 <objectname\_string> 必须与场景文件里的对象的.name 属性匹配。如果指定了参数 #proxy，被装载对象将作为一个代理对象，否则作为一个非代理对象。

XRef 对象被装载后，总是被放在 World 坐标系的中心。

### 属性

属性名称	数据类型	默认值	说明
<XRefObject>.proxyFileName	String	不定	显示包含外部参照对象源的.max 文件的路径和文件名。可以将其指定给不同的路径和文件名
<XRefObject>.fileName	String	不定	显示源文件中源对象的名称
<XRefObject>.currentFileName	String	不定	指定包含代理对象的.max 场景的路径和文件名
<XRefObject>.objectName	String	不定	为同一指定场景中的代理对象指定名称
<XRefObject>.proxyObjectName	String	不定	设置非代理换对象的名称

(续表)

属性名称	数据类型	默认值	说明
<XRefObject>.currentObjectName	String	不定	如果属性.useProxy 为 False, 表示非代理对象的名称; 如果属性.useProxy 为 True, 表示代理对象的名称
<XRefObject>.useProxy	Boolean	不定	如果设置为 True, 视窗中显示代理对象, 否则, 视窗中显示非代理对象
<XRefObject>.renderProxy	Boolean	False	当设置为 On 时, 代理对象也将在渲染中显示; 否则, 对源外部参照对象进行渲染
<XRefObject>.updateMaterial	Boolean	False	当设置为 On 时, 更新为源文件中外部参照对象指定的材质; 否则, 不更新源材质
<XRefObject>.ignoreAnimation	Boolean	False	当设置为 On 时, 忽略源对象的堆栈动画, 但是保留所有的变换(旋转、移动、放缩等)动画设置

### 方法

updateXRef <XRefObject>

重新装载指定的 XRef 对象。

#### 8.11.1 XRefScene Values

在 3ds max 里一个 XRefScene 类值代表一个 XRef 场景对象。当调用方法 xref.addNewXRefFile() 和 xref.getXRefFile() 时, 返回一个 XRefScene 类值。

XRef 场景对象可以嵌套。比如: 在场景文件 A 里包含一个 XRef 文件 B 里定义的 XRef 场景对象, 而 XRef 文件 B 里又包含一个 XRef 文件 C 里定义的 XRef 场景对象。为了存取嵌套的 XRef 场景对象, 在 XrefScene 类方法里提供了一个可选的关键词参数 root:。如果在调用方法时指定了该参数, 这些方法将作用在参数 root: 指定的<XrefScene>值的根对象上; 否则将作用在当前场景的根对象上。为了从场景文件 A 获取 XRef 文件 C 里的 XRef 场景对象, 必须先装载文件 A, 然后做如下操作:

```
bXref = xrefs.getXRefFile 1           --作用在当前场景的根对象上
cXref = xrefs.getXRefFile 1 root:bXref --作用在 bXref 的根对象上
cRoot = cXref.tree                   --获取 c 对象的根对象
```

### 构造函数

1 . xrefs.addNewXRefFile <filename\_string> [ #noLoad ] [root:<XRefScene>]

向场景里加入一个 XRef 场景对象, 然后返回一个 XrefScene 值。

如果没有指定可选参数#noLoad, XRef 场景文件<filename\_string>被马上装载, 场景被刷新。如果指定可选参数#noLoad, XRef 场景文件<filename\_string>不会马上被装载, 直到用户要求刷新时, 场景才会被刷新。可以用方法 updateXRef() 来装载 XRef 场景文件并刷新场景。

如果指定参数 root:<XRefScene>, 新的 XRef 场景对象被看作是 root: 参数指定的<XrefScene>对象所在场景文件里的一个 XRef 场景对象。如果 root: 参数指定的<XrefScene>

对象被重新装载——可能是因为当前场景被存盘后又重载;也可能是因为对 root: 参数指定的<XrefScene>对象调用方法 updateXRef(), 新 XRef 场景对象都会从场景里被删除。

## 2. xrefs.getXRefFile <index> [root:<XRefScene>]

返回一个 XRefScene 值。

如果没有指定参数 root:<XRefScene>, 返回指定序号<index>对应的 XRef 场景对象, 该序号与 XRef Scenes 对话框里 XRef 场景对象列表序号对应。

如果有指定参数 root:<XRefScene>, 返回嵌套在 Root:参数指定<XrefScene>对象所在 XRef 场景文件里指定序号的 XRef 场景对象。序号从 1 开始, 可以用方法 xrefs.getXRefFileCount()获取场景里 XRef 场景对象的数目。

### 属性

属性名称	数据类型	默认值	说明
<XRefScene>.filename	String	不定	设置 XRef 场景对象的文件名称。如果将本属性直接修改成新文件名, 场景里将显示指定文件里的 XRef 场景对象, 用其来取代原来的 XRef 场景对象
<XRefScene>.tree	Node	Node	XRef 场景对象的根对象。可以用它来存取 XRef 场景对象的子对象。如:  axref=xrefs.getXRefFile 1 axref.tree.children 返回值为: #children (\$Box01, \$Box02) 而用下面语句: axref.tree.children[1].width 返回对象 Box01 的 width 属性
<XRefScene>.parent	Node	undefined	XRef 场景对象的父对象, 可以设置成场景里任何 Node 对象
<XRefScene>.autoUpdate	Boolean	False	当设置为 On 时, 对 XRef 场景对象的原文件修改并保存之后, 在当前场景中引用的 XRef 场景会自动更新
<XRefScene>.boxDisp	Boolean	False	当设置为 On 时, XRef 场景对象中的 Node 对象将显示为一个粗略的边界盒
<XRefScene>.hidden	Boolean	False	当设置为 On 时, XRef 场景对象将被隐藏
<XRefScene>.disabled	Boolean	False	当设置为 On 时, XRef 场景对象将被禁用
<XRefScene>.ignoreLights	Boolean	False	当设置为 On 时, XRef 场景对象中的灯光对象不会显示在场景里
<XRefScene>.ignoreCameras	Boolean	False	当设置为 On 时, XRef 场景对象中的摄影机对象不会显示在场景里
<XRefScene>.ignoreShapes	Boolean	False	当设置为 On 时, XRef 场景对象中的 Shape 类对象不会显示在场景里
<XRefScene>.ignoreHelpers	Boolean	False	当设置为 On 时, XRef 场景对象中的 Helper 类对象不会显示在场景里
<XRefScene>.ignoreAnimation	Boolean	False	当设置为 On 时, 将忽略 XRef 场景对象中的动画设置

## 方法

### 1. delete <XRefScene>

删除指定 XRef 场景对象。

### 2. merge <XRefScene>

将指定 XRef 场景对象中的 Node 对象合并到场景里，然后删除该 XRef 场景对象。

### 3. updateXRef <XRefScene>

重载指定的 XRef 场景对象。如果返回 True，表示操作成功；否则，操作失败。

### 4. flagChanged <XRefScene>

本方法告诉系统指定 XRef 场景对象被修改过，需要被更新。用方法 xrefs.updateChangedXRefs() 可以更新 XRef 场景对象。

## 相关方法

### 1. xrefs.getXRefFileCount [root:<XRefScene>]

如果没有指定可选参数 root:<XRefScene>，返回场景中顶级 XRef 场景对象的总数；如果有指定参数 root:，返回<XRefScene>所在的 XRef 文件里的 XRef 场景对象总数。

### 2. xrefs.deleteAllXRefs [root:<XRefScene>]

如果没有指定参数 root:<XRefScene>，删除当前场景中所有 XRef 场景对象；如果有指定参数 root:，删除指定 XRef 场景对象<XRefScene>和它的嵌套 XRef 场景对象。

### 3. xrefs.updateChangedXRefs [#noRedraw] [root:<XRefScene>]

如果没有指定参数 root:<XRefScene>，更新当前场景中所被修改过的 XRef 场景对象；如果有指定参数 root:，仅更新指定 XRef 场景对象<XRefScene>被修改过的嵌套 XRef 场景对象。

如果指定参数#noRedraw，视窗直到装载 XRef 场景对象时才会被更新。

如果 XRef 场景对象更新成功，返回 True；否则，返回 False。

### 4. xrefs.findUnresolvedXRefs [root:<XRefScene>]

返回一个字符串数组。如果没有指定参数 root:<XRefScene>，数组元素为所有未装载的 XRef 场景对象的文件名，如果有指定参数 root:，数组元素仅包含<XRefScene>里嵌套的未装载的场景对象。

### 5. xrefs.attemptUnresolvedXRefs [root:<XRefScene>]

如果没有指定参数 root:<XRefScene>，装载场景中所有未装载的 XRef 场景对象，如果有指定参数 root:，仅装载<XRefScene>里嵌套的未装载的场景对象。

## 8.12 Track View Node (轨迹视窗节点)

我们可以用 MAXScript 在 Track View 视窗里存取、修改 Global Track (Global 轨道) 和 Video Post 轨道里的控制器。除此之外，还可以创建新的命名 Global Track，这些可以用来在脚本工具里作为可动画的参数。

3ds max 系统提供了一些与 Track View 视窗相关的方法，具体参见本书 20.8 节。

在 3ds max 里，用 MAXTVNode 类值来表示 Track View 视窗的 Global Track 对象。一个

MAXTVNode 值可以包含任意数量的 MAXTVNode 或动画控制器。包含在 MAXTVNode 里的 MAXTVNode 或动画控制器可以像属性一样在 MAXScript 里进行存取。MAXTVNode 值本身并不存储动画数据，其更像是一个容器，存储着一个或多个 MAXTVNode 或动画控制器。

有三个 TrackView 类对象可以通过全局系统变量进行存取：一是 TrackViewNodes，包含根对象，可以向其中加入自己的 Track；一是 globalTracks，包含现有的 Global Track 对象，通过它可以存取 Global Track 控制器；还有一个是 VideoPostTracks，包含 VideoPost track 对象，通过它可以存取 Video Post 控制器。

### 构造函数

#### 1. TrackViewNodes

包含 Track View 视窗里所有顶级对象（或根对象）的全局变量。该全局变量为只读。

#### 2. GlobalTracks

包含 Track View 视窗里所有顶级 Global Tracks 对象的全局变量。该全局变量为只读。

#### 3. VideoPostTracks

包含 Track View 视窗里所有顶级 Video Post Track 对象的全局变量。该全局变量为只读。在 3D Studio VIZ 里，本变量的值为 undefined。

#### 4. NewTrackViewNode [ <maxtvnode> ] <name\_string> [ #hidden ]

如果有指定参数<maxtvnode>，向它添加一个子级 Track 对象；否则，在 Track View 视窗里创建一个新的顶级 Track。如果有指定可选参数#hidden，新加入 Track 对象在 Track View 视窗里不可见。参数<name\_string>为新加入 Track 对象的名称。

### 属性

#### <maxtvnode>.name: String

Track View 对象的名称，它与 Track View 视窗里显示的字符相同。

一个 Track View 对象的子 Track 和子控制器也可以像属性一样进行存取。如全局变量 TrackViewNodes 有下面两个默认的属性：

- ◆ TrackViewNodes.Global\_Tracks: MAXTVNode
- ◆ TrackViewNodes.Video\_Post: MAXTVNode

全局变量 globalTracks 有下面默认的属性：

- ◆ GlobalTracks.Float: Float\_list Controller
- ◆ GlobalTracks.point3: Point3\_list Controller
- ◆ GlobalTracks.position: Position\_list Controller
- ◆ GlobalTracks.rotation: Rotation\_list Controller
- ◆ GlobalTracks.scale: Scale\_list Controller
- ◆ GlobalTracks.Block\_Control: Block\_control Controller

### 方法

#### 1. deleteTrackViewNode [ <parent\_maxtvnode> ] <maxtvnode>

从指定父对象里删除指定对象。如果参数<parent\_maxtvnode>没有指定，从顶级对象里删除指定对象。

2. addTrackViewController <maxtvnode> <controller> <name>

给指定 Track View 对象添加指定子控制器。

3. deleteTrackViewController <maxtvnode> <controller>

从指定 Track View 对象里删除指定控制器。

示例：

```
--存取 Video Post track 属性
videoPostTracks.glow.size = 5
--为 Rollout 面板里可动画的参数建立几个可见的 Track
my_tracks = newTrackViewNode "Grinner"
happy_ctrl = bezier_position()
addTrackViewController my_tracks happy_ctrl "Happy"
...
rollout grinner ...
(
  ...
  on happy changed val do
  (
    ...
    happy_ctrl.value = [x, y, z]
    ...
  )
  ...
)
--向 globaltracks.Float.controller 里添加一个新的 Float Controller
globalTracks.Float.controller.available.controller = bezier_Float()
```

## 8.13 NURBS Node 属性和方法

### 属性

下面是 NURBS 对象子对象选择集的几个可用属性：

- ◆ <node>.selectedCurveCVs
- ◆ <node>.selectedCurves
- ◆ <node>.selectedImports
- ◆ <node>.selectedPoints
- ◆ <node>.selectedSurfaces
- ◆ <node>.selectedSurfCVs
- ◆ <node>.curveCVs
- ◆ <node>.curves
- ◆ <node>.imports
- ◆ <node>.surfaces

- ◆ <node>.surfCVs
- ◆ <node>.points

所有上面这些方法都返回一个 NURBSSelection 值，并且可以像 Editable\_Mesh 对象一样对它进行子对象操作。

### 方法

下面这些与 NURBS 有关的方法操作对象为场景对象。

#### 1. getNURBSSet <node> [#relational]

返回与指定 NURBS 场景对象对应的 NURBSSet 值。本方法可以用来存取 3ds max 可编辑 NURBS 对象的内部对象。

如果可选参数#relational 没有被指定，返回的 NURBSSet 值仅包含独立的曲线和曲面，所有互相关联的对象被分解成相应独立的对象；如果指定参数<node>为一个包含两个 CV 曲面和一个关联的混合曲面的场景对象，则会被分解成三个 CV 曲面，混合关联数据将被丢失；如果可选参数#relational 被指定，返回的 NURBSSet 值包含两个 CV 曲面和一个 NURBS 混合曲面，这样返回的 NURBSSet 值就是一个场景对象的完全代表：对 NURBSSet 值里的 NURBS 对象属性所作的任何更改会直接反映到源场景对象上。

#### 2. addNURBSSet <node> <nurbsSet>

将指定 NURBSSet 包含的 NURBS 对象添加为指定场景对象的子对象。在调用本方法后，<nurbsSet>就是新的场景对象子对象的代表。对 NURBSSet 值里的 NURBS 对象属性所作的任何更改都会直接反映到新子对象上。

#### 3. transform <node> <nurbsId\_or\_Id\_array> <matrix3>

将指定场景对象的指定子对象在场景对象的 Local 坐标系里进行指定转换。子对象由.NURBSId 属性指定。

#### 4. breakCurve <node> <nurbsId> <u\_param>

将指定场景对象的指定子对象曲线在沿曲线指定长度<u\_param>处断开。

#### 5. breakSurface <node> <nurbsId> (#U | #V) <u\_or\_v\_param>

将指定场景对象的指定子对象曲线在指定 U 或 V 方向（由#U|#V 指定）的指定长度<u\_or\_v\_param>处断开。

#### 6. joinCurves <node> <nurbsId1> <nurbsId2> <tolerance\_Float> \

[flip1:<boolean>] [flip2:<boolean>]

将指定场景对象的两个指定子对象曲线连接为单条曲线。原有两条曲线的端点用新的线段连接。

参数<tolerance\_Float>为一个距离，单位为 3ds max 系统单位。如果要连接的两条曲线的距离大于该值，本方法先创建一个混合曲线，然后连接三部分；如果要连接的两条曲线的距离小于该值，或指定两条曲线迭合为同一条曲线，不创建混合曲线。

创建一个混合曲线，然后将三部分连接为一条曲线，这是一种比较好的技术。生成的结果与源曲线更吻合。如果没有这一步，为了保持曲线的平滑度，其结果曲线可能与源曲线偏离。

#### 7. joinSurfaces <node> <nurbsId1> <nurbsId2> \

<edge1\_Integer><edge2\_Integer><tolerance\_Float>

将指定场景对象的两个指定子对象面连接为单个曲面。原有两个曲面的指定边用新的曲面连接。

参数<tolerance\_Float>为一个距离，单位为3ds max系统单位。如果要连接的两个曲面的距离大于该值，本方法先创建一个混合边，然后连接三部分；如果要连接的两个曲面的距离小于该值，或指定两个曲面迭合为同一个曲面，不创建混合面。

#### 8. makeIndependent <node> <nurbsId>

将指定关联子对象(Fillet、Offset、Blend等操作生成的子对象)变成一个独立的对象。

#### 9. setViewApproximation <node> <surfaceApproximation>

setRenderApproximation <node> <surfaceApproximation>

setSurfaceDisplay <node> <nurbsdisplay>

这些方法为指定NURBS场景对象设置显示和曲面/曲线拟合参数。

### 相关方法

#### 1. canConvertTo <node> <class>

参见本书8.2.8节。

#### 2. convertTo <node> <class> mapped

参见本书8.2.8节。

#### 3. convertToNURBSSurface <node> mapped

convertToNURBSCurve <node> mapped

参见本书8.2.8节。

#### 4. pointSelection <node> <point\_index>

参见本书8.2.11节。

#### 5. stopCreating <node>

参见本书8.2.11节。

### 8.13.1 NURBS类

下面是所有的NURBS类：

- ◆ NURBSObject(NURBS对象)
- ◆ NURBSPoint: NURBSObject(点子对象)
- ◆ NURBSCurveConstPoint: NURBSPoint(曲线点)
- ◆ NURBSCurveIntersectPoint: NURBSPoint(曲线-曲线相交点)
- ◆ NURBSCurveSurfaceIntersectPoint: NURBSPoint(曲面-曲线相交点)
- ◆ NURBSIndependentPoint: NURBSPoint(独立点)
- ◆ NURBSPointConstPoint: NURBSPoint(偏移点)
- ◆ NURBSSurfConstPoint: NURBSPoint(曲面点)
- ◆ NURBSControlVertex: NURBSObject(控制顶点对象)
- ◆ NURBSCurve: NURBSObject(曲线子对象)

- ◆ NURBSBlendCurve: NURBSCurve (混合曲线)
- ◆ NURBSChamferCurve: NURBSCurve (切角曲线)
- ◆ NURBSCVCurve: NURBSCurve (CV 曲线子对象)
- ◆ NURBSCurveOnSurface: NURBSCVCurve (曲面上的 CV 曲线)
- ◆ NURBSFilletCurve: NURBSCurve (圆角曲线)
- ◆ NURBSIsoCurve: NURBSCurve (U 向和 V 向等参曲线)
- ◆ NURBSMirrorCurve: NURBSCurve (镜像曲线)
- ◆ NURBSOffsetCurve: NURBSCurve (偏移曲线)
- ◆ NURBSPointCurve: NURBSCurve (点曲线子对象)
- ◆ NURBSPointCurveOnSurface: NURBSPointCurve (曲面上的点曲线)
- ◆ NURBSProjectNormalCurve: NURBSCurve (法向投射曲线)
- ◆ NURBSProjectVectorCurve: NURBSCurve (矢量投射曲线)
- ◆ NURBSSurfaceEdgeCurve: NURBSCurve (曲面边曲线)
- ◆ NURBSSurfaceNormalCurve: NURBSCurve (曲面法线曲线)
- ◆ NURBSSurfSurfIntersectionCurve: NURBSCurve (曲面-曲面相交曲线)
- ◆ NURBSXFormCurve: NURBSCurve (变换曲线)
- ◆ NURBSSurface: NURBSObject (曲面子对象)
- ◆ NURBS1RailSweepSurface: NURBSSurface (单轨扫描曲面)
- ◆ NURBS2RailSweepSurface: NURBSSurface (双轨扫描曲面)
- ◆ NURBSBlendSurface: NURBSSurface (混合曲面)
- ◆ NURBSCapSurface: NURBSSurface (封口曲面)
- ◆ NURBSCVSurface: NURBSSurface (CV 曲面子对象)
- ◆ NURBSExtrudeSurface: NURBSSurface (挤出曲面)
- ◆ NURBSFilletSurface: NURBSSurface (圆角曲面)
- ◆ NURBSLatheSurface: NURBSSurface (车削曲面)
- ◆ NURBSMirrorSurface: NURBSSurface (镜像曲面)
- ◆ NURBSMultiCurveTrimSurface: NURBSSurface (多重曲线修剪曲面)
- ◆ NURBSNblendSurface: NURBSSurface (混合曲面)
- ◆ NURBSOffsetSurface: NURBSSurface (偏移曲面)
- ◆ NURBSPointSurface: NURBSSurface (点曲面子对象)
- ◆ NURBSRuledSurface: NURBSSurface (规则曲面)
- ◆ NURBSULoftSurface: NURBSSurface (U 向放样曲面)
- ◆ NURBSUVLoftSurface: NURBSSurface (UV 放样曲面)
- ◆ NURBSXFormSurface: NURBSSurface (变换曲面)
- ◆ NURBSTexturePoint: NURBSObject (纹理曲面)
- ◆ NURBSDisplay: Value
- ◆ NURBSSelection: Value
- ◆ NURBSSet: Value

- ◆ NURBSSurfaceApproximation: Value
- ◆ NURBSTextureSurface: Value

### 8.13.2 NURBSCurveshape: Shape

不能由 MAXScript 创建本类实例。

#### 属性

属性名称	数据类型	说明
<nurbscurveshape>.angle	Float	设置曲线对象旋转的角度
<nurbscurveshape>.renderable	Boolean	当设置为 On 时, 可以将曲线对象设定为可渲染的 NURBS 曲线
<nurbscurveshape>.mapCoords	Boolean	当设置为 On 时, 为可渲染的曲线对象自动指定贴图坐标
<nurbscurveshape>.thickness	Float	设置可渲染曲线对象剖面的直径
<nurbscurveshape>.sides	Integer	设置可渲染曲线对象剖面的边数
<nurbscurveshape>.viewport_thickness	Float	设置二维图形在视窗中的剖面直径
<nurbscurveshape>.viewport_sides	Integer	设置二维图形在视窗中的剖面边数
<nurbscurveshape>.viewport_angle	Float	设置剖面图形在视窗中的沿路径轴向旋转的角度
<nurbscurveshape>.displayRenderMesh	Boolean	当设置为 On 时, 将可渲染的二维图形对象显示为网格对象
<nurbscurveshape>.useViewportSettings	Boolean	当设置为 On 时, 二维图形对象可以被渲染
<nurbscurveshape>.displayRenderSettings	Boolean	当设置为 On 时, 依据视图设置将可渲染的二维图形对象显示为网格对象

### 8.13.3 NURBSObject 通用属性

NURBSObject 是所有 NURBS 类的父类, 所以不能直接创建。以下是所有 NURBS 类共有的属性, 但 NURBSSet 除外。

属性名称	数据类型	说明
<nurbsobject>.name	string	存取 NURBS 对象的名称
<nurbsobject>.index	Integer	只读, 返回指定 NURBS 对象的指定序号子对象, 仅在 NURBSSet 值通过 NURBSNode() 函数创建一个 NURBS 对象实例或指定 NURBS 对象为一个独立 NURBSSet 值的子对象时有效
<nurbsobject>.nurbsID	Integer	只读, 返回指定 NURBS 对象的指定 ID 码子对象, 仅在 NURBSSet 值通过 NURBSNode() 函数创建一个 NURBS 对象实例或指定 NURBS 对象为一个独立 NURBSSet 值的子对象时有效
<nurbsobject>.selected	Boolean	表示指定子对象是否在 3ds max 用户界面的 Modifier 面板下被选择。可以通过将本属性设为 True 或 False 来强制选择子对象或取消对子对象的选择

### 8.13.4 NURBSPoint: NURBSObject (点子对象)

NURBSPoint 是所有 NURBS point 类的父类，所以不能直接创建。以下是所有 NURBSPoint 类共有的属性：

- ◆ <nurbspoint>.pos: point3
- ◆ <nurbspoint>.x: Float
- ◆ <nurbspoint>.y: Float
- ◆ <nurbspoint>.z: Float

### 8.13.5 NURBSCurveConstPoint: NURBSPoint (曲线点)

#### 构造函数

```
NURBSCurveConstPoint [<property>:<val>]...
getObject <nurbsset> <index>
```

#### 属性

属性名称	数据类型	说明
<nurbscurveconstpoint>.parent	Integer	存取由 NURBSet 序号指定的父对象曲线
<nurbscurveconstpoint>.parentID	Integer	存取由 NURBSId 指定的父对象曲线
<nurbscurveconstpoint>.uParam	Float	指定点在曲线上的位置或相对于曲线的位置
<nurbscurveconstpoint>.type	Name	指定曲线与点之间的变换关系。参数说明如下： #onObject: 点依赖于 U 向位置上的曲线； #offset: 按照相对 (对象空间) X、Y、Z 位置移动点； #normal: 在 U 向位置上, 沿着曲线法线的方向移动点； #tangent: 在 U 向位置上, 沿着切线方向移动点
<nurbscurveconstpoint>.offset	point3	指定偏移曲线点的对象空间位置
<nurbscurveconstpoint>.normal	Float	指定曲线法线沿线的距离
<nurbscurveconstpoint>.uTangent	Float	指定沿着切线方向与曲线的距离
<nurbscurveconstpoint>.trimCurve	Boolean	当设置为 On 时, 将针对曲线点的 U 向位置修剪父曲线；否则 (默认情况下), 不能修剪父曲线
<nurbscurveconstpoint>.flipTrim	Boolean	当设置为 On 时, 将以相反的方向进行修剪

### 8.13.6 NURBSCurveIntersectPoint: NURBSPoint (曲线-曲线相交点)

#### 构造函数

```
NURBSCurveIntersectPoint [<property>:<val>]...
getObject <nurbsset> <index>
```

#### 属性

属性名称	数据类型	说明
<nurbscurveintersectpoint>.parent1	Integer	存取由 NURBSet 序号指定的第一个父对象曲线
<nurbscurveintersectpoint>.parent1ID	Integer	存取由 NURBSId 指定的第一个父对象曲线
<nurbscurveintersectpoint>.parent2	Integer	存取由 NURBSet 序号指定的第二个父对象曲线
<nurbscurveintersectpoint>.parent2ID	Integer	存取由 NURBSId 指定的第二个父对象曲线
<nurbscurveintersectpoint>.trimCurve1	Boolean	当设置为 On 时, 在相交点剪切第一个父对象曲线
<nurbscurveintersectpoint>.trimCurve2	Boolean	当设置为 On 时, 在相交点剪切第二个父对象曲线
<nurbscurveintersectpoint>.flipTrim1	Boolean	当设置为 On 时, 将从相反的方向开始剪切第一个父对象曲线
<nurbscurveintersectpoint>.flipTrim2	Boolean	当设置为 On 时, 将从相反的方向开始剪切第二个父对象曲线

### 8.13.7 NURBSCurveSurfaceIntersectPoint: NURBSPoint (曲面-曲线相交点)

#### 构造函数

```
NURBSCurveSurfaceIntersectPoint [<property>:<val>]...
getObject <nurbsset> <index>
```

#### 属性

属性名称	数据类型	说明
<nurbscurvesurfaceintersectpoint>.parent1	Integer	存取由 NURBSet 序号指定的父对象曲面
<nurbscurvesurfaceintersectpoint>.parent1ID	Integer	存取由 NURBSId 指定的父对象曲面
<nurbscurvesurfaceintersectpoint>.parent2	Integer	存取由 NURBSet 序号指定的父对象曲线
<nurbscurvesurfaceintersectpoint>.parent2ID	Integer	存取由 NURBSId 指定的父对象曲线
<nurbscurvesurfaceintersectpoint>.trimCurve	Boolean	当设置为 On 时, 将针对曲线点的 U 向位置修剪父曲线; 否则(默认情况下), 不能修剪父曲线
<nurbscurvesurfaceintersectpoint>.flipTrim	Boolean	当设置为 On 时, 将以相反的方向进行修剪
<nurbscurvesurfaceintersectpoint>.seed	Float	更改曲线上种子值的 U 位置

### 8.13.8 NURBSIndependentPoint: NURBSPoint (独立点)

#### 构造函数

```
NURBSIndependentPoint <point3>
getObject <nurbsset> <index>
getPoint <nurbspointsurface> <index>
getPoint <nurbspointcurve> <index>
getPoint <nurbspointcurveonsurface> <index>
```

## 操作符

<nurbsindependentpoint> == <nurbsindependentpoint>  
<nurbsindependentpoint> != <nurbsindependentpoint>

## 属性

NURBSIndependentPoint 除了通用属性外，没有额外的属性。

### 8.13.9 NURBSPointConstPoint: NURBSPoint (偏移点)

#### 构造函数

NURBSPointConstPoint [<property>:<val>]...  
getObject <nurbsset> <index>

#### 属性

属性名称	数据类型	说明
<nurbspointconstpoint>.parent	Integer	存取由 NURBSet 序号指定的父对象点
<nurbspointconstpoint>.parentID	Integer	存取由 NURBSId 指定的父对象点
<nurbspointconstpoint>.type	Name	指定曲线与点之间的变换关系。参数说明如下： #onObject: 从属点具有与原始点、父点相同的位置 #offset: 启用点偏移
<nurbspointconstpoint>.offset	point3	指定偏移曲线点的对象空间位置

### 8.13.10 NURBSSurfConstPoint: NURBSPoint (曲面点)

#### 构造函数

NURBSSurfConstPoint [<property>:<val>]...  
getObject <nurbsset> <index>

#### 属性

属性名称	数据类型	说明
<nurbssurfconstpoint>.parent	Integer	存取由 NURBSet 序号指定的父对象曲面
<nurbssurfconstpoint>.parentID	Integer	存取由 NURBSId 指定的父对象曲面
<nurbssurfconstpoint>.type	Name	指定曲线与点之间的变换关系。参数说明如下： #onObject: 依赖于曲面的点位于 U 向位置和 V 向位置指定的位置 #offset: 按照相对 (对象空间) X、Y、Z 位置移动点 #normal: 沿着曲面法线的方向移动点 #tangent: 沿着 UV 向位置的切线移动点
<nurbssurfconstpoint>.uParam	Float	如果该点位于曲面上，指定点基于曲面局部 U 坐标值
<nurbssurfconstpoint>.vParam	Float	如果该点位于曲面上，指定点基于曲面局部 V 坐标值

(续表)

属性名称	数据类型	说明
<nurbssurfconstpoint>.offset	point3	指定偏移曲线点的对象位置空间
<nurbssurfconstpoint>.normal	Float	指定曲线法线沿线的距离
<nurbssurfconstpoint>.uTangent	Float	指定曲面切线U向沿线的距离
<nurbssurfconstpoint>.vTangent	Float	指定曲面切线V向沿线的距离

### 8.13.11 NURBSControlVertex: NURBSObject (控制顶点对象)

#### 构造函数

```
NURBSControlVertex <point3> [<weight_Float>]
getCV <nurbscurve> <index>
getCV <nurbssurface> <u_index> <v_index>
```

#### 操作符

```
<nurbscontrolvertex> == <nurbscontrolvertex>
<nurbscontrolvertex> != <nurbscontrolvertex>
```

#### 属性

<nurbscontrolvertex>.weight: Float

设置控制顶点的权重。

### 8.13.12 NURBSCurve: NURBSObject (曲线子对象)

NURBSObject 是所有 NURBS curve 类的父类, 所以不能直接创建。以下是所有 NURBS curve 类共有的属性。

#### 属性

属性名称	数据类型	说明
<nurbscurve>.isClosed	Boolean	只读, 当设置为 On 时, 曲线是封闭的; 否则, 是开放的
<nurbscurve>.numTrimPoints	Integer	只读, 存取曲线剪切点的数量
<nurbscurve>.parameterRangeMin	Float	只读, 在 NURBSCurves 关联方法中参数<u_parm>所包含的最小值
<nurbscurve>.parameterRangeMax	Float	只读, 在 NURBSCurves 关联方法中参数<u_parm>所包含的最大值
<nurbscurve>.matID	Integer	存取曲线的材质 ID 码

#### 方法

以下是所有 NURBSCurve 类共有的方法:

1. evalPos <nurbscurve> <u\_param>

返回沿曲线指定长度处点的坐标。

2. evalTangent <nurbscurve> <u\_param>

返回沿曲线指定长度处点的正切矢量。

### 8.13.13 NURBSBlendCurve: NURBSCurve (混合曲线)

#### 构造函数

NURBSBlendCurve [<property>:<val>]...

getObject <nurbsset> <index>

#### 属性

属性名称	数据类型	说明
<nurbsblendcurve>.parent1	Integer	存取由 NURBSet 序号指定的第一个父对象曲线
<nurbsblendcurve>.parent1ID	Integer	存取由 NURBSId 指定的第一个父对象曲线
<nurbsblendcurve>.parent2	Integer	存取由 NURBSet 序号指定的第二个父对象曲线
<nurbsblendcurve>.parent2ID	Integer	存取由 NURBSId 指定的第二个父对象曲线
<nurbsblendcurve>.flip1	Boolean	当设置为 On 时, 使用第一个父对象曲线的末端; 否则, 使用曲线的始端
<nurbsblendcurve>.flip2	Boolean	当设置为 On 时, 使用第二个父对象曲线的末端; 否则, 使用曲线的始端
<nurbsblendcurve>.tension1	Float	控制第一条曲线边上的张力
<nurbsblendcurve>.tension2	Float	控制第二条曲线边上的张力

### 8.13.14 NURBSChamferCurve: NURBSCurve (切角曲线)

#### 构造函数

NURBSChamferCurve [<property>:<val>]...

getObject <nurbsset> <index>

#### 属性

属性名称	数据类型	说明
<nurbschamfercurve>.parent1	Integer	存取由 NURBSet 序号指定的第一个父对象曲线
<nurbschamfercurve>.parent1ID	Integer	存取由 NURBSId 指定的第一个父对象曲线
<nurbschamfercurve>.parent2	Integer	存取由 NURBSet 序号指定的第二个父对象曲线
<nurbschamfercurve>.parent2ID	Integer	存取由 NURBSId 指定的第二个父对象曲线
<nurbschamfercurve>.length1	Float	设置第一条曲线的沿线距离
<nurbschamfercurve>.length2	Float	设置第二条曲线的沿线距离
<nurbschamfercurve>.flip1	Boolean	当设置为 On 时, 使用第一个父对象曲线的末端; 否则, 使用曲线的始端

(续表)

属性名称	数据类型	说明
<nurbschamfercurve>.flip2	Boolean	当设置为 On 时, 使用第二个父对象曲线的末端; 否则, 使用曲线的始端
<nurbschamfercurve>.trim1	Boolean	当设置为 On 时, 将针对切角曲线修剪第一个父对象曲线; 否则(默认情况下), 不能修剪父曲线
<nurbschamfercurve>.trim2	Boolean	当设置为 On 时, 将针对切角曲线修剪第二个父对象曲线; 否则(默认情况下), 不能修剪父曲线
<nurbschamfercurve>.flipTrim1	Boolean	当设置为 On 时, 将从相反的方向开始修剪第一个父对象曲线
<nurbschamfercurve>.flipTrim2	Boolean	当设置为 On 时, 将从相反的方向开始修剪第二个父对象曲线

### 8.13.15 NURBSCVCurve: NURBSCurve (CV 曲线条子对象)

#### 构造函数

NURBSCVCurve [<property>:<val>] [closed:<boolean>]...

getObject <nurbsset> <index>

#### 属性

属性名称	数据类型	说明
<nurbscvcurve>.order	Integer	返回曲线的级数, 比如线性方程返回 1, 二次方程返回 2, 三次方程返回 3
<nurbscvcurve>.numKnots	Integer	返回曲线里的结点数。如果本属性被改变, 原有的结点数据不会被保留
<nurbscvcurve>.numCVs	Integer	返回曲线里的控制点(CV)数。如果本属性被改变, 原有的控制点数据不会被保留
<nurbscvcurve>.transform	matrix3	指定 NURBSCVCurve 的转换矩阵。本矩阵控制 NURBSSet 对象里子对象的相对位置
<nurbscvcurve>.endsOverlap	Boolean	只读, 如果返回 True, 表示曲线末端重叠, 即使曲线不闭合
<nurbscvcurve>.autoParam	Name	本属性与 CV Curve 用户界面卷展栏里的 Automatic Reparam 选项对应, 有效值有: #notAutomatic、#autoCentripetal、#autoUniform、#notAutomatic、#autoCentripetal 和#autoUniform, 默认值为#notAutomatic

#### 方法

1. close <nurbscvcurve>

强制将曲线闭合。

2. getKnot <nurbscvcurve> <index>

setKnot <nurbscvcurve> <index> <Float>

存取指定序号的结点, 序号从 1 开始。

3. getCV <nurbscurve> <cv\_index>

返回一个 NURBSControlVertex 值，表示指定序号的 CV 点，序号从 1 开始。

4. setCV <nurbscurve> <cv\_index> <NURBSControlVertex>

将指定序号的 CV 点设置为指定 NURBSControlVertex 值。

5. refine <nurbscurve> <u\_param>

沿曲线指定距离处插入一个新的 CV 点。

6. reparameterize <nurbscurve> (#centripetal | #uniform)

按指定方式重新计算曲线参数。

### 注意

CV 曲线和曲面必须遵循一个原则：曲线的级数+CV 点数=顶点数。如果不符合这个原则，大多数情况下不能创建任何对象，还可能导致系统给出一个错误声明。

## 8.13.16 NURBSCurveOnSurface: NURBSCVCurve (曲面上的 CV 曲线)

### 构造函数

NURBSCurveOnSurface [<property>:<val>] [closed:<boolean>]...

getObject <nurbsset> <index>

### 属性

属性名称	数据类型	说明
<nurbscurveonsurface>.parent	Integer	存取由 NURBSet 序号指定的父对象曲线
<nurbscurveonsurface>.parentID	Integer	存取由 NURBSId 指定的父对象曲线
<nurbscurveonsurface>.trim	Boolean	当设置为 On 时，将针对曲线修剪曲面；否则，不能修剪曲面
<nurbscurveonsurface>.flipTrim	Boolean	当设置为 On 时，将以相反的方向进行修剪

## 8.13.17 NURBSFilletCurve: NURBSCurve (圆角曲线)

### 构造函数

NURBSFilletCurve [<property>:<val>]...

getObject <nurbsset> <index>

### 属性

属性名称	数据类型	说明
<nurbsfilletcurve>.parent1	Integer	存取由 NURBSet 序号指定的第一个父对象曲线
<nurbsfilletcurve>.parent1ID	Integer	存取由 NURBSId 指定的第一个父对象曲线
<nurbsfilletcurve>.parent2	Integer	存取由 NURBSet 序号指定的第二个父对象曲线
<nurbsfilletcurve>.parent2ID	Integer	存取由 NURBSId 指定的第二个父对象曲线
<nurbsfilletcurve>.radius	Float	以当前 3ds max 单位表示圆角弧形的半径。默认设置为 10.0

(续表)

属性名称	数据类型	说明
<nurbsfilletcurve>.flip1	Boolean	当设置为 On 时, 使用第一个父对象曲线的末端; 否则, 使用曲线的始端
<nurbsfilletcurve>.flip2	Boolean	当设置为 On 时, 使用第二个父对象曲线的末端; 否则, 使用曲线的始端
<nurbsfilletcurve>.trim1	Boolean	当设置为 On 时, 将针对圆角曲线修剪第一个父对象曲线; 否则, 不能修剪父曲线
<nurbsfilletcurve>.trim2	Boolean	当设置为 On 时, 将针对圆角曲线修剪第二个父对象曲线; 否则, 不能修剪父曲线
<nurbsfilletcurve>.flipTrim1	Boolean	当设置为 On 时, 将从相反的方向开始修剪第一个父对象曲线
<nurbsfilletcurve>.flipTrim2	Boolean	当设置为 On 时, 将从相反的方向开始修剪第二个父对象曲线

### 8.13.18 NURBSIsoCurve: NURBSCurve (U 向和 V 向等参曲线)

#### 构造函数

NURBSIsoCurve [<property>:<val>]...

getObject <nurbsset> <index>

#### 属性

属性名称	数据类型	说明
<nurbsisocurve>.parent	Integer	存取由 NURBSet 序号指定的父对象曲线
<nurbsisocurve>.parentID	Integer	存取由 NURBSId 指定的父对象曲线
<nurbsisocurve>.dir	Name	指定等参曲线 U/V 轴向: #U 或#V
<nurbsisocurve>.parameter	Float	沿着曲面的 U 轴或 V 轴方向设置等参曲线的位置
<nurbsisocurve>.trim	Boolean	当设置为 On 时, 针对等参曲线修剪曲面
<nurbsisocurve>.flipTrim	Boolean	当设置为 On 时, 翻转修剪的方向
<nurbsisocurve>.seed	point2	设置等参曲线 U/V 方向的种子数

### 8.13.19 NURBSMirrorCurve: NURBSCurve (镜像曲线)

#### 构造函数

NURBSMirrorCurve [<property>:<val>]...

getObject <nurbsset> <index>

#### 属性

属性名称	数据类型	说明
<nurbsmirrorcurve>.parent	Integer	存取由 NURBSet 序号指定的父对象曲线
<nurbsmirrorcurve>.parentID	Integer	存取由 NURBSId 指定的父对象曲线
<nurbsmirrorcurve>.axis	Name	指定镜像依据的轴向: #X、#Y、#Z、#XY、#XZ、#YZ
<nurbsmirrorcurve>.distance	Float	控制镜像与原始曲线之间的距离
<nurbsmirrorcurve>.transform	matrix3	设置镜像曲线的变换矩阵

### 8.13.20 NURBSOffsetCurve: NURBSCurve (偏移曲线)

#### 构造函数

```
NURBSOffsetCurve [<property>:<val>]...
getObject <nurbsset> <index>
```

#### 属性

属性名称	数据类型	说明
<nurbsoffsetcurve>.parent	Integer	存取由 NURBSet 序号指定的父对象曲线
<nurbsoffsetcurve>.parentID	Integer	存取由 NURBSId 指定的父对象曲线
<nurbsoffsetcurve>.distance	Float	控制偏移与原始曲线之间的距离

### 8.13.21 NURBSPointCurve: NURBSCurve (点曲线子对象)

#### 构造函数

```
NURBSPointCurve [<property>:<val>]...
getObject <nurbsset> <index>
```

#### 属性

属性名称	数据类型	说明
<nurbspointcurve>.numPoints	Integer	存取点曲线上的点数量
<nurbspointcurve>.closed	Boolean	当设置为 On 时, 曲线是封闭的; 否则是开放的
<nurbspointcurve>.transform	matrix3	存取 NURBSPointCurve 的转换矩阵。它控制 NURBSSet 里子对象的相对位置

#### 方法

```
close <nurbspointcurve>
getPoint <nurbspointcurve> <index>
setPoint <nurbspointcurve> <index> <nurbs_independent_pt>
refine <nurbspointcurve> <u_param>
```

## 8.13.22 NURBSPointCurveOnSurface: NURBSPointCurve (曲面上的点曲线)

## 构造函数

NURBSPointCurveOnSurface [<property>:<val>]...  
getObject <nurbsset> <index>

## 属性

属性名称	数据类型	说明
<nurbspointcurveonsurface>.parent	Integer	存取由 NURBSet 序号指定的父对象曲线
<nurbspointcurveonsurface>.parentID	Integer	存取由 NURBSId 指定的父对象曲线
<nurbspointcurveonsurface>.trim	Boolean	当设置为 On 时, 将针对曲线修剪曲面
<nurbspointcurveonsurface>.flipTrim	Boolean	当设置为 On 时, 将以相反的方向修剪曲面

## 8.13.23 NURBSProjectNormalCurve: NURBSCurve (法向投射曲线)

## 构造函数

NURBSProjectNormalCurve [<property>:<val>]...  
getObject <nurbsset> <index>

## 属性

属性名称	数据类型	说明
<nurbsprojectnormalcurve>.parent1	Integer	存取由 NURBSet 序号指定的父对象曲面
<nurbsprojectnormalcurve>.parent1ID	Integer	存取由 NURBSId 指定的父对象曲面
<nurbsprojectnormalcurve>.parent2	Integer	存取由 NURBSet 序号指定的父对象曲线
<nurbsprojectnormalcurve>.parent2ID	Integer	存取由 NURBSId 指定的父对象曲线
<nurbsprojectnormalcurve>.trim	Boolean	当设置为 On 时, 将针对曲线修剪曲面
<nurbsprojectnormalcurve>.flipTrim	Boolean	当设置为 On 时, 将以相反的方向修剪曲面
<nurbsprojectnormalcurve>.seed	point2	设置法向投射曲线 U V 向种子数

## 8.13.24 NURBSProjectVectorCurve: NURBSCurve (矢量投射曲线)

## 构造函数

NURBSProjectVectorCurve [<property>:<val>]...  
getObject <nurbsset> <index>

## 属性

属性名称	数据类型	说明
<nurbsprojectvectorcurve>.parent1	Integer	存取由 NURBSet 序号指定的父对象曲面
<nurbsprojectvectorcurve>.parent1ID	Integer	存取由 NURBSId 指定的父对象曲面
<nurbsprojectvectorcurve>.parent2	Integer	存取由 NURBSet 序号指定的父对象曲线
<nurbsprojectvectorcurve>.parent2ID	Integer	存取由 NURBSId 指定的父对象曲线
<nurbsprojectvectorcurve>.trim	Boolean	当设置为 On 时, 将针对曲线修剪曲面
<nurbsprojectvectorcurve>.flipTrim	Boolean	当设置为 On 时, 将以相反的方向修剪曲面
<nurbsprojectvectorcurve>.seed	point2	设置矢量投射曲线 UV 向种子数
<nurbsprojectvectorcurve>.pVec	point3	设置投射向量值

### 8.13.25 NURBSSurfaceEdgeCurve: NURBSCurve (曲面边曲线)

#### 构造函数

```
NURBSSurfaceEdgeCurve [<property>:<val>]...
getObject <nurbsset> <index>
```

#### 属性

属性名称	数据类型	说明
<nurbssurfaceedgecurve>.parent	Integer	存取由 NURBSet 序号指定的父对象曲面
<nurbssurfaceedgecurve>.parentID	Integer	存取由 NURBSId 指定的父对象曲面
<nurbssurfaceedgecurve>.seed	Point2	设置曲面边曲线 UV 向种子数

### 8.13.26 NURBSSurfaceNormalCurve: NURBSCurve (曲面法线曲线)

#### 构造函数

```
NURBSSurfaceNormalCurve [<property>:<val>]...
getObject <nurbsset> <index>
```

#### 属性

属性名称	数据类型	说明
<nurbssurfacenormalcurve>.parent	Integer	存取由 NURBSet 序号指定的父对象曲面
<nurbssurfacenormalcurve>.parentID	Integer	存取由 NURBSId 指定的父对象曲面
<nurbssurfacenormalcurve>.distance	Float	设置父曲线依赖于曲面的偏移量

### 8.13.27 NURBSSurfSurfIntersectionCurve: NURBSCurve (曲面-曲面相交曲线)

#### 构造函数

NURBSSurfSurfIntersectionCurve [<property>:<val>]...

getObject <nurbsset> <index>

#### 属性

属性名称	数据类型	说明
<nurbssurfsurfintersectioncurve>.parent1	Integer	存取由 NURBSet 序号指定的第一个父对象曲面
<nurbssurfsurfintersectioncurve>.parent1ID	Integer	存取由 NURBSId 指定的第一个父对象曲面
<nurbssurfsurfintersectioncurve>.parent2	Integer	存取由 NURBSet 序号指定的第二个父对象曲面
<nurbssurfsurfintersectioncurve>.parent2ID	Integer	存取由 NURBSId 指定的第二个父对象曲面
<nurbssurfsurfintersectioncurve>.trimCurve1	Boolean	当设置为 On 时, 将针对相交曲线修剪第一个父对象曲面; 否则, 不能修剪曲面
<nurbssurfsurfintersectioncurve>.trimCurve2	Boolean	当设置为 On 时, 将针对相交曲线修剪第二个父对象曲面; 否则, 不能修剪曲面
<nurbssurfsurfintersectioncurve>.flipTrim1	Boolean	当设置为 On 时, 将从相反的方向开始修剪第一个父对象曲面
<nurbssurfsurfintersectioncurve>.flipTrim2	Boolean	当设置为 On 时, 将从相反的方向开始修剪第二个父对象曲面
<nurbssurfsurfintersectioncurve>.seed	point2	设置曲面-曲面相交曲线 UV 向种子数

### 8.13.28 NURBSXFormCurve: NURBSCurve (变换曲线)

#### 构造函数

NURBSXFormCurve [<property>:<val>]...

getObject <nurbsset> <index>

#### 属性

属性名称	数据类型	说明
<nurbsxformcurve>.parent	Integer	存取由 NURBSet 序号指定的父对象曲线
<nurbsxformcurve>.parentID	Integer	存取由 NURBSId 指定的父对象曲线
<nurbsxformcurve>.transform	matrix3	存取变换曲线变换的矩阵坐标

### 8.13.29 NURBSSurface: NURBSObject (曲面子对象)

因为本类是所有 NURBS Surface 类的父类对象，所以不能直接创建。所有 NURBS Surface 类都有以下属性。

#### 属性

属性名称	数据类型	说明
<nurbssurface>.renderable	Boolean	指定曲面是否被渲染
<nurbssurface>.flipNormals	Boolean	指定所有曲面法线是否被反转
<nurbssurface>.generateUVs1	Boolean	当设置为 On 时, 允许为第一个 UVW 通道生成 UV 贴图
<nurbssurface>.generateUVs2	Boolean	当设置为 On 时, 允许为第二个 UVW 通道生成 UV 贴图
<nurbssurface>.matID	Integer	返回曲面的材质 ID 码
<nurbssurface>.closedInU	Boolean	只读, 如果返回 True, 表示曲面在 U 方向闭合; 否则, 返回 False
<nurbssurface>.closedInV	Boolean	只读, 如果返回 True, 表示曲面在 V 方向闭合; 否则, 返回 False
<nurbssurface>.uParamerRangeMin	Float	只读, 返回与 NURBSSurface 有关的方法里用到的参数<u_parm>的最小值
<nurbssurface>.uParamerRangeMax	Float	只读, 返回与 NURBSSurface 有关的方法里用到的参数<u_parm>的最大值
<nurbssurface>.vParameterRangeMin	Float	只读, 返回与 NURBSSurface 有关的方法里用到的参数<v_parm>的最小值
<nurbssurface>.vParameterRangeMax	Float	只读, 返回与 NURBSSurface 有关的方法里用到的参数<v_parm>的最大值
<nurbssurface>.textureSurface1		存取贴图通道 1 的纹理曲面
<nurbssurface>.textureSurface2		存取贴图通道 2 的纹理曲面

## 方法

所有 NURBS surface 类都有下面的方法:

1. evalPos <nurbssurface> <u\_param> <v\_param>

返回指定面上指定 U/V 方向参数位置处点的坐标。

2. evalUTangent <nurbssurface> <u\_param> <v\_param>

返回指定面上指定 U/V 方向参数位置处点的 U 方向正切矢量。

3. evalVTangent <nurbssurface> <u\_param> <v\_param>

返回指定面上指定 U/V 方向参数位置处点的 V 方向正切矢量。

4. getTiling <nurbssurface> [channel:<index>]

setTiling <nurbssurface> <ut> <vt> [channel:<index>]

存取曲面在 U/V 方向贴图的铺贴因数。返回或设置一个二元数组: 第一个元素表示 U 方向, 第二个元素表示 V 方向。可选参数 channel:用来选择贴图通道, 默认值为 1。

5. getTilingOffset <nurbssurface> [channel:<index>]

setTilingOffset <nurbssurface> <uo> <vo> [channel:<index>]

存取曲面在指定贴图通道的铺贴偏移。返回或设置一个二元数组: 第一个元素表示 U 方向, 第二个元素表示 V 方向。可选参数 channel:用来选择贴图通道, 默认值为 1。

6. `getGenerateUVs <nurbssurface> <channel_index>`  
`setGenerateUVs <nurbssurface> <channel_index> <boolean>`  
存取是否为指定通道生成 UV 贴图坐标。
7. `getTextureUVs <nurbssurface> <index> [channel:<index>]`  
`setTextureUVs <nurbssurface> <index> <point2> [channel:<index>]`  
存取指定通道的指定序号坐标的纹理 UV 坐标值。坐标序号<index>必须大于等于 1 且小于等于 4。可选参数 channel:默认值为 1。
8. `getTextureSurface <nurbssurface> <channel_index>`  
`setTextureSurface <nurbssurface> <channel_index> <NURBSSurface>`  
存取指定通道的纹理由。
9. `getProdTess <nurbssurface> <tessType_name>`  
`setProdTess <nurbssurface> <tessType_name> <NURBSSurfaceApproximation>`  
`getViewTess <nurbssurface> <tessType_name>`  
`setViewTess <nurbssurface> <tessType_name> <NURBSSurfaceApproximation>`  
为单独曲面存取渲染和视窗的 NURBSSurface 逼近值。参数<tessType\_name>有效值有#surface、#displacement、#curve，分别对应三种网格形式。
10. `clearProdTess <nurbssurface> <tessType_name>`  
`clearViewTess <nurbssurface> <tessType_name>`  
重置指定曲面、指定<tessType\_name>的 NURBSSurface 逼近值。参数<tessType\_name>有效值有#surface、#displacement、#curve，分别对应三种网格形式。

### 8.13.30 NURBS1RailSweepSurface: NURBSSurface (单轨扫描曲面)

#### 构造函数

```
NURBS1RailSweepSurface [<property>:<val>]...
getObject <nurbsset> <index>
```

#### 属性

属性名称	数据类型	说明
<nurbs1railsweepsurface>.rail	Integer	存取由 NURBSet 序号指定的父对象轨道曲线
<nurbs1railsweepsurface>.railID	Integer	存取由 NURBSId 指定的父对象轨道曲线
<nurbs1railsweepsurface>.numCurves	Integer	存取交叉截面曲线的数量
<nurbs1railsweepsurface>.parallel	Boolean	设置是否启用平行扫描
<nurbs1railsweepsurface>.axisTM	matrix3	设置轴向矩阵值

#### 方法

1. `appendCurve <nurbs1railsweepsurface> \`  
`<curve> [flip:<boolean>] [startPoint:<Float>]`  
将指定曲线<curve>添加到<nurbs1railsweepsurface>的截面曲线列表的末尾。如果指定

`flip:True`, 截面曲线被反转。参数 `startPoint` 指定父曲线的起点, 仅在父曲线为闭合曲线时有用。

2. `appendCurveByID <nurbs1railsweepsurface> \<curveID> [flip:<boolean>] [startPoint:<Float>]`

将指定 NURBSID 号`<curveID>`的曲线添加到`<nurbs1railsweepsurface>`的截面曲线列表的末尾。如果指定 `flip:True`, 截面曲线被反转。参数 `startPoint` 指定父曲线的起点, 仅在父曲线为闭合曲线时有用。

3. `getCurve <nurbs1railsweepsurface> <index>`

`setCurve <nurbs1railsweepsurface> <index> <curve>`

存取由`<nurbs1railsweepsurface>`指定序号的截面曲线。

4. `getCurveID <nurbs1railsweepsurface> <index>`

`setCurveByID <nurbs1railsweepsurface> <index> <curveID>`

存取由`<nurbs1railsweepsurface>`指定 ID 码的截面曲线。

5. `getFlip <nurbs1railsweepsurface> <index>`

`setFlip <nurbs1railsweepsurface> <index> <boolean>`

存取指定序号的截面曲线是否被反转。

6. `getCurveStartPoint <nurbs1railsweepsurface> <index>`

`setCurveStartPoint <nurbs1railsweepsurface> <index> <Float>`

存取指定序号的截面曲线在父曲线上的起点。

## 8.13.31 NURBS2RailSweepSurface: NURBSSurface (双轨扫描曲面)

### 构造函数

`NURBS2RailSweepSurface [<property>:<val>]...`

`getObject <nurbsset> <index>`

### 属性

属性名称	数据类型	说明
<code>&lt;nurbs2railsweepsurface&gt;.rail1</code>	Integer	存取由 NURBSet 序号指定的第一个父对象轨道曲线
<code>&lt;nurbs2railsweepsurface&gt;.rail1ID</code>	Integer	存取由 NURBSId 指定的第一个父对象轨道曲线
<code>&lt;nurbs2railsweepsurface&gt;.rail2</code>	Integer	存取由 NURBSId 指定的第二个父对象轨道曲线
<code>&lt;nurbs2railsweepsurface&gt;.rail2ID</code>	Integer	存取由 NURBSId 指定的第二个父对象轨道曲线
<code>&lt;nurbs2railsweepsurface&gt;.numCurves</code>	Integer	存取交叉截面曲线的数量
<code>&lt;nurbs2railsweepsurface&gt;.parallel</code>	Boolean	设置是否启用平行扫描

### 方法

1. `appendCurve <nurbs2railsweepsurface> \<curve> [flip:<boolean>] [startPoint:<Float>]`

将指定曲线`<curve>`添加到`<nurbs2railsweepsurface>`的截面曲线列表的末尾。如果指定

`flip:True`, 截面曲线被反转。参数 `startPoint` 指定父曲线的起点, 仅在父曲线为闭合曲线时有用。

2. `appendCurveByID <nurbs2railsweepsurface> <curveID>\[flip:<boolean>] [startPoint:<Float>]`

将指定 NURBSId 号`<curveID>`的曲线添加到`<nurbs2railsweepsurface>`的截面曲线列表的末尾。如果指定 `flip:True`, 截面曲线被反转。参数 `startPoint` 指定父曲线的起点, 仅在父曲线为闭合曲线时有用。

3. `getCurve <nurbs2railsweepsurface> <index>`

`setCurve <nurbs2railsweepsurface> <index> <curve>`

存取由`<nurbs2railsweepsurface>`指定序号的截面曲线。

4. `getCurveID <nurbs2railsweepsurface> <index>`

`setCurveByID <nurbs2railsweepsurface> <index> <curveID>`

存取由`<nurbs2railsweepsurface>`指定 ID 序号的截面曲线。

5. `getFlip <nurbs2railsweepsurface> <index>`

`setFlip <nurbs2railsweepsurface> <index> <boolean>`

存取指定序号的截面曲线是否被反转。

6. `getCurveStartPoint <nurbs2railsweepsurface> <index>`

`setCurveStartPoint <nurbs2railsweepsurface> <index> <Float>`

存取指定序号的截面曲线在父曲线上的起点。

## 8. 13. 32 NURBSBlendSurface: NURBSSurface (混合曲面)

### 构造函数

`NURBSBlendSurface [<property>:<val>]...`

`getObject <nurbsset> <index>`

### 属性

属性名称	数据类型	说明
<code>&lt;nurbsblendsurface&gt;.parent1</code>	Integer	存取由 NURBSet 序号指定的第一个父对象曲面
<code>&lt;nurbsblendsurface&gt;.parent1ID</code>	Integer	存取由 NURBSId 指定的第一个父对象曲面
<code>&lt;nurbsblendsurface&gt;.parent2</code>	Integer	存取由 NURBSet 序号指定的第一个父对象曲面
<code>&lt;nurbsblendsurface&gt;.parent2ID</code>	Integer	存取由 NURBSId 指定的第一个父对象曲面
<code>&lt;nurbsblendsurface&gt;.edge1</code>	Integer	指定第一个父对象曲面用于混合的边。下面是其属性值: 1: U 方向底边 2: U 方向顶边 3: V 方向底边 4: V 方向顶边
<code>&lt;nurbsblendsurface&gt;.edge2</code>	Integer	指定第二个父对象曲面用于混合的边

(续表)

属性名称	数据类型	说明
<nurbsblendsurface>.flip1 <nurbsblendsurface>.flip2	Boolean	设置是否反转混合曲面法线的方向
<nurbsblendsurface>.tension1 <nurbsblendsurface>.tension2	Float	控制单击的两曲面边上的张力
<nurbsblendsurface>.curveStartPoint1 <nurbsblendsurface>.curveStartPoint2	Float	调整混合曲面两边的起始点位置

### 方法

1. getParent <nurbsblendsurface> <index>  
    setParent <nurbsblendsurface> <index> <curve>  
    存取由 NURBSSet 指定的混合曲面的父对象曲面。
2. getParentID <nurbsblendsurface> <index>  
    setParentID <nurbsblendsurface> <index> <curveID>  
    存取由 NURBSID 指定的混合曲面的父对象曲面。
3. getEdge <nurbsblendsurface> <index>  
    setEdge <nurbsblendsurface> <index> <edge>  
    存取父对象曲面上用于混合的边。

## 8.13.33 NURBSCapSurface: NURBSSurface (封口曲面)

### 构造函数

NURBSCapSurface [<property>:<val>]...  
getObject <nurbsset> <index>

### 属性

属性名称	数据类型	说明
<nurbscapsurface>.parent	Integer	存取由 NURBSet 序号指定的父对象曲面
<nurbscapsurface>.parentID	Integer	存取由 NURBSID 指定的父对象曲面
<nurbscapsurface>.edge	Integer	指定父对象曲面封口的边。下面是其属性值： 1: U 方向底边 2: U 方向顶边 3: V 方向底边 4: V 方向顶边
<nurbscapsurface>.curveStartPoint	Integer	调整封口曲面的起始点位置

## 8.13.34 NURBSCVSurface: NURBSSurface (CV 曲面子对象)

## 构造函数

```
NURBSCVSurface [<property>:<val>]...
```

```
getObject <nurbsset> <index>
```

## 属性

属性名称	数据类型	说明
<nurbscvsurface>.uOrder <nurbscvsurface>.vOrder	Integer	曲面在 U/V 方向的级数
<nurbscvsurface>.numUKnots <nurbscvsurface>.numVKnots	Integer	曲面在 U/V 方向的结点数。如果改变该属性的值，原来的结点数据会丢失
<nurbscvsurface>.numCVs	point2	曲面在 U/V 方向的控制点 (CV) 数。如果改变该属性的值，原来的结点数据会丢失
<nurbscvsurface>.transform	matrix3	指定 NURBSCVSurface 的转换矩阵，控制对象内部各子对象的相对位置
<nurbscvsurface>.uEdgesOverlap <nurbscvsurface>.vEdgesOverlap	Boolean	只读，如果指定曲面的边在 U/V 方向重叠，即使该曲面不闭合，返回 True
<nurbscvsurface>.autoParam		本属性与 CV Surface 用户界面里的 Automatic Reparam 选项对应，有效值有：#notAutomatic、#autoCentripetal、#autoUniform、#notAutomatic、#autoCentripetal 和#autoUniform，默认值为：#notAutomatic
<nurbscvsurface>.rigid	Boolean	如果指定曲面为 rigid，返回 True；否则，返回 False。对 rigid 曲面仅能进行下面的修改：在 Surface 子对象级进行转换操作。rigid 曲面上的结点或控制点不能被移动，也不能改变控制点的数目。rigid 曲面可以减少 NURBS 对象占用的内存，加快系统运行速度

## 方法

1. closeU <nurbscvsurface>

封闭 U 方向的曲面。

2. closeV <nurbscvsurface>

封闭 V 方向的曲面。

3. getUKnot <nurbscvsurface> <u\_index>

返回 U 方向指定序号的结点，结点序号从 1 开始。

4. setUKnot <nurbscvsurface> <u\_index> <Float>

将 U 方向指定序号的结点设为指定值。

5. getVKnot <nurbscvsurface> <v\_index>

返回 V 方向指定序号的结点，结点序号从 1 开始。

6. setVKnot <nurbscvsurface> <v\_index> <Float>

将 V 方向指定序号的结点设为指定值。

7. getCV <nurbscvsurface> <u\_index> <v\_index>

返回一个 NURBSControlVertex 值，表示指定 U/V 方向指定序号的控制点。

8. setCV <nurbscvsurface> <u\_index> <v\_index> <NURBSControlVertex>

将指定 U/V 方向序号的控制点设为指定 NURBSControlVertex 值，。

9. refineU <nurbscvsurface> <v\_param>

在指定曲面的 U 方向指定位置插入一行新控制点。

10. refineV <nurbscvsurface> <u\_param>

在指定曲面的 V 方向指定位置插入一列新控制点。

11. refine <nurbscvsurface> <u\_param> <v\_param>

在指定曲面的 U/V 方向指定位置分别插入一行、一列新控制点。

12. reparameterize <nurbscvsurface> (#centripetal / #uniform)

按指定方式重新计算曲面参数。

### 8.13.35 NURBSExtrudeSurface: NURBSSurface (挤出曲面)

#### 构造函数

NURBSExtrudeSurface [<property>:<val>]...

getObject <nurbsset> <index>

#### 属性

属性名称	数据类型	说明
<nurbsextrudesurface>.parent	Integer	存取由 NURBSet 序号指定的父对象曲线
<nurbsextrudesurface>.parentID	Integer	存取由 NURBSId 指定的父对象曲线
<nurbsextrudesurface>.distance	Float	曲面从父曲线挤出的距离，采用当前 3ds max 单位。 此参数可设置动画
<nurbsextrudesurface>.axisTM	matrix3	指定挤出的轴
<nurbsextrudesurface>.curveStartPoint	Float	调整挤出曲面的起始点位置

### 8.13.36 NURBSFilletSurface: NURBSSurface (圆角曲面)

#### 构造函数

NURBSFilletSurface [<property>:<val>]...

getObject <nurbsset> <index>

#### 属性

<nurbsfilletsurface>.cubic: Boolean

本属性如果为 True，半径为线性的；如果为 False，将半径看作是立方体功能，允许其在父曲面几何体基础上进行更改。

## 方法

1. getParent <nurbsfilletsurface> <index>  
    setParent <nurbsfilletsurface> <index> <curve>  
存取由 NURBSet 序号指定的父对象曲面。
2. setParentID <nurbsfilletsurface> <index> <curveID>  
    getParentID <nurbsfilletsurface> <index>  
存取由 NURBSId 指定的父对象曲面。
3. getSeed <nurbsfilletsurface> <index>  
    setSeed <nurbsfilletsurface> <index> <point2>  
存取父对象曲面 U/V 向的种子值。
4. getRadius <nurbsfilletsurface> <index>  
    setRadius <nurbsfilletsurface> <index> <Float>  
存取圆角曲面的半径。
5. getTrimSurface <nurbsfilletsurface> <index>  
    setTrimSurface <nurbsfilletsurface> <index> <bool>  
存取是否修剪父对象曲面。
6. getFlipTrim <nurbsfilletsurface> <index>  
    setFlipTrim <nurbsfilletsurface> <index> <bool>  
存取修剪父对象曲面的方向。

## 8. 13. 37 NURBSLatheSurface: NURBSSurface (车削曲面)

### 构造函数

NURBSLatheSurface [<property>:<val>]...  
getObject <nurbsset> <index>

### 属性

属性名称	数据类型	说明
<nurbslatthesurface>.parent	Integer	存取由 NURBSet 序号指定的父对象曲线
<nurbslatthesurface>.parentID	Integer	存取由 NURBSId 指定的父对象曲线
<nurbslatthesurface>.axisTM	matrix3	指定车削的轴
<nurbslatthesurface>.sweep	Float	设置旋转的角度
<nurbsextrudesurface>.curveStartPoint	Float	调整挤出曲面的起始点位置

## 8. 13. 38 NURBSMirrorSurface: NURBSSurface (镜像曲面)

### 构造函数

NURBSMirrorSurface [<property>:<val>]...  
getObject <nurbsset> <index>

### 属性

属性名称	数据类型	说明
<nurbsmirrorsurface>.parent	Integer	存取由 NURBSet 序号指定的父对象曲线
<nurbsmirrorsurface>.parentID	Integer	存取由 NURBSId 指定的父对象曲线
<nurbsmirrorsurface>.axis		指定镜像依据的轴向: #X、#Y、#Z、#XY、#XZ、#YZ
<nurbsmirrorsurface>.distance	Float	控制镜像与原始曲面之间的距离。此参数可设置动画
<nurbsmirrorsurface>.transform	matrix3	设置转换的矩阵值

### 8.13.39 NURBSMultiCurveTrimSurface: NURBSSurface (多重曲线修剪曲面)

#### 构造函数

```
NURBSMultiCurveTrimSurface [<property>:<val>]...
getObject <nurbsset> <index>
```

#### 属性

属性名称	数据类型	说明
<nurbsmulticurvetrimsurface>.surfaceParent	Integer	存取由 NURBSet 序号指定的父对象曲线
<nurbsmulticurvetrimsurface>.surfaceParentID	Integer	存取由 NURBSId 指定的父对象曲线
<nurbsmulticurvetrimsurface>.numCurves	Integer	存取组成修剪环的曲线数量
<nurbsmulticurvetrimsurface>.flipTrim	Boolean	当设置为 On 时, 将以相反的方向修剪曲面

#### 方法

1. getParent <nurbsmulticurvetrimsurface> <index>  
setParent <nurbsmulticurvetrimsurface> <index> <curve>  
存取由 NURBSet 序号指定的父对象曲线。
2. getParentID <nurbsmulticurvetrimsurface> <index>  
setParentID <nurbsmulticurvetrimsurface> <index> <curveID>  
存取由 NURBSId 指定的父对象曲线。
3. appendCurve <nurbsmulticurvetrimsurface> <curve>  
将指定曲线<curve>添加到<nurbsmulticurvetrimsurface>trimloop 列表的末尾。
4. appendCurveByID <nurbsmulticurvetrimsurface> <curveID>  
将指定 ID <curveID>的曲线添加到<nurbsmulticurvetrimsurface>trimloop 列表的末尾。

### 8.13.40 NURBSNblendSurface: NURBSSurface (混合曲面)

#### 构造函数

```
NURBSNblendSurface [<property>:<val>]...
getObject <nurbsset> <index>
```

## 属性

除了通用属性外，NURBSNBlendSurface 没有额外的属性。

## 方法

1. `getParent <nurbsnblendsurface> <index>`

返回<nurbsnblendsurface>里指定序号的曲线或曲面。

2. `setParent <nurbsnblendsurface> <index> <curve>`

将指定序号的曲线或曲面设为指定曲线<curve>。

3. `getParentID <nurbsnblendsurface> <index>`

返回指定序号的曲线或曲面的 NURBSId。

4. `setParentID <nurbsnblendsurface> <index> <curveID>`

将指定序号的曲线或曲面的设置为指定 NURBSId 的曲线。

5. `getEdge <nurbsnblendsurface> <index>`

`setEdge <nurbsnblendsurface> <index> <edge>`

存取指定序号父曲面的哪一条边被用于混合。参数<edge>的有效值及含义说明如下：

1: U 方向底边

2: U 方向顶边

3: V 方向底边

4: V 方向顶边

## 8. 13. 41 NURBSOffsetSurface: NURBSSurface (偏移曲面)

### 构造函数

`NURBSOffsetSurface [<property>:<val>]...`

`getObject <nurbsset> <index>`

### 属性

属性名称	数据类型	说明
<code>&lt;nurbsoffsetsurface&gt;.parent</code>	Integer	存取由 NURBSet 序号指定的父对象曲面
<code>&lt;nurbsoffsetsurface&gt;.parentID</code>	Integer	存取由 NURBSId 指定的父对象曲面
<code>&lt;nurbsoffsetsurface&gt;.distance</code>	Float	以 3ds max 单位表示父曲面和偏移曲面之间的距离

## 8. 13. 42 NURBSPointSurface: NURBSSurface (点曲面子对象)

### 构造函数

`NURBSPointSurface [<property>:<val>]...`

`getObject <nurbsset> <index>`

## 属性

属性名称	数据类型	说明
<nurbspointsurface>.numPoints	point2	存取点曲面 U/V 方向上点的数量
<nurbspointsurface>.transform	matrix3	存取 NURBSPointSurface 的转换矩阵
<nurbspointsurface>.closedU	Boolean	当设置为 On 时, 表示 U 方向上的曲面是闭合的; 否则是开放的
<nurbspointsurface>.closedV	Boolean	当设置为 On 时, 表示 V 方向上的曲面是闭合的; 否则是开放的

## 方法

1. closeU <nurbspointsurface>

封闭 U 方向的曲线。

2. closeV <nurbspointsurface>

封闭 V 方向的曲线。

3. getPoint <nurbspointsurface> <u\_index> <v\_index>

返回一个 NURBSIndependentPoint 值, 表示曲面上 U/V 方向指定序号的点。点序号从 1 开始。

4. setPoint <nurbspointsurface> <u\_index>\<v\_index> <NURBSIndependentPoint>

将曲面上 U/V 方向指定序号的点设置为指定 NURBSIndependentPoint 值。

5. refineU <nurbspointsurface> <v\_param>

在指定曲面的 U 方向指定位置插入一行新点。

6. refineV <nurbspointsurface> <u\_param>

在指定曲面的 V 方向指定位置插入一列新点。

7. refine <nurbspointsurface> <u\_param> <v\_param>

在指定曲面的 U/V 方向指定位置分别插入一行、一列新点。

## 8.13.43 NURBSRuledSurface: NURBSSurface (规则曲面)

### 构造函数

NURBSRuledSurface [<property>:<val>]...

getObject <nurbsset> <index>

### 属性

属性名称	数据类型	说明
<nurbsruledsurface>.parent1	Integer	存取由 NURBSet 序号指定的第一个父对象曲线
<nurbsruledsurface>.parent1ID	Integer	存取由 NURBSId 指定的第一个父对象曲线
<nurbsruledsurface>.parent2	Integer	存取由 NURBSet 序号指定的第二个父对象曲线
<nurbsruledsurface>.parent2ID	Integer	存取由 NURBSId 指定的第二个父对象曲线
<nurbsruledsurface>.flip1 <nurbsruledsurface>.flip2	Boolean	控制创建规则曲面时与父对象曲线匹配的方向
<nurbsruledsurface>.curveStartPoint1 <nurbsruledsurface>.curveStartPoint2	Float	调整规则曲面的起始点位置。该属性只有在曲线是闭合的情况下才可适用

## 8.13.44 NURBSULoftSurface: NURBSSurface (U 向放样曲面)

## 构造函数

```
NURBSULoftSurface [<property>:<val>]...
getObject <nurbsset> <index>
```

## 属性

<nurbsuloftsurface>.numCurves: Integer  
存取 U 向放样曲线的数量。

## 方法

1. getCurve <nurbsuloftsurface> <index>  
返回指定序号 U 向放样曲线的 NURBSSet 序号。曲线序号从 1 开始。

2. setCurve <nurbsuloftsurface> <index> <curve>  
将指定序号 U 向放样曲线设置为指定的曲线。

3. getCurveID <nurbsuloftsurface> <index>  
返回指定序号 U 向放样曲线的 NURBSID 号。曲线序号从 1 开始。

4. setCurveByID <nurbsuloftsurface> <index> <curveID>  
将指定序号 U 向放样曲线设置为指定 NURBSId 的曲线。

5. appendCurve <nurbsuloftsurface> <curve>|
 [flip:<boolean>] [startPoint:<Float>] \
 [tension:<Float>] [useTangent:<boolean>] \
 [flipTangent:<boolean>]

将指定曲线<curve>添加到 U 向放样曲线列表的末尾。如果指定参数 flip:True，曲线的起点和终点被反转。参数 startPoint:指定了曲线的起点，仅在曲线为闭合情况下有用。参数 tension:指定了曲线的张力。如果指定参数 useTangent:True，曲线处在曲面上，指定本参数可以将 Loft 曲线平滑的混合到指定曲面上。如果指定参数 flipTangent:True，曲线的切线矢量被反转。

6. appendCurveByID <nurbsuloftsurface> <curveID>|
 [flip:<boolean>] [startPoint:<Float>] \
 [tension:<Float>] [useTangent:<boolean>] \
 [flipTangent:<boolean>]

将指定 ID<curveID>的曲线添加到 U 向放样曲面列表的末尾。

7. getFlip <nurbsuloftsurface> <index>
 setFlip <nurbsuloftsurface> <index> <boolean>  
存取指定序号 U 向放样曲线的反转状态，用来控制放样时的扭曲。

### 8.13.45 NURBSUVLoftSurface: NURBSSurface (UV 放样曲面)

#### 构造函数

NURBSUVLoftSurface [<property>:<val>]...  
getObject <nurbsset> <index>

#### 属性

<nurbsuvloftsurface>.numUCurves: Integer  
<nurbsuvloftsurface>.numVCurves: Integer  
存取 U/V 方向放样曲线的数量。

#### 方法

1. getUCurve <nurbsuvloftsurface> <index>  
返回指定序号 UV 放样曲线在 U 方向的 NURBSSet 序号。曲线序号从 1 开始。
2. setUCurve <nurbsuvloftsurface> <index> <curve>  
将指定序号 UV 放样曲线的 U 方向设置为指定的曲线。
3. getUCurveID <nurbsuvloftsurface> <index>  
返回指定序号 UV 放样曲线在 U 方向的 NURBSId。曲线序号从 1 开始。
4. setUCurveByID <nurbsuvloftsurface> <index> <curveID>  
将指定序号 UV 放样曲线的 U 方向设置为指定<curveID>的曲线。
5. getVCurve <nurbsuvloftsurface> <index>  
返回指定序号 UV 放样曲线在 V 方向的 NURBSSet 序号。曲线序号从 1 开始。
6. setVCurve <nurbsuvloftsurface> <index> <curve>  
将指定序号 UV 放样曲线的 V 方向设置为指定的曲线。
7. getVCurveID <nurbsuvloftsurface> <index>  
返回指定序号 UV 放样曲线在 V 方向的 NURBSId。曲线序号从 1 开始。
8. setVCurveByID <nurbsuvloftsurface> <index> <curveID>  
将指定序号 UV 放样曲线的 V 方向设置为指定<curveID>的曲线。
9. appendUCurve <nurbsuvloftsurface> <curve>  
将指定曲线<curve>添加到 U 方向 UV 放样曲线列表的末尾。
10. appendUCurveByID <nurbsuvloftsurface> <curveID>  
将指定 ID 的<curveID>曲线添加到 U 方向 UV 放样曲线列表的末尾。
11. appendVCurve <nurbsuvloftsurface> <curve>  
将指定曲线<curve>添加到 V 方向 UV 放样曲线列表的末尾。
12. appendVCurveByID <nurbsuvloftsurface> <curveID>  
将指定 ID 的<curveID>曲线添加到 V 方向 UV 放样曲线列表的末尾。

## 8.13.46 NURBSXFormSurface: NURBSSurface (变换曲面)

## 构造函数

```
NURBSXFormSurface [<property>:<val>]...
getObject <nurbsset> <index>
```

## 属性

属性名称	数据类型	说明
<nurbsxformsurface>.parent	Integer	存取由 NURBSet 序号指定的父对象曲线面
<nurbsxformsurface>.parentID	Integer	存取由 NURBSId 指定的父对象曲面
<nurbsxformsurface>.transform	matrix3	设置变换的矩阵值

## 8.13.47 NURBSTexturePoint: NURBSObject (纹理曲面)

## 构造函数

```
NURBSTexturePoint <point2> [indices:<point2>]
getPoint <NURBSTextureSurface> <u_index> <v_index>
```

## 属性

<nurbstexturepoint>.pos: point2

设置纹理角点的 UV 坐标值。

## 方法

```
setIndices <nurbstexturepoint> <u_index> <v_index>
为纹理曲面上指定纹理角点设置索引。
```

## 8.13.48 NURBSDisplay: Value

## 构造函数

```
NURBSDisplay [<property>:<val>]...
<nurbsset>.display
```

## 属性

属性名称	数据类型	说明
<nurbsdisplay>.displayCurves	Boolean	如果曲线显示在视窗里, 返回 True; 否则, 返回 False
<nurbsdisplay>.displaySurfaces	Boolean	如果曲面显示在视窗里, 返回 True; 否则, 返回 False
<nurbsdisplay>.displayLattices	Boolean	如果晶格显示在视窗里, 返回 True; 否则, 返回 False

(续表)

属性名称	数据类型	说明
<nurbsdisplay>.displaySurfCVLattices	Boolean	如果曲面的 CV 晶格显示在视窗里, 返回 True; 否则, 返回 False
<nurbsdisplay>.displayCurveCVLattices	Boolean	如果曲线的 CV 晶格显示在视窗里, 返回 True; 否则, 返回 False
<nurbsdisplay>.displayDependents	Boolean	如果非独立的子对象显示在视窗里, 返回 True; 否则, 返回 False
<nurbsdisplay>.displayTrimming	Boolean	如果曲面裁剪显示在视窗里, 返回 True; 否则, 返回 False
<nurbsdisplay>.degradeOnMove	Boolean	如果曲面在进行转换运算时可以退化, 返回 True; 否则, 返回 False
<nurbsdisplay>.displayShadedLattice	Boolean	如果在着色视窗里, NURBS 曲面显示为着色的晶格, 而在线框视窗里, 显示为未着色的晶格, 返回 True; 如果在着色视窗里, NURBS 曲面显示为棋盘状面片, 而在线框视窗里, 显示为等参曲线或线框面片, 返回 False

### 8.13.49 NURBSSelection: Value

一个NURBSSelection值为一个虚拟数组, 代表一个NURBS场景对象的NURBS子对象集合。这样, 可以通过索引存取指定子对象, 还可以对这个集合里的子对象执行映射函数。NURBSSelection数组是动态的, 当其对应的NURBS场景对象包含的NURBS子对象发生变化时, 它也会跟着变化。

#### 构造函数

```

<node>.selectedCurveCVs
<node>.selectedCurves
<node>.selectedImports
<node>.selectedPoints
<node>.selectedSurfaces
<node>.selectedSurfCVs
<node>.curveCVs      只读
<node>.curves        只读
<node>.imports       只读
<node>.surfaces      只读
<node>.surfCVs       只读
<node>.points         只读

```

#### 属性

1. <nurbsselection>.count: Integer, 只读

返回NURBSSelection数组里子对象的数目。

2. <nurbsselection>.selSetNames: name 类数组, 只读

返回一个元素数据类型为 Name 的数组, 表示所有与 NURBSSelection 值对对应象同级别的命名选择集的名称。

下面的属性用于单个的选择 (如\$foo.curveCVs[n]):

3. <nurbsselection>.index: Integer, 只读

返回 NURBS 对象里指定子对象的序号。如:

```
$foo.selectedcurveCVs[2].index
```

会返回当前选择子对象里第二条 CV 曲线的序号。

### 操作符

1. <node>.selectedCurveCVs = (<array> | <bitarray>)

<node>.selectedCurves = (<array> | <bitarray>)

<node>.selectedImports = (<array> | <bitarray>)

<node>.selectedPoints = (<array> | <bitarray>)

<node>.selectedSurfaces = (<array> | <bitarray>)

<node>.selectedSurfCVs = (<array> | <bitarray>)

以上均可以选择指定子对象。

2. <nurbsselection>[<Integer>]

将指定序号子对象作为只有一个元素的 NURBSSelection 值返回。序号从 1 开始。

3. <nurbsselection>[(<Integer\_array> | <bitarray>)]

返回一个 NURBSSelection 值, 表示指定序号的子对象集合。

4. <nurbsselection>[(<#name> | <string>)]

返回指定子对象级命名选择集。

5. <nurbsselection>[(<#name> | <string>)] = \

(<nurbsselection> | <Integer\_array> | <bitarray>)

将子对象级命名选择集设置为指定边。

### 方法

1. move <nurbsselection> <point3>

将 NURBSSelection 的子对象移动指定距离。

2. rotate <nurbsselection> <quat\_or\_other\_rotation\_form>

将 NURBSSelection 的子对象旋转指定角度。

3. scale <nurbsselection> <point3>

将 NURBSSelection 的子对象进行缩放。

4. select <nurbsselection>

选择 NURBSSelection 的子对象。

5. deselect <nurbsselection>

取消对 NURBSSelection 的子对象的选择。

6. delete <nurbsselection>  
删除 NURBSSelection 的子对象。
7. append <nurbsselection> (<nurbsselection> | <Integer>)  
将指定子对象添加为 NURBSSelection 的子对象。
8. findItem <nurbsselection> (<nurbsselection[<Integer>]> | <Integer>)  
如果在第一个<nurbsselection>里有找到与第二个<nurbsselection>匹配的子对象，返回其序号；否则，返回 0。  
如：

```
$foo.surfCVs[#baz]
move $foo.curves[#bar] [0,0,10] --将 bar 曲线集在 z 轴方向移动 10
move $foo.selectedSurfCVs [0,0,10]
coordsys local scale $foo.surfaces[#baz] [2,2,2]
select $foo.curveCVs[#(1,2,3,4,5)] --选择指定 curveCVs
deselect $foo.curves --取消对所有曲线的选择
delete $foo.points#[{1..31, 40..50}] --删除指定点
$foo.curveCVs[#fred] = $foo.selectedCurveCVs
```

### 8.13.50 NURBSSet: Value

本类用来创建 NURBS 对象，或从现有的 NURBS 对象获取数据。NURBSSet 是其他对象的一个容器。

本类包含各种从点、曲线、曲面衍生的 NURBS 对象，此外，它还有两个“融合”数组：一个数组用来融合曲线，另一个用来融合曲面。这样曲线或曲面里的控制点被融合在一起，当移动一个曲线或曲面时，另外的曲线或曲面会跟着移动。本类还有一些拟合属性，用来控制对象的方格显示在视窗和渲染时为三角形面片。

NURBSSet 值可执行映射操作。

#### 构造函数

##### 1. NURBSSet [<property>:<val>]...

所有对象属性都可以在构造函数里用可选参数的形式进行设置。

##### 2. getNURBSSet <node> [#relational]

本函数用来获取与指定 NURBS 场景对象对应的 NURBSSet 值。如果可选参数#relational 没有被指定，返回的 NURBSSet 值仅包含独立的曲面和曲线，所有非独立的对象将被分解成对应的独立对象。如果一个<node>对象有两个 CV 曲面和一个非独立的混和曲面，调用本函数时不指定可选参数#relational 会返回三个 CV 曲面，原来混和曲面的数据会丢失。

如果调用本函数时可选参数#relational 被指定，就会返回两个 CV 曲面和一个 NURBS 混和曲面，而且返回的 NURBSSet 值是“活动”的：对返回 NURBSSet 值里的元素进行修改会直接反映到源场景对象上。

## 属性

<nurbsset>.numObjects: Integer, 只读

<nurbsset>.count: Integer, 只读

NURBSSet 值内部对象数目。

下面属性用来控制表面网格。共有两套网格属性：一套控制视窗网格，一套控制渲染网格。这些属性对应 NURBS 对象在 Modify 面板里的 Surface Approximation 各控件：

```

<nurbsset>.viewConfig: #isoOnly, #isoAndMesh, #meshOnly
<nurbsset>.viewIsoULines: Integer
<nurbsset>.viewIsoVLines: Integer
<nurbsset>.viewMeshUSteps: Integer
<nurbsset>.viewMeshVSteps: Integer
<nurbsset>.viewMeshApproxType:
    #parametric, #spatial, #curvature, : #regular, #spatialAndCurvature
<nurbsset>.viewSpacialEdge: Float
<nurbsset>.viewCurvatureAngle: Float
<nurbsset>.viewCurvatureDistance: Float
<nurbsset>.viewViewDependent: Boolean
<nurbsset>.renderConfig:
    #isoOnly, #isoAndMesh, #meshOnly, : #regular, #spatialAndCurvature
<nurbsset>.renderIsoULines: Integer
<nurbsset>.renderIsoVLines: Integer
<nurbsset>.renderMeshUSteps: Integer
<nurbsset>.renderMeshVSteps: Integer
<nurbsset>.renderMeshApproxType:
    #parametric, #spatial, #curvature, : #regular, #spatialAndCurvature
<nurbsset>.renderSpacialEdge: Float
<nurbsset>.renderCurvatureAngle: Float
<nurbsset>.renderCurvatureDistance: Float
<nurbsset>.renderViewDependent: Boolean
<nurbsset>.display: NURBSDisplay
控制 NURBS 对象的子对象在视窗里的可见性。
<nurbsset>.viewApproximation: NURBSSurfaceApproximation
<nurbsset>.renderApproximation: NURBSSurfaceApproximation
返回 NURBS 对象的所有拟合设置。
<nurbsset>.merge: Float

```

## 操作符

<nurbsset>[<index>]

<nurbsset>[<index>] = <nurbsobj>

NURBSSet里的子对象可以象MAXScript数组索引一样存取NURBSSet对象的子对象。这和下面讲到的getObject()和setObject()方法是一样的。

### 方法

1. appendObject <nurbsset> <nurbsobj>

将指定NURBS对象(NURBSObject的子类及子类的子类、所有点、曲线、曲面)作为NURBSSet的一个新子对象。

2. setObject <nurbsset> <index> <nurbsobj>

将指定NURBS对象<nurbsobj>作为NURBSSet的指定序号<index>的子对象，该序号原来的子对象被取代。

3. getObject <nurbsset> (<index> | <NURBSSelection>)

返回NURBSSet的指定序号<index>的子对象或由<NURBSSelection>指定的子对象集。如果<NURBSSelection>没有指定任何子对象，返回undefined。

4. removeObject <nurbsset> <index>

将指定序号<index>的子对象从NURBSSet移走。该序号以上的子对象的序号减1。

5. disconnect <nurbsset>

取消指定NURBSSet与场景对象之间的关联，将它转换为一个非关联的集合。

6. deleteObjects <nurbsset>

删除NURBSSet里所有子对象，并释放它所占用的内存。

7. getProdTess <nurbsset> <tessType\_name>

setProdTess <nurbsset> <tessType\_name> <NURBSSurfaceApproximation>

getViewTess <nurbsset> <tessType\_name>

setViewTess <nurbsset> <tessType\_name> <NURBSSurfaceApproximation>

为单独曲面存取渲染和视窗的NURBSSurface逼近值。参数<tessType\_name>有效值有#surface、#displacement、#curve，分别对应三种网格形式。

8. clearProdTess <nurbsset> <tessType\_name>

clearViewTess <nurbsset> <tessType\_name>

重置指定曲面、指定<tessType\_name>的NURBSSurface逼近值。参数<tessType\_name>有效值有#surface、#displacement、#curve，分别对应三种网格形式。

### 8.13.51 NURBSSurfaceApproximation: Value

本类的实例包含一个完整的曲面拟合设置，可以用来赋给NURBS对象的拟合设置。

可以对已有的场景对象调用函数setViewApproximation()和setRenderApproximation()，或使用NURBSSet值的.viewApproximation和.renderApproximation属性来存取NURBS对象的拟合设置。

### 构造函数

1. NURBSSurfaceApproximation [<property>:<val>]...

对象的所有属性均可在构造函数里用可选参数的形式进行设定。

2. <nurbsset>.viewApproximation
3. <nurbsset>.renderApproximation

#### 属性

1. <nurbssurfaceapproximation>.config: #isoOnly, #isoAndMesh, #meshOnly

本属性控制在交互窗口和 Scanline Renderer 窗口的显示内容。

如果指定参数#isoOnly，仅显示等参线，等参线和轮廓线类似。这些线条表明该处的 NURBS 曲面有一个不变的 U 值或 V 值或两者都是。等参线和面片线框相比，不如后者密集，且更直观。

如果指定参数#isoAndMesh，显示等参线和面片。此时，在线框视窗里用等参线来显示曲面，在着色视窗里显示着色的曲面。

如果指定参数#meshOnly，仅显示面片。此时，在线框视窗里用曲面来显示线框相比，在着色视窗里显示着色的曲面。在线框视窗里，采用本参数可以观察视窗里用到的曲线拟合。

2. <nurbssurfaceapproximation>.isoULines: Integer

本属性用在.config 属性为#isoOnly 的情况下，表示在 U 方向额外的等参线。

3. <nurbssurfaceapproximation>.isoVLines: Integer

本属性用在.config 属性为#isoOnly 的情况下，表示在 V 方向额外的等参线。

4. <nurbssurfaceapproximation>.meshUSteps: Integer

本属性为细分设置参数，表示 U 方向细分数目。

5. <nurbssurfaceapproximation>.meshVSteps: Integer

表示 V 方向细分数目。

6. <nurbssurfaceapproximation>.meshApproxType :

#parametric, #spatial, #curvature,: #regular, #spatialAndCurvature

设置曲面细分类型：

- ◆ #parametric 表示执行参数细分。采用属性.meshUSteps 和.meshVSteps 的设置进行曲面细分。总共有.meshUSteps 乘以.meshVSteps 个四边形，每个四边形被劈成两个三角形；
- ◆ #spatial 表示执行空间细分。使用属性.spacialEdge 作为其参数；
- ◆ #curvature 表示执行 view\_dependent 细分，使用属性.curvatureAngle 和.curvatureDistance 的值作为其参数；
- ◆ #regular 表示对曲面执行固定、规则的细分；
- ◆ #spatialAndCurvature 表示执行#spatial 和#curvature 两种细分。

7. <nurbssurfaceapproximation>.spacialEdge: Float

当.meshApproxType 属性为#spatial 时，指定空间细分的边长。在 view\_dependent 细分方式下，单位为像素；否则为一个边界框对角长度的百分数。

8. <nurbssurfaceapproximation>.curvatureAngle: Float

当.meshApproxType 属性为#curvature 时, 如果指定 0.0, 属性被忽略。如果大于 0.0, 表示细分后的相邻曲面的法线角度差不能大于该值。单位为弧度。

9. <nurbssurfaceapproximation>.curvatureDistance: Float

当.meshApproxType 属性为#curvature 时, 如果指定 0.0, 属性被忽略。如果大于 0.0, 表示细分后的曲面顶点之间的距离不能大于该值。

10. <nurbssurfaceapproximation>.viewDependent: Boolean

指定是否为 view\_dependent 细分, 如果为 True, 对象离摄影机越远, 细分精度越差; 如果为 False, 细分精度与对象离摄影机距离无关。

# 第9章 Editable\_Mesh、SplineShape、Patch 和 Editable\_Poly

## 9.1 Editable\_Mesh 和 TriMesh: GeometryClass (可编辑网格和三角网格)

如果将对象的修改器堆栈进行塌陷操作，生成的结果对象的类名就是 Editable\_Mesh。在 MAXScript 里许多 Mesh 操作仅适用于 Editable\_Mesh 类场景对象。

3ds max 里还有一种 Mesh 类 TriMesh，是专为低级的 3ds max SDK Mesh 类（这种类在 3ds max 里大量地用于网格基本对象和修改器）设计的一种数据类型。设计 TriMesh 类的基本目的是为脚本参数对象插件在几何对象内部存取、创建 Mesh 对象。

在本节中讲述的属性和方法如果同时适用于 Editable\_Mesh 和 TriMesh 类，操作对象被标记为`<mesh>`，如果仅适用于 TriMesh 类，操作对象被标记为`<trimesh>`。

以下几点是读者在使用 Mesh 类对象时要特别注意的：

- ◆ 在 3ds max 视窗里，只有方法 `update()` 才可以更新 Mesh 对象的内部缓存和结构。因为这种更新需要消耗很多时间，所以没有必要在每一次函数调用后都对 Mesh 对象进行更新。但在对 Mesh 对象执行一系列操作后，必须要在它被 3ds max 或其他脚本操作之前调用一次 `update()` 方法；
- ◆ 这些属性和方法都在 MAXScript 的当前坐标系下工作；
- ◆ 如果 Modify 面板正被打开时对当前选择的对象调用 `update()` 方法，为了避免在 3ds max 里可能发生的系统崩溃，系统会自动取消对象的选择，这样如果此时脚本程序有对该对象的修改，这种修改不会生效；
- ◆ 如果创建或修改一个 Mesh 对象后，没有执行方法 `update()`，此时如果最小化后又最大化 3ds max 窗口，会发生系统崩溃。这是因为 Mesh 对象的创建还没有完成，其内部结构还不稳定。

### 9.1.1 Editable\_Mesh 和 TriMesh 构造函数、操作符、属性

#### Editable\_Mesh 的构造函数

```
1. mesh [ length:<Integer> ] [ width:<Integer> ] \
           [ lengthsegs:<Integer> ] [ widthsegs:<Integer> ]
```

用给定的尺寸和线段数创建一个曲面的矩形 Mesh 对象。默认长度和宽度为 50，默认

线段数为 5。

### 2. mesh [ numverts:<number> ] [ numfaces:<number> ]

用指定参数创建一个 Mesh 对象，但是没有拓扑结构（也即无边无面），必须逐个设置顶点的坐标并用这些顶点创建面。参数 numverts: 的默认值为 36；参数 numfaces: 的默认值为 50。

### 3. mesh vertices:<array\_of\_point3s> faces:<array\_of\_point3s> \

[ MaterialIDs:<array\_of\_Integers> ] [ tverts:<array\_of\_point3s> ]

用指定顶点数组和面数组创建一个 Mesh 对象。

顶点数组由可选参数 vertices: 指定，元素数据类型为 Point3，每一个元素表示一个顶点的坐标（在当前坐标系下）。

面数组由可选参数 faces: 指定，元素数据类型为 Point3，每一个元素表示一个面的三个顶点的序列号。

可选参数 MaterialIDs: 为每一个面指定材质 ID 号。

可选参数 tverts: 是一个元素类型为 Point3 的数组，每一个元素为纹理顶点指定 UVW 坐标。

下面是一个用以上格式构造 Mesh 对象的例子：

```
mesh vertices:# ([0,0,0],[10,0,0],[0,10,0],[10,10,0]) \
faces:# ([1,2,3],[2,4,3]) MaterialIDs:# (1,2)
```

### 4. collapseStack <node> mapped

将指定<node>对象的修改器堆栈塌陷，生成一个 Editable\_Mesh 类的结果对象。如果调用本方法时 Modifier Stack 里还没有 Modifier，什么也不会发生。如果想强制将一个对象变成一个 Editable\_Mesh 类对象，以对它进行 Mesh 类操作，可以向修改器堆栈里添加一个空 Modifier，然后塌陷它。如：

```
addModifier $foo (normalModifier())
collapseStack $foo
```

collapseStack()方法仅适用于 Object 坐标系下的 Modifier，对 World 坐标系下的 Modifier（包括 SpaceWarp）不起作用，这些 Modifier 仍然保留在修改器堆栈里。

### 5. snapshot <node> mapped

本方法功能与 3ds max 用户界面的 SnapShot 工具栏类似。它生成一个新的 Node 对象，这个 Node 对象为指定源对象<node>在方法调用时的一个 Mesh 复制对象。源对象的所有 Modifier Stack 里的 Modifier 都塌陷到新 Edidtable\_Mesh 对象里，包括 World 坐标系的 Modifier。

### 6. convertToMesh <node> mapped

直接将指定场景对象转换为 Editable\_Mesh 对象。本方法也会塌陷<node>对象当前的 Modifier Stack，这一点和 collapseStack()方法相同，与 collapseStack()方法不同的是：本方法总是会将源对象转换成 Editable\_Mesh 对象，即使当前在 Modifier Stack 里没有 Modifier。

本方法适用于所有可以添加 Edit\_Mesh Modifier 的任何对象，如 Geometry 和 Shape，但不适用于 Helpers、Spacewarp、Light 等。

### 7. convertTo <node> [ TriMeshGeometry | Mesh ] mapped

将合适的场景对象转换成为 Editable\_Mesh 对象。<node> 的说明同方法 convertToMesh()。

#### TriMesh 的构造函数

##### 1. TriMesh()

创建一个空 TriMesh 类对象。

使用下面的方法来建立一个 Mesh。

##### 2. <node>.mesh

如果<node>对象可以转换成 Mesh，本方法返回它的 Mesh 副本。

##### 3. <editable\_mesh\_baseobject>.mesh

返回基本对象（baseobject）的 Mesh 副本。

##### 4. <trimesh>.mesh

返回另一个 TriMesh 对象的 Mesh 副本。

**注意** 对 GeometryClass 类对象进行布尔运算，其结果也生成一个 Editable\_Mesh 对象。

#### 操作符

下面的操作符对 Mesh 对象执行布尔运算。这些运算对第一个 Mesh 操作对象的结构进行解构，来包含布尔运算的结果，正如<node>对象的布尔运算一样。

<mesh> + <mesh> (联合)

<mesh>-<mesh> (求异)

<mesh> \* <mesh> (相交)

#### 属性

属性名称	数据类型	说明
<mesh>.numverts	Integer	存取 Mesh 对象的顶点数
<mesh>.numfaces	Integer	存取 Mesh 对象的面数
<mesh>.numtverts	Integer	存取 Mesh 对象的纹理顶点数
<mesh>.numcpvverts	Integer	存取 Mesh 对象的 color-per-vertex 顶点数
<mesh>.mesh	TriMesh	读取该属性返回 Mesh 对象的一个副本，该副本为全部 Modifier 施加之后，而在任何 Spacewarp 施加之前。如果给该属性赋值，将<Mesh>对象的.Mesh 属性修改成指定的 TriMesh 值。可以用这个属性和函数 Copy() 来从别的对象的 Mesh 信息创建一个 TriMesh
<mesh>.selectedVerts	VertexArray	获取 Editable_Mesh 对象当前被选择的顶点
<mesh>.verts	VertexArray	获取 Editable_Mesh 对象所有的顶点
<mesh>.selectedFaces	FaceArray	获取 Editable_Mesh 对象当前被选择的面

(续表)

属性名称	数据类型	说明
<mesh>.Faces	FaceArray	获取 Editable_Mesh 对象所有的面
<mesh>.selectedEdges	EdgeArray	获取 Editable_Mesh 对象当前被选择的边
<mesh>.Edges	EdgeArray	获取 Editable_Mesh 对象所有的边
<mesh>.displacementMapping	Boolean	存取 Editable_Mesh 对象是否使用置换贴图。本属性在用户界面的参数卷展栏里没有相应的控件
<mesh>.subdivisionDisplacement	Boolean	存取是否对 Editable_Mesh 对象执行细分位移操作。本属性在用户界面的曲面属性卷展栏里与 Subdivision Displacement 复选框相对应
<mesh>.splitMesh	Boolean	存取在执行细分位移操作时是否分割 Mesh 对象。本属性在用户界面的曲面属性卷展栏里与 Split Mesh 复选框相对应。将本属性设为 True 会自动将属性 .subdivisionDisplacement 变为 True

除了那些返回值为 VertexSelection、FaceSelection 或 EdgeSelection 的属性，可以对其他的属性进行读写操作。如果新赋的值比原来大（以容纳更多的顶点或面）Mesh 对象会丢失所有当前的顶点、面等数据，必须用方法 setNumVerts()、setNumTVerts() 和 setNumFaces() 来重新建立它们。

如果一个顶点被赋给一个控制器（Controller），那么该顶点可以作为 Mesh 对象的属性来存取，其格式为 .vertex\_N，N 为顶点的序号，如：\$Sphere01.vertex\_209、\$Sphere01.vertex\_210。

### 9.1.2 Mesh 通用方法

如果对象当前还没有被赋给 Object Space 类 Modifier，下面所述的方法的操作对象为 Node 基本对象；如果对象当前已被赋给 Object Space 类 Modifier，下面的 Get 方法获取的是 Modifier 执行之后的 Mesh。下面的 Set 方法仅适用于.baseObject 属性为 Editable\_Mesh 的对象，如果对象当前已有 Object Space 类 Modifier，会导致一个错误信息，通知用户因为有 Modifier 的存在，Mesh 对象不可修改。用户可以用方法 collapseStack()、snapshot() 和 convertToMesh() 来先将对象转换成 Editable\_Mesh 对象。

如果对象当前已有 World Space 类 Modifier，Get 函数操作对象为 Object Space 类 Modifier 执行之后，而在 World Space 类 Modifier 执行之前。目前还没有办法获取 World Space 类 Modifier 之后顶点的坐标值。

下面这些方法中的顶点和面的序号均从 1 开始，并遵循 MAXScript 里其他的索引规则，坐标系均使用当前工作坐标系。

```
1. update <mesh> [ geometry:<boolean> ] [ topology:<boolean> ] \
    [ normals:<boolean> ] mapped
```

更新指定<Mesh>对象，让所有在脚本里对 Mesh 对象进行的修改变得可见，用户在对 Mesh 对象进行一些修改后必须调用它以在视窗里观察修改的结果是否正确。因为 update() 方法需要花费很多时间，因此系统单独将它作为一个方法，这样可以在完成一系列修改之

后再调用它，以节约运行时间。

三个关键字参数指定了更新类型：

- ◆ 如果参数 geometry: 为 True， Geometry 对象缓存被重建（包括法线和边）， Mesh 对象的 bounding box 变得无效；
- ◆ 如果参数 topology: 为 True， Mesh 对象的拓扑结构将被重建；
- ◆ 如果参数 normals: 为 True， Mesh 对象的面法线被重新计算。

上面三个参数的默认值都为 True，这样一个简单的 update()方法调用会导致 Mesh 对象的完全重建。

## 2. attach <mesh> <node>

本方法与 Editable\_Mesh 的 Modify 面板里 Mesh attach 的功能对应，允许向<mesh>里添加对象。先从<node>对象里提取 Mesh 信息（如果需要，先将<node>转换成 Mesh 类对象），然后将它添加到<Mesh>对象（必须为 Editable\_Mesh 类对象）里，最后删除<node>对象。<node>对象里的材质和材质 ID 被合并到<mesh>对象里，使用 Editable\_Mesh 的 Modify 面板里有关 attach 操作的默认设置。本方法不适用于 TriMesh 类对象。

```
3. meshop.attach {<target_editable_mesh_Node> | <target_mesh>} \
    {<source_Node> | <source_mesh>} \
    targetNode:<Node=unsupplied> \
    sourceNode:<Node=unsupplied> \
    attachMat:<{#neither | #MatToID | #IDToMat}=#neither> \
    condenseMat:<boolean=True> \
    deleteSourceNode:<boolean=True>
```

将源对象（对象类型为 Mesh 或 Node）附加到目标对象（对象类型为 Mesh 或 Node）里去。

如果指定的源对象和目标对象类型为 Mesh 而不是 Node，本操作使用 Mesh 对象的 Local 坐标系，且不执行材质修正；如果指定的源对象或目标对象类型为 Mesh，相应参数 targetNode: 或 sourceNode: 被检查是否指定为 Node 类对象，如果是，本操作使用该 Node 对象的转换矩阵和材质。如果指定的源对象或目标对象类型为 Node，不用检查相应的参数 targetNode: 或 sourceNode:。

可选参数 attachMat: 和 condenseMat: 仅适用于 Editable\_Mesh 对象。

可选参数 condenseMat: 只有在可选参数 attachMat:#IDToMat 被指定时才有效。

如果参数 deleteSourceNode:False 没有被指定，且源对象类型为 Node（或源对象类型为 Mesh 且参数 sourceNode: 被指定），在本方法执行完成后源对象会被删除。

例如：

```
meshop.attach $baseobject $sphere02 targetNode:$
meshop.attach $Box01 $Box02 attachMat:#IDToMat \
condenseMat:True deleteSourceNode:False
```

## 4. meshop.getUIParam <mesh> <param\_name>

```
meshop.getUIParam <node> [ <Modifier_or_index> ] <param_name>
```

```
meshop.setUIParam <mesh> <param_name> { <Float> | <boolean> }
meshop.setUIParam <node> [ <Modifier_or_index> ] \
    <param_name> { <number> | <boolean> }
```

存取 Mesh 对象在用户界面里的参数值。可选参数<Modifier\_or\_index>表示指定场景对象里要被赋值的 Edit\_Mesh Modifier，这些方法与用户界面相关，且指定的 Editable\_Mesh 对象或 Edit\_Mesh Modifier 当前必须显示在 Modify 面板里。

有效的 param\_name 值及它们的含义、参数值数据类型见下表：

param_name	返回数据类型	含义
#SelByVert	Boolean	按顶点选择
#IgBack	Boolean	是否忽略背面
#IgnoreVis	Boolean	是否忽略可见边
#PolyThresh	Number	平面选择阈值
#SoftSel	Boolean	是否使用 Soft Selection
#SSUseEDist	Boolean	是否使用 Edge Distance
#SSEDist	Number	设置 Edge Distance 值
#SSBack	Boolean	是否忽略背面，其值与#IgBack 的值相反
#Falloff	Number	设置衰减值
#Pinch	Number	设置曲线收缩值
#Bubble	Number	设置曲线膨胀值
#WeldDist	Number	设置顶点焊接值
#WeldBoxSize	Number	设置长方体焊接尺寸，单位为像素
#ExtrudeType	Boolean	设置挤出类型：True: Group; False: Local
#ShowVNormals	Boolean	设置是否显示顶点法线
#ShowFNormals	Boolean	设置是否显示面法线
#NormalSize	Number	设置法线缩放比例

### 9.1.3 Mesh Vertex 方法

下面是一些存取 Mesh 顶点和顶点法线的基本方法。

#### 1. getNumVerts <mesh>

返回一个整数，表示 Mesh 对象的顶点数，等于属性<mesh>.numverts 的返回值。

#### 2. setNumVerts <mesh> <vert\_index\_Integer> [ <boolean> ]

将 Mesh 对象的顶点数设置为指定数。如果可选参数<boolean>为 True，已有的拓扑结构将被保留；如果<boolean>为 False 或没有被指定，已有的拓扑结构会被丢失。

#### 3. getVert <mesh> <vert\_index\_Integer>

返回一个 Point3 值，表示指定顶点在当前工作坐标系下的坐标值。

#### 4. setVert <mesh> <vert\_index\_Integer> \

(<point3> | <Float> <Float> <Float>)

为指定顶点的坐标值赋值，可以是一个 Point3 值，也可以是三个 Float 值（分别表示 X、Y、Z 坐标）。

5. deleteVert <mesh> <vert\_index\_Integer>

从 Mesh 对象删除指定顶点和使用该点的所有面。本方法调用后系统会对顶点和面重新编号。

6. getNormal <mesh> <vert\_index\_Integer>

返回一个 Point3 值，表示指定顶点所处位置的法线，法线基于使用顶点的面以及这些面的光滑组。

7. setNormal <mesh> <vert\_index\_Integer> <point3>

为指定顶点设置法线矢量。

8. getVertSelection <node> [ <Modifier\_or\_index> ] [ name:<name> ]

getVertSelection <mesh>

返回一个 BitArray 数组，表示当前顶点选择集，或表示对象指定 Modifier 里指定命名选择集里的顶点（如果可选参数 name: 被指定）。可选参数<Modifier\_or\_index>表示指定场景对象<node>的 Edit Mesh 或 Mesh Select 修改器的序号，如：

```
getVertSelection $foo $foo.Modifiers[3]
```

返回#{2..5,9,25..27}。

如果没有指定可选参数<Modifier\_or\_index>，<node>必须是一个 Editable\_Mesh 对象，并且指定选择集取自该基本对象；如果指定可选参数<Modifier\_or\_index>，<node>必须是一个修改器 Edit Mesh 或 Mesh Select 修改器或一个数字（表示从修改器堆栈顶部往下的第几个 Edit Mesh/Mesh Select 修改器）。

本方法实际是当前对象顶点选择集的一个“快照”，当改变选择集时，返回的 BitArray 值不会自动刷新。也就是说，它保存了对象顶点选择集，即使 Modify 面板已不处在 Vertex 子对象级，这样当重新进到 Vertex 子对象级时，又可以恢复先前的顶点选择集。

如果在 Modifier Stack 里有几个 Edit\_Mesh Modifier，而它们的名称又不是惟一的，可以使用选择集 Modifiers 数组来存取某一 Edit\_Mesh 修改器，还可以构造一个字符串来对数组进行索引，如：

```
$foo.Modifiers["edit mesh" + i as string]

9. setVertSelection <node> [ <Modifier_or_index> ]      \
    (<sel_bitarray> | <sel_array>) \
    [ name:<name> ] [ keep:<boolean> ]
setVertSelection <mesh>(<sel_bitarray> | <sel_array>)\ \
    [ keep:<boolean> ]
```

为 Mesh 类对象、Mesh Select 修改器、Edit\_Mesh Modifier 或 TriMesh 类对象设置顶点选择集。本方法是 getVertSelection 方法的镜像操作。

例如：

```
setVertSelection $foo #{1..4,100..102}
```

```
setFaceSelection $foo \
(getFaceSelection $baz)keep:True
```

如果可选参数 name:被指定, Editable\_Mesh 或 Mesh Select 修改器里指定名称选择集里的顶点将被选择; 如果可选参数 name:没有被指定, <sel\_bitarray>或<sel\_array>里指定的顶点将被选择。

如果可选参数 keep:被指定为 False 或没有被指定, 所有当前选择会被新选择替代; 如果可选参数 keep:被指定为 True, 新选择集被添加到当前选择集里。

name:参数不适用于 TriMesh 对象。

#### 10. averageSelVertCenter <node> [ <Modifier\_or\_index> ]

返回一个 Point3 值, 表示指定 Mesh 对象里被选择顶点的中心。如果<node>对象类型不是 Mesh, 返回 undefined。如果在 Mesh 对象里没有顶点被选择, 返回[0,0,0]。本方法不适用于 TriMesh 对象。

参数<Modifier\_or\_index>解释详见 getVertSelection。

#### 11. averageSelVertNormal <node> [ <Modifier\_or\_index> ]

返回一个 Point3 值, 表示指定 Mesh 对象里被选择顶点的平均标准化法线。如果参数<node>指定对象的类型不是 Mesh, 返回 undefined。如果在 Mesh 对象里没有顶点被选择, 返回[0,0,0]。本方法不适用于 TriMesh 对象。

参数<Modifier\_or\_index>解释详见 getVertSelection。

本方法里使用的顶点法线不考虑这些顶点所在面的光滑组数据。

### 9.1.4 Meshop Vertex 方法

下面是一些存取 Mesh 顶点和顶点法线的高级方法。

#### 1. meshop.breakVerts <Mesh mesh> <vertlist>

将数组<vertlist>里的每一个顶点, 在同样的位置生成 N-1 个新的顶点 (N 为使用这个顶点的面数), 让每一个面使用不同的顶点。

#### 2. meshop.chamferVerts <Mesh mesh> <vertlist> <Float amount>

用指定数值为指定顶点切角。

#### 3. meshop.cloneVerts <Mesh mesh> <vertlist>

克隆指定顶点。

#### 4. meshop.cutVert <Mesh mesh> <int start\_vert> <point3 destination> \ <point3 projdir> Node:<Node=unspecified>

剪切指定顶点。

如果<Mesh>对象类型为 Node 或 Editable\_Mesh, 且参数 Node:被指定, 参数<destination>和<projdir>使用当前坐标系; 如果<Mesh>是一个 Editable\_Mesh 对象, 而参数 Node:没有被指定, 参数<destination>和<projdir>使用 Mesh 对象的 Local 坐标系。

如果顶点被创建, 返回值为新顶点的序号; 否则, 返回 undefined。

#### 5. meshop.deleteIsoVerts <Mesh mesh>

删除所有没有被面使用的顶点。

6. `meshop.deleteVerts <Mesh mesh> <vertlist>`

删除指定顶点。

7. `meshop.detachVerts <Mesh mesh> <vertlist> \`

`delete:<boolean=True> asMesh:<boolean=False>`

分离那些使用指定顶点`<vertlist>`的面。

如果参数 `delete:` 为 `True`, 这些面在分离后被删除; 如果参数 `delete:` 为 `False`, 这些面在被分离后不会被删除。

如果参数 `asMesh:` 为 `True`, 这些面在被分离后作为一个 `Mesh` 值被返回; 如果参数 `asMesh:` 为 `False`, 这些面作为 `Mesh` 对象的一个元素, 返回 `OK`。

8. `meshop.getHiddenVerts <Mesh mesh>`

返回一个 `BitArray` 值, 数组元素为 `Mesh` 对象里被隐藏的顶点序号。

9. `meshop.getIsoVerts <Mesh mesh>`

返回一个 `BitArray` 值, 数组元素为 `Mesh` 对象里没有被面使用的顶点序号。

10. `meshop.getNumVerts <Mesh mesh>`

返回 `Mesh` 对象的顶点数。

11. `meshop.getVert <Mesh mesh> <int vertIndex> Node:<Node=unsupplied>`

返回指定顶点的坐标值。如果参数`<mesh>`为 `Node` 对象, 或为 `Editable_Mesh` 类对象且参数 `Node:` 被指定, 返回在当前工作坐标系下的坐标值; 如果`<mesh>`为 `Editable_Mesh` 类对象而参数 `Node:` 没有被指定, 返回在 `Mesh` 对象的 `Local` 坐标系下的坐标值。

12. `meshop.getVertexAngles <Mesh mesh> <vertlist>`

返回一个数组, 长度为`<Mesh>`对象的顶点数, 那些没有出现在参数`<vertlist>`里的顶点在返回数组里对应的值为 0。

本方法为每一个顶点计算其所在面上由该顶点形成的角度的总和。比如: 一个处于网格中间的点得到的顶点角度总和是  $2\pi$ , 而一个 `Box` 对象的一个角点得到的顶点角度总和是  $3.0/2\pi$ 。

13. `meshop.getvertsByColor <Mesh mesh> <color color> \`

`<Float red_thresh> <Float green_thresh> \`

`<Float blue_thresh> channel:<int=0>`

返回一个`<BitArray>`值, 元素为那些颜色在指定范围的顶点序号, 颜色范围值应该在 0~255 之间。

14. `meshop.getvertsByColor <Mesh mesh> <point3 uvw> \`

`<Float u_thresh> <Float v_thresh> \`

`<Float w_thresh> channel:<int=0>`

返回一个`<BitArray>`值, 元素为那些 UVW 在指定范围的顶点序号, UVW 范围值应该在 0~1 之间。

15. `meshop.makevertsPlanar <Mesh mesh> <vertlist>`

移动指定顶点, 使其处在同一个平面上。

16. `meshop.minVertexDistanceFrom <Mesh mesh> <int vertIndex> <vertlist>`

返回从指定顶点<int vertIndex>到数组<vertlist>包含的顶点之间最短的距离值。

17. meshop.minVertexDistancesFrom <Mesh mesh> <vertlist> <int iterations>

本方法用于计算从被选顶点（由参数<vertlist>指定）到未选顶点之间的距离（以边为路径），返回值为数组，长度为 Mesh 对象的顶点数。数组元素值有下面几种情况：

- ◆ 如果<vertlist>为一个空数组，返回数组元素值都为-1；
- ◆ 被选顶点对应的元素返回值为 0；
- ◆ 如果一未选顶点到被选顶点数组中某一顶点之间连接的边数小于或等于参数<int iterations>指定的值，该未选顶点对应元素返回值为从该顶点到被选顶点之间以边为路径的最小距离；
- ◆ 如果一未选顶点到被选顶点中某一顶点之间连接的边数大于参数<int iterations>指定的值，该未选顶点对应元素返回值为-1；
- ◆ 如果<int iterations>为 0，返回值为从未选顶点到被选顶点之间的最近距离。

18. meshop.moveVert <Mesh mesh> <vertlist> \

<point3 offset> Node:<Node=unsupplied>

将指定顶点移动指定距离。如果<Mesh>为 Node 类对象，或者为 Editable\_Mesh 对象，而参数 Node:被指定，移动距离在当前工作坐标系下，如果<Mesh>为 Editable\_Mesh 对象，而参数 Node:没有被指定，移动距离在 Mesh 对象的 Local 坐标系下。

19. meshop.moveVertsToPlane <Mesh mesh> <vertlist> <point3 normal> \

<Float offset> Node:<Node=unsupplied>

将指定顶点<vertlist>移动到指定平面上。顶点都沿参数<point3 normal>指定的法线方向移动。如果<Mesh>为 Node 对象，或者为 Editable\_Mesh 对象，且参数 Node:被指定，那么法线在当前工作坐标系下；如果<Mesh>为 Editable\_Mesh 对象，而参数 Node:没有被指定，那么法线在 Mesh 对象的 Local 坐标系下。

20. meshop.setHiddenVerts <Mesh mesh> <vertlist>

隐藏指定顶点，未选顶点为可视的。

21. meshop.setNumVerts <Mesh mesh> <int>

设置指定<Mesh>对象的顶点数。新增的顶点均在 Mesh 对象的 Local 坐标系原点被创建。

22. meshop.setVert <Mesh mesh> \

<vertlist> <point3 pos> Node:<Node=unsupplied>

将指定顶点移到指定位置。如果<Mesh>为 Node 对象，或者<Mesh>为 Editable\_Mesh 且参数 Node:被指定，那么参数<point3 pos>表示在当前工作坐标系下；如果<Mesh>为 Editable\_Mesh 对象，而参数 Node:没有被指定，那么参数<point3 pos>表示在 Mesh 对象的 Local 坐标系下。

23. meshop.setVertColor <Mesh mesh> \

<int mapChannel> <vertlist> <Color color>

为指定通道里的指定顶点设置颜色。

24. meshop.weldVertsByThreshold <Mesh mesh> <vertlist> <Float threshold>

焊接指定距离范围内的顶点。

```
25. meshop.weldVertSet <Mesh mesh> <vertlist> \
    weldpoint:<point3=unspecified> Node:<Node=unspecified>
    meshop.weldVerts
```

焊接指定顶点。如果有指定参数 `weldpoint:`, 焊接后的顶点就在该参数指定的位置; 否则, 焊接后的顶点处在被选顶点的中心位置。如果`<Mesh>`为 `Node` 对象, 或者为 `Editable_Mesh` 对象, 且参数 `Node:`被指定, 那么参数 `weldpoint:`表示在当前工作坐标系下; 如果`<Mesh>`为 `Editable_Mesh` 对象, 而参数 `Node:`没有被指定, 那么参数 `weldpoint:`表示在 `Mesh` 对象的 `Local` 坐标系下。

### 9.1.5 Meshop Vertex 数据方法

下面是一些存取 `Mesh` 顶点数据的方法。

```
1. meshop.setNumVDataChannels <Mesh mesh> \
    <Integer count> keep:<boolean=False>]
```

设置可用的顶点数据通道数。其取值范围为 0~100。如果参数 `keep:`为 `False` 或没有被指定, 旧的数据通道被丢弃; 如果为 `True`, 旧的数据通道被保留。`3ds max` 有以下几个系统预定义的通道:

- ◆ channel 1 Soft Selection (软选择)
- ◆ channel 2 Vertex weight (顶点权重, 用于 NURMS MeshSmooth)
- ◆ channel 3 Vertex Alpha value (顶点 Alpha 通道数据)
- ◆ channel 4 Cornering value (用于细分时的角点数据)

```
2. meshop.getNumVDataChannels <Mesh mesh>
```

返回一个整数, 表示可用的顶点数据通道数。

```
3. meshop.setVDataChannelSupport <Mesh mesh> \
    <Integer vdChannel> <Boolean support>
```

设置是否支持指定的顶点数据通道, 如果参数`<support>`为 `True`, 且当前通道支持状态为 `False`, 系统会给 `Mesh` 对象分配一个新的顶点数据数组, 其长度为 `Mesh` 对象的顶点数; 如果参数`<support>`为 `False`, 而当前通道支持状态为 `True`, 现有的顶点数据数组会被取消; 如果参数`<support>`和当前通道支持状态一样, 不会发生任何操作。如果指定的顶点数据数组在调用本方法时尚不可用, 系统会自动调用方法 `setNumVDataChannels()`。通道序号`<vdChannel>`从 1 开始。

```
4. meshop.getVDataChannelSupport <Mesh mesh> <Integer vdChannel>
```

测试指定 `Mesh` 对象是否支持指定顶点数据通道。

```
5. meshop.getVDataValue <Mesh mesh> \
    <Integer vdChannel> <Integer vert_index>
```

返回一个浮点数, 表示指定数据通道`<Integer vdChannel>`里与指定顶点`<Integer vert_index>`对应的通道数据。

6. meshop.setVDataValue <Mesh mesh> <Integer vdChannel>  
<Integer vert\_index> <Float value>

设置指定数据通道<Integer vdChannel>里与指定顶点<Integer vert\_index>对应的通道数据。

7. meshop.freeVData <Mesh mesh> <Integer vdChannel>

取消指定顶点数据通道数组并关闭该顶点数据通道。

8. meshop.getVSelectWeight <Mesh mesh> <int vertIndex>

返回指定顶点的顶点选择权重值。

9. meshop.setVSelectWeight <Mesh mesh> <vertlist> <Float weight>

设置顶点的顶点选择权重值。

10. meshop.resetVSelectWeights <Mesh mesh>

将所有顶点的顶点选择权重值设为 1.0。

11. meshop.supportVSelectWeights <Mesh mesh>

启用 Vertex Selection Weight 通道。

12. meshop.freeVSelectWeights <Mesh mesh>

删除顶点选择权重数据数组。

13. meshop.getVertWeight <Mesh mesh> <int vertIndex>

返回指定顶点的权重值。

14. meshop.setVertWeight <Mesh mesh> <vertlist> <Float weight>

设置顶点的权重值。

15. meshop.resetVertWeights <Mesh mesh>

为所有顶点设置权重值为 1.0。

16. meshop.supportVertWeights <Mesh mesh>

启用顶点权重通道。

17. meshop.freeVertWeights <Mesh mesh>

删除顶点权重数据数组。

18. meshop.getVAlpha <Mesh mesh> <int vertIndex>

返回指定顶点的 Alpha 值。

19. meshop.setVAlpha <Mesh mesh> <vertlist> <Float alpha>

设置顶点的 Alpha 值。

20. meshop.resetVAlphas <Mesh mesh>

为所有顶点设置 Alpha 值为 1.0。

21. meshop.supportVAlphas <Mesh mesh>

启用顶点 Alpha 通道。

22. meshop.freeVAlphas <Mesh mesh>

删除顶点 Alpha 数据数组。

23. meshop.getVertCorner <Mesh mesh> <int vertIndex>

返回指定顶点的角点值。

24. meshop.setVertCorner <Mesh mesh> <vertlist> <Float weight>

设置指定顶点的角点值。

25. meshop.resetVertCorners <Mesh mesh>

为所有顶点设置角点值为 0。

26. meshop.supportVertCorners <Mesh mesh>

启用 Vertex Corner 通道。

27. meshop.freeVertCorners <Mesh mesh>

删除角点数据数组。

### 9.1.6 Mesh Edge 方法

下面是一些存取 Mesh 对象边的基本方法。

1. getEdgeVis <mesh> <face\_index\_Integer> <edge\_index\_Integer>

返回指定面的指定序号边（1、2 或 3）的可见性，其数据类型为 Boolean。

2. setEdgeVis <mesh> <face\_index\_Integer> <edge\_index\_Integer> <boolean>

设置指定面的指定序号边的可见性。

3. getEdgeSelection <node> [ <Modifier\_or\_index> ] [ name:<name> ]

getEdgeSelection <mesh>

返回一个 BitArray 值，表示当前的边选择集，或指定的命名选择集（如果参数 name: 被指定）。参见方法 getVertSelection() 的说明。

4. setEdgeSelection <node> [ <Modifier\_or\_index> ] \

(<sel\_bitarray> | <sel\_array>)\

[ name:<name> ] [ keep:<boolean> ]

setEdgeSelection <mesh>(<sel\_bitarray> | <sel\_array>)\

[ keep:<boolean> ]

设置指定 Editable\_Mesh 基本对象或 Mesh Select 修改器、Edit\_Mesh Modifier、TriMesh 的边选择集。参见方法 getVertSelection() 的说明。

5. deselectHiddenEdges <mesh>

取消对 Mesh 对象里隐藏边的选择。本方法不适用于 TriMesh 对象。

### 9.1.7 Meshop Edge 方法

下面是一些存取 Mesh 对象边的高级方法。

1. meshop.chamferEdges <Mesh mesh> <edgelist> <Float amount>

将指定边进行切角操作。

2. meshop.cloneEdges <Mesh mesh> <edgelist>

克隆指定边。

3. meshop.collapseEdges <Mesh mesh> <edgelist>

塌陷指定边。

4. meshop.cutEdge <Mesh mesh> <int edge1> <Float prop1> <int edge2> \

<Float prop2> <point3 projdir> Node:<Node=unsupplied>

剪切指定边。

如果有新顶点创建，返回新顶点的序号；否则，返回 undefined。

如果<Mesh>为 Node 对象，或者为 Editable\_Mesh 对象，且参数 Node:被指定，那么参数<point3 projdir>表示在当前工作坐标系下；如果<Mesh>为 Editable\_Mesh 对象，而参数 Node:没有被指定，那么参数<point3 projdir>表示在 Mesh 对象的 Local 坐标系下。

5. meshop.deleteEdges <Mesh mesh> <edgelist> delIsoVerts:<boolean=True>

删除指定边。如果参数 delIsoVerts: 为 True，所有单独顶点（不被任何边使用的边）都将被删除。

6. meshop.divideEdge <Mesh mesh> <int edgeIndex> <Float edgef> \  
 visDiag1:<boolean=False> visDiag2:<boolean=False> \  
 fixNeighbors:<boolean=True> split:<boolean=False>

将指定边<int edgeIndex>在指定长度比例<edgef>处拆分。参数<visDiag1>和<visDiag2>指定新生成的两条边是否可见。如果参数<fixNeighbors>为 True，与指定边相邻的面也会被拆分。如果参数<split>为 True，被拆分的两条边在拆分处拥有单独的顶点，Mesh 对象被沿对角线拆分。

7. meshop.divideEdges <Mesh mesh> <edgelist>

将<edgelist>指定的所有边在中点处拆分成两段，创建新顶点并细分面。

8. meshop.edgeTessellate <Mesh mesh> <facelist> <Float tension>

细分指定边。本运算和 Tessellate Modifier 相同。

参数<tension> 指定边细分的张力，其取值范围很小，为 0~0.5，对应于修改器 Tessellate、Edit Mesh、Editable\_Mesh 的用户界面里的值除以 400。

9. meshop.extrudeEdges <Mesh mesh> <edgelist> <Float height> \  
 dir:<{<point3 dir> | #independent | #common }=#independent> \  
 Node:<Node=unsupplied>

创建一个与指定边对应的新边，然后将新边移动指定值<height>。边移动的方向由参数<dir>指定：

- ◆ 如果<dir>为一个<Point3>值，新边沿该值的方向移动；
- ◆ 如果指定 dir:#independent，新边沿使用该边的面的法线方向移动；
- ◆ 如果指定 dir:#common，新边沿该边的面族的法线方向移动；
- ◆ 如果<Mesh>为 Node 对象，或者为 Editable\_Mesh 对象，且参数 Node:被指定，那么参数 dir:表示在当前工作坐标系下；如果<Mesh>为 Editable\_Mesh 对象，而参数 Node:没有被指定，那么参数 dir:表示在 Mesh 对象的 Local 坐标系下。

10. meshop.getEdgesUsingVert <Mesh mesh> <vertlist>

返回一个<BitArray>值，元素为 Mesh 对象里边的序号，这些边有使用顶点列表<vertlist>中的顶点。

11. meshop.getOpenEdges <Mesh mesh>

返回一个<BitArray>值，元素为 Mesh 对象里开放边的序号。

12. `meshop.turnEdge <Mesh mesh> <int edgeIndex>`

将指定边改向。仅对那些在两边都有面的边有用。

13. `meshop.getVertsUsingEdge <Mesh mesh> <edgelist>`

返回一个 BitArray 值，元素为 Mesh 对象里顶点的序号，这些顶点有被参数`<edgelist>`指定的边使用。

14. `meshop.getFacesUsingEdge <Mesh mesh> <edgelist>`

返回一个 BitArray 值，元素为 Mesh 对象里面的序号，这些面有被参数`<edgelist>`指定的边使用。

15. `meshop.getPolysUsingEdge <Mesh mesh> <edgelist> \`

`ignoreVisEdges:<boolean=False> threshold:<Float=45.>`

返回一个 BitArray 值，元素为 Mesh 对象里面的序号，这些面被参数`<edgelist>`组成的多边形所包含。参数 `threshold:`的默认值为 45 度。如果参数 `ignoreVisEdges:`为 True，边的可见性被忽略但参数 `threshold:`仍有效。

16. `meshop.getEdgesReverseEdge <Mesh mesh> <edgelist>`

返回一个 BitArray 值，元素为 Mesh 对象里边的序号，这些边在参数`<edgelist>`之外，且与参数`<edgelist>`里某一边共用一个顶点。

17. `meshop.autoEdge <Mesh mesh> <edgelist> <Float threshold> \`

`type:<{#SetClear | #Set | #Clear}=#SetClear>`

根据参数`<Float threshold>`指定的值，设置指定边的可见性。如果某条边处在本方法的参数`<edgelist>`里，那么与它共用顶点的另一边必须也在参数`<edgelist>`里。

### 9.1.8 Mesh Face 方法

下面是一些存取 Mesh 对象面的基本方法。

1. `getNumFaces <mesh>`

返回一个整数，表示 Mesh 对象的面数，等于属性`<mesh>.numfaces`的值。

2. `getFace <mesh> <face_index_Integer>`

返回一个 Point3 值，表示指定面的三个顶点序号。

3. `setFace <mesh> <face_index_Integer> <vert_indices_point3>`

用一个三元数组`<vert_indices_point3>`为指定面设置三个顶点序号。

4. `setFace <mesh> <face_index_Integer> <vert_index_Integer> \`

`<vert_index_Integer> <vert_index_Integer>`

用三个整数为指定面设置三个顶点序号。

5. `deleteFace <mesh> <face_index_Integer>`

从 Mesh 对象删除指定面，并自动对面进行重新编号。

6. `collapseFace <mesh> <face_index_Integer>`

从 Mesh 对象删除指定面，并将它的三个顶点在该面的中心焊接，自动对面进行重新编号。

7. `getFaceNormal <mesh> <face_index_Integer>`

返回一个 Point3 值，表示指定面的法线。

8. setFaceNormal <mesh> <face\_index\_Integer> <point3>

为指定面设置法线矢量。当对 Mesh 对象调用方法 update() 时，这个值会被覆盖。

9. getFaceMatID <mesh> <face\_index\_Integer>

返回一个整数，表示指定面的材质 ID 码。

10. setFaceMatID <mesh> <face\_index\_Integer> <Integer>

为指定面设置材质 ID 码。

11. getFaceSmoothGroup <mesh> <face\_index\_Integer>

返回一个 32 位整数，表示指定面的光滑组数据。共有 32 种可能的光滑组。返回值的每一位对应一个光滑组。可以用下面的函数将一个面的光滑组数据作为一个 BitArray 值返回：

```
fn getfacesmoothgroupB obj face =
(local sgroup_val=getfacesmoothgroup obj face
local sg_bitarray=#{}
if sgroup_val < 0 do
(sg_bitarray[32]=True
sgroup_val -= 2^31
) --if end
for i = 1 to 31 do
(
sg_bitarray[i]=(mod sgroup_val 2 > .5)
sgroup_val /= 2
) --for end
sg_bitarray
) --fn end
```

12. setFaceSmoothGroup <mesh> <face\_index\_Integer> \

<smoothing\_group\_Integer>

为指定面设置光滑组数据。如果有一个 BitArray 值想作为一个面的光滑组，可以用下面函数进行设置：

```
fn setfacesmoothgroupB obj face sg_bitarray =
( local sgroup_val=0
for i in sg_bitarray do sgroup_val += 2^(i-1)
setfacesmoothgroup obj face sgroup_val
update obj
)
```

13. getFaceSelection <node> [ <Modifier\_or\_index> ] [ name:<name> ]

getFaceSelection <mesh>

返回面选择集，或指定命名选择集（如果可选参数 name: 被指定），返回值类型为 BitArray。请参见方法 getVertSelection() 的说明。

14. setFaceSelection <node> [ <Modifier\_or\_index> ] \

(<sel\_bitarray> | <sel\_array>) [ name:<name> ] \

```
[ keep:<boolean> ] setFaceSelection <mesh> \
(<sel_bitarray> | <sel_array>) [ keep:<boolean> ]
```

为一个 Editable\_Mesh 基本对象或 Mesh Select 修改器、Edit\_Mesh Modifier、TriMesh 设置面选择集。请参见方法 setVertSelection()的说明。

#### 15. deselectHiddenFaces <mesh>

取消对 Mesh 对象里隐藏面的选择。本方法不适用于 TriMesh。

#### 16. extrudeFace <mesh>(<index> | <bitarray> | \ <Integer\_array> | #selection)<amount> <scale> \ [ dir (<point3> | #common | #independent) ]

本方法能实现与 Extrude Face Modifier 相同的功能。本方法仅适用于 Editable\_Mesh 对象。第二个参数指定了要挤出的面，可以是单个面序号，也可以是一个指定系列面的 BitArray 值、一个指定面序列号的数组，还可以是一个 Name 类值#selection，表示在 Editable\_Mesh 里的当前面选择集。

参数<amount>指定挤出距离；参数<scale>指定面的缩放百分比。

如果参数 dir:没有被指定，每一个顶点都沿其平均法线单独移动，就像在 Extrude Face Modifier 里一样；如果参数 dir:被指定，它可以是一个 Point3 值，所有顶点都沿它指定的方向移动；也可以是默认值#independent 或#common，则沿所有面的平均法线移动顶点。#common 设置与 Editable\_Mesh 里的 Group Normal 选项一样，而#independent 设置与 Local Normal 选项一样。例如：

```
s=sphere()
extrudeface s #{1..20} 10 100 dir:#independent
update s
```

#### 17. meshop.getVertsUsingFace <Mesh mesh> <facelist>

返回一个 BitArray 值，元素为 Mesh 对象的顶点序号，这些顶点被参数<facelist>指定的面使用。

#### 18. meshop.getEdgesUsingFace <Mesh mesh> <facelist>

返回一个 BitArray 值，元素为 Mesh 对象的边序号，这些边被参数<facelist>指定的面使用。

#### 19. meshop.getPolysUsingFace <Mesh mesh> <facelist> \ ignoreVisEdges:<boolean=False> threshold:<Float=45.>

返回一个 BitArray 值，元素为 Mesh 对象的面序号，这些面被参数<facelist>组成的多边形所包含。参数 threshold:的默认值为 45 度。如果参数 ignoreVisEdges:为 True，边的可见性被忽略，但参数 threshold:仍有效。

#### 20. meshop.autoSmooth <Mesh mesh> <facelist> <Float threshold>

自动将面夹角小于参数<Float threshold>指定角度的面做光滑处理。

#### 21. meshop.unifyNormals <Mesh mesh> <facelist>

统一指定面的法线。

#### 22. meshop.flipNormals <Mesh mesh> <facelist>

反转指定面的法线。

### 23. meshop.getFaceArea <Mesh mesh> <facelist>

返回一个浮点数，表示参数<facelist>指定面的面积。

#### 9.1.9 Meshop Face 方法

下面是一些存取 Mesh 对象面的高级方法。

```
1. meshop.bevelFaces <Mesh mesh> <facelist> \
    <Float height> <Float outline> \
    dir:<{<point3 dir> | #independent | #common} =#independent> \
    Node:<Node=unsupplied>
```

将指定面移动指定高度<height>，并按指定距离<outline>偏移复制这些面，面移动的方向由参数<dir>指定：

- ◆ 如果<dir>为一个<Point3>值，面沿该值的方向移动；
- ◆ 如果指定 dir:#independent，新边沿单独面法线方向移动；
- ◆ 如果指定 dir:#common，新边沿面族的法线方向移动；
- ◆ 如果<Mesh>为 Node 对象，或者为 Editable\_Mesh 对象，且参数 Node:被指定，参数 dir:表示在当前工作坐标系下；如果<Mesh>为 Editable\_Mesh 对象，而参数 Node:没有被指定，参数 dir:表示在 Mesh 对象的 Local 坐标系下。

### 2. meshop.cloneFaces <Mesh mesh> <facelist>

克隆指定面。

### 3. meshop.collapseFaces <Mesh mesh> <facelist>

塌陷指定面。

### 4. meshop.cutFace <Mesh mesh> <int face> <point3 start> \

```
<point3 destination> <point3 projdir> Node:<Node=unsupplied>
```

剪切指定面。

如果<Mesh>为 Node 对象，或者为 Editable\_Mesh 对象，且参数 Node:被指定，那么参数 start、destination 和 projdir 表示在当前工作坐标系下；如果<Mesh>为 Editable\_Mesh 对象，而参数 Node:没有被指定，那么参数<start>、<destination>和<projdir>表示在 Mesh 对象的 Local 坐标系下。

如果创建了新顶点，返回新顶点的序号；否则，返回 undefined。

### 5. meshop.deleteFaces <Mesh mesh> <facelist> delIsoVerts:<boolean=True>

删除指定面。如果参数 delIsoVerts: 为 True，任何单独顶点都将被删除。

### 6. meshop.detachFaces <Mesh mesh> <facelist> \

```
delete:<boolean=True> asMesh:<boolean=False>
```

分离指定面。

如果参数 delete: 为 True，这些面在分离后被删除；如果参数 delete: 为 False，这些面在分离后不会被删除。

如果参数 `asMesh`: 为 `True`, 这些面在分离后作为一个 `Mesh` 值返回; 如果参数 `asMesh`: 为 `False`, 这些面作为 `Mesh` 对象的一个元素, 返回 `ok`。

7. `meshop.divideFace <Mesh mesh> <int faceIndex> \<baryCoord:<point3=unsupplied>`

拆分指定面。如果指定 `<baryCoord>`, 在对应位置会创建新顶点; 如果没有指定 `<baryCoord>`, 新顶点位于面的中心。

8. `meshop.divideFaceByEdges <Mesh mesh> <int faceIndex>\<int edge1Index> <Float edge1f> <int edge2Index> <Float edge2f> \<fixNeighbors:<boolean=True> split:<boolean=False>`

拆分指定面。新顶点在指定边的指定距离处创建。如果参数 `<fixNeighbors>` 为 `True`, 相邻面也会被拆分, 以避免产生接缝。如果参数 `<Split>` 为 `True`, 会为两条新边创建两个单独的顶点, 将指定面沿对角线分割。

9. `meshop.divideFaces <Mesh mesh> <facelist>`

拆分指定面。新顶点在指定面的中心处创建。

10. `meshop.explodeAllFaces <Mesh mesh> <Float threshold>`

将指定 `Mesh` 对象炸开成单独的元素。

参数 `<threshold>` 指定了面与面之间的角度值, 如果大于这个值, 它们就被放在不同的元素里; 否则就在同一个元素里。

11. `meshop.explodeFaces <Mesh mesh> <facelist> <Float threshold>`

将指定 `Mesh` 对象的指定面炸开成单独的元素。

参数 `<threshold>` 指定了面与面之间的角度值, 如果大于这个值, 它们就被放在不同的元素里; 否则就在同一个元素里。

12. `meshop.extrudeFaces <Mesh mesh> <facelist> \<Float height> <Float outline> \<dir:<{<point3 dir> | #independent | #common }=&#independent> \<Node:<Node=unsupplied>`

创建一个与指定面对应的新面, 然后将新面移动指定 `<height>` 值, 并将它们以指定 `<outline>` 进行偏移复制。面移动的方向由参数 `<dir>` 指定:

- ◆ 如果 `<dir>` 为一个 `<Point3>` 值, 新面沿该值的方向移动;
- ◆ 如果指定 `dir:#independent`, 新面沿该面的法线方向移动;
- ◆ 如果指定 `dir:#common`, 新面沿该面族的法线方向移动;
- ◆ 如果 `<Mesh>` 为 `Node` 对象, 或者为 `Editable_Mesh` 对象, 且参数 `Node`: 被指定, 那么参数 `dir`: 表示在当前工作坐标系下; 如果 `<Mesh>` 为 `Editable_Mesh` 对象, 而参数 `Node`: 没有被指定, 那么参数 `dir`: 表示在 `Mesh` 对象的 Local 坐标系下。

13. `meshop.getBaryCoords <Mesh mesh> <int faceIndex> \<point3 pos> Node:<Node=unsupplied>`

返回在指定面上指定点的重心坐标值。

如果 `<Mesh>` 为 `Node` 对象, 或者为 `Editable_Mesh` 对象, 且参数 `Node`: 被指定, 返回值

表示在当前工作坐标系下；如果<Mesh>为 Editable\_Mesh 对象，而参数 Node:没有被指定，返回值表示在 Mesh 对象的 Local 坐标系下。

14. meshop.getElementsUsingFace <Mesh mesh> <facelist> \  
fence:<facelist=unsupplied>

返回一个 BitArray 值，元素为 Mesh 对象的面序号，这些面所在元素的边中至少有一个面在参数<facelist>所指定的面列表中。如果有指定参数 fence:，其指定面所在的元素不会被处理。

15. meshop.getFaceCenter <Mesh mesh> <int faceIndex> Node:<Node=unsupplied>  
返回指定面的中心坐标。

如果<Mesh>为 Node 对象，或者为 Editable\_Mesh 对象，且参数 Node:被指定，返回坐标表示在当前工作坐标系下；如果<Mesh>为 Editable\_Mesh 对象，而参数 Node:没有被指定，返回坐标表示在 Mesh 对象的 Local 坐标系下。

16. meshop.getFaceRNormals <Mesh mesh> <int faceIndex> Node:<Node=unsupplied>  
返回一个三元数组，表示指定面的三个顶点的渲染法线。

如果<Mesh>为 Node 对象，或者为 Editable\_Mesh 对象，且参数 Node:被指定，法线表示在当前工作坐标系下；如果<Mesh>为 Editable\_Mesh 对象，而参数 Node:没有被指定，法线表示在 Mesh 对象的 Local 坐标系下。

17. meshop.getFacesUsingVert <Mesh mesh> <vertlist>

返回一个 BitArray 值，元素为 Mesh 对象的面序号，这些面使用了参数<vertlist>指定的顶点。

18. meshop.getHiddenFaces <Mesh mesh>

返回一个 BitArray 值，元素为 Mesh 对象的面序号，这些面当前状态为 Hidden。

19. meshop.getNumFaces <Mesh mesh>

返回指定 Mesh 对象的面数。

20. meshop.getVertsUsedOnlyByFaces <Mesh mesh> <facelist>

返回一个 BitArray 值，元素为 Mesh 对象的顶点序号，这些顶点被参数<facelist>指定的面使用。

21. meshop.makeFacesPlanar <Mesh mesh> <facelist>

移动指定面，使它们在同一个平面上。

22. meshop.removeDegenerateFaces <Mesh mesh>

删除 Mesh 对象里所有退化面。所谓退化面就是那些两个或以上的顶点共点的面。

23. meshop.removeIllegalFaces <Mesh mesh>

删除 Mesh 对象里所有非法面。所谓非法面就是那些两个或以上的顶点序号超出范围的面。

24. meshop.setHiddenFaces <Mesh mesh> <facelist>

将参数<facelist>指定的面设为 Hidden，其他的面设为可见。

25. meshop.setNumFaces <Mesh mesh> <int>

设置 Mesh 对象的面数。

### 9.1.10 Mesh 方法

1. meshop.createPolygon <Mesh mesh> <vertIndex array> \  
 smGroup:<int=0> matID:<int=1>

用参数<vertIndex array>指定的顶点创建一个多边形（由一个面集合组成）。多边形里顶点的顺序为数组里顶点的顺序。

参数<smGroup>和<matID>为新面指定了光滑组数据和材质 ID 码。  
 2. meshop.cut <Mesh mesh> <int edge1Index> <Float edge1f> \  
 <int edge2Index> <Float edge2f> <point3 normal> \  
 fixNeighbors:<boolean=True> split:<boolean=True> \  
 Node:<Node=unsupplied>

剪切 Mesh 对象。各参数含义如下：

- ◆ <edge1Index>指定开始剪切的边；
- ◆ <edge1f>指定沿起始边开始剪切的长度比例；
- ◆ <edge2Index>指定剪切结束的边；
- ◆ <edge2f>指定沿结束边开始剪切的长度比例；
- ◆ <normal>指定视图方向。剪切将从 Mesh 对象的这一视图方向开始进行，剪切平面由<normal>矢量和剪切起止点形成的矢量组成；
- ◆ 如果参数<fixNeighbors>为 True，与剪切边相邻的面会被分割；
- ◆ 如果<Mesh>为 Node 对象，或者为 Editable\_Mesh 对象，且参数 Node:被指定，法线<normal>表示在当前工作坐标系下；如果<Mesh>为 Editable\_Mesh 对象，而参数 Node:没有被指定，法线<normal>表示在 Mesh 对象的 Local 坐标系下。

3. meshop.getPolysUsingVert <Mesh mesh> <vertlist> \  
 ignoreVisEdges:<boolean=False> threshold:<Float=45.>

返回一个 BitArray 值，元素为 Mesh 对象的面序号，这些面处在由顶点列表<vertlist>定义的多边形里。参数 threshold:的默认值为 45 度。如果参数 ignoreVisEdges:为 True，边的可见性被忽略，但参数 threshold:仍有效。

4. meshop.optimize <Mesh mesh> <Float normalThreshold> \  
 <Float edgeThreshold> <Float bias> <Float maxEdge> \  
 saveMatBoundries:<boolean=True> \  
 saveSmoothBoundries:<boolean=True> autoEdge:<boolean=True>

基于一个表面法线极限值来优化 Mesh 对象的面。如果相邻面的表面法线角度值小于参数<Float normalThreshold>指定的极限值，它们会被塌陷成一个三角面。

如果参数 autoEdge:为 True，当相邻面的表面法线角度值小于参数<edgeThreshold>指定的极限值时，这两个面的共用边为可见。

当对 Mesh 对象进行优化时，随着优化程度的加深，会产生一些退化的三角形。增加参数<bias>的值可以减少这种退化的产生。参数<bias>的取值范围为 0~1，取 0 会关闭这种功能，接近 1 会减少优化程度，而保留那些等边三角形。

如果指定参数<maxEdge>的值大于 0，优化函数不会产生长度小于该值的边；如果该参数的值小于等于 0，优化处理对边的长度没有限制。

如果参数 saveMatBoundries:为 True，沿材质边界的面不会发生优化。

如果参数 saveSmoothBoundries:为 True，沿不同的光滑组边界的面被优化。

```
5. meshop.slice <Mesh mesh> <facelist> <point3 normal> <Float offset> \
    separate:<boolean=False> delete:<boolean=False> \
    Node:<Node=unsupplied>
```

沿指定的切片平面对 Mesh 对象进行切片操作。

- ◆ 如果 separate:为 True，指定切片操作将 Mesh 对象分成两个单独的元素；
- ◆ 参数 delete:指定切片操作是否删除切片平面的“下”部。如果<delete>为 True，则参数 separate:的值会被忽略；
- ◆ 如果<Mesh>为 Node 对象，或者为 Editable\_Mesh 对象，且参数 Node:被指定，法线<normal>表示在当前工作坐标系下；如果<Mesh>为 Editable\_Mesh 对象，而参数 Node:没有被指定，法线<normal>表示在 Mesh 对象的 Local 坐标系下。

6. copy <trimesh>

复制源 TriMesh 对象。

7. delete <trimesh>

删除 TriMesh 对象，并释放它所占的内存。

下面的方法不适用于 TriMesh 类对象：

8. animateVertex <mesh> <vertex\_spec>

对 Mesh 对象的指定顶点添加 Controller，参数<vertex\_spec>为指定顶点，其格式有：

- ◆ <Integer\_index> (指定单个顶点)
- ◆ <Integer\_index\_array> (指定顶点数组)
- ◆ #all (指定所有顶点)

在为顶点赋予 Controller 后，这些顶点会显示在 Track View 视图里，允许对它们进一步的脚本操作，例如：

```
animateVertex $Sphere01 #all
```

9. setMesh <mesh> [ numverts:<Integer> ] [ numfaces:<Integer> ]

将 Mesh 对象重置为指定的几何体，但不再有拓扑结构（即没有边和面），所有的 Mesh 数据都会丢失。必须逐个指定顶点的坐标，并用这些顶点来创建面。默认的顶点（参数 numverts:）和面（参数 numfaces:）的数目分别为 36 和 50。

```
10. setMesh <mesh> [ vertices:<array_of_point3s> ] \
    [ faces:<array_of_point3s> ] \
    [ MaterialIDs:<array_of_Integers> ] \
    [ tverts:<array_of_point3s> ]
```

基于给定数组重置 Mesh 对象。只有那些指定的部分被重置。数组 vertices 里指定的所

所有 Point3 值都是在当前坐标系里；数组 faces 里每一个 Point3 值表示一个面的三个顶点的序号；数组 MaterialIDs 指定面的材质 ID 号；数组 tverts 里每一个 Point3 值指定了纹理顶点的 UVW 坐标。

```
11. setMesh <mesh> [ length:<Integer> ] [ width:<Integer> ] \
[ lengthsegs:<Integer> ] [ widthsegs:<Integer> ]
```

用一个平面矩形来重置 Mesh 对象，这个平面矩形的默认长度和宽度为 50， 默认线段数为 5。

```
12. setMesh <mesh> <trimesh>
```

用指定<trimesh>的副本来重置 Mesh 对象。

### 9.1.11 Meshop Mapping 通用方法

1. meshop.deleteIsoMapVertsAll <Mesh mesh>

删除所有贴图通道里没有被贴图面使用的贴图顶点。

2. meshop.freeMapChannel <Mesh mesh> <int mapChannel>

删除指定通道里的贴图顶点和面数组里的数据，并将它们的数量设为 0。

3. meshop.getMapFacesUsingMapVert <Mesh mesh> <int mapChannel> <mapVertlist>

返回一个 BitArray 数组，元素为贴图面序号，这些贴图面使用了参数<mapVertlist>里的贴图顶点。

4. meshop.getMapVertsUsingMapFace <Mesh mesh> <int mapChannel> <mapFacelist>

返回一个 BitArray 数组，元素为贴图顶点序号，这些贴图顶点使用了参数<mapFacelist>里的贴图面。

5. meshop.setFaceAlpha <Mesh mesh> <int mapChannel> <facelist> <Float alpha>

将指定贴图通道里的部分贴图顶点的颜色设为指定颜色（color <alpha> <alpha> <alpha>），这些贴图顶点被参数<facelist>里的贴图面使用。

6. meshop.setFaceColor <Mesh mesh> <int mapChannel><facelist> <color color>

将指定贴图通道里的部分贴图顶点的颜色设为<color>指定的颜色，这些贴图顶点被参数<facelist>里的贴图面使用。

7. meshop.setVertAlpha <Mesh mesh> <int mapChannel> <vertlist> <Float alpha>

将指定贴图通道里参数<vertlist>指定的贴图顶点颜色设为指定颜色（color <alpha> <alpha> <alpha> <alpha>）。

8. meshop.setVertColor <Mesh mesh> <int mapChannel> <vertlist> <color color>

将指定贴图通道里参数<vertlist>指定的贴图顶点颜色设为<color>指定的颜色。

9. meshop.setNumMaps <Mesh mesh> <int count> keep:<boolean=False>

设置可用的贴图通道，取值范围为 2~100。通道 1 和通道 2 分别为顶点颜色通道和默认纹理贴图通道。如果可选参数 keep: 为 False，旧的贴图通道信息会被丢弃；如果为 True，旧的贴图通道信息在调整后会被保留。

10. meshop.getNumMaps <Mesh mesh>

返回一个整数，表示 Mesh 对象可用的贴图通道。

## 11. meshop.setMapSupport &lt;Mesh mesh&gt; &lt;Integer mapChannel&gt; &lt;Boolean support&gt;

设置 Mesh 对象是否支持指定的贴图通道。如果参数<support>为 True，且当前通道支持状态为 False，系统会给 Mesh 对象分配一个新贴图面数组，其长度为 Mesh 对象的面数，但没有贴图顶点数组；如果参数<support>为 False，而当前通道支持状态为 True，现有的贴图通道的面数组和顶点数组会被取消；如果参数<support>和当前通道支持状态一样，不会发生任何操作。如果指定的顶点数据数组在调用本方法时尚不可用，系统会自动调用方法 SetNumMaps()。贴图通道索引<mapChannel>从 1 开始。

## 12. meshop.getMapSupport &lt;Mesh mesh&gt; &lt;Integer mapChannel&gt;

返回 Mesh 对象是否支持指定的贴图通道。如果返回 True，表示当前存在一个与通道对应的贴图面数组，但不一定有贴图顶点数组。

## 13. meshop.setNumMapVerts &lt;Mesh mesh&gt; &lt;Integer mapChannel&gt; \&lt;Integer count&gt; keep:&lt;boolean=False&gt;

为指定贴图通道设置顶点数，并初始化贴图顶点数组。如果参数 keep: 为 False 或没有被指定，旧的贴图顶点信息将被丢弃；如果为 True，通道顶点数调整大小后，贴图顶点信息会被保留。

## 14. meshop.getNumMapVerts &lt;Mesh mesh&gt; &lt;Integer mapChannel&gt;

返回一个整数，表示指定贴图通道的顶点数。

## 15. meshop.setNumMapFaces &lt;Mesh mesh&gt; &lt;Integer mapChannel&gt; \&lt;Integer count&gt; keep:&lt;boolean=False&gt; keepCount:&lt;int=0&gt;

为指定贴图通道设置面数。如果可选参数 keep: 为 False 或没有被指定，旧的贴图面信息将被丢弃；如果为 True，由参数 keepCount: 指定数目的旧贴图面信息会被保留。对通道 0 和通道 1 应用本方法是危险的，所以参数<mapChannel>的值应不小于 2。

## 16. meshop.getNumMapFaces &lt;Mesh mesh&gt; &lt;Integer mapChannel&gt;

返回一个整数，表示指定贴图通道的面数。

## 17. meshop.setMapVert &lt;Mesh mesh&gt; &lt;Integer mapChannel&gt; \&lt;Integer index&gt; &lt;Point3 xyz&gt;

为指定贴图顶点设置坐标值。

## 18. meshop.getMapVert &lt;Mesh mesh&gt; &lt;Integer mapChannel&gt; &lt;Integer index&gt;

返回一个<Point3>值，表示指定贴图顶点的坐标值。

## 19. meshop.setMapFace &lt;Mesh mesh&gt; &lt;Integer mapChannel&gt; \&lt;Integer index&gt; &lt;Point3 face&gt;

为指定贴图面设置贴图顶点。

## 20. meshop.getMapFace &lt;Mesh mesh&gt; &lt;Integer mapChannel&gt; &lt;Integer index&gt;

返回一个<Point3>值，表示指定贴图面的三个贴图顶点。

## 21. meshop.makeMapPlanar &lt;Mesh mesh&gt; &lt;Integer mapChannel&gt;

对指定通道施加简易平面贴图。

## 22. meshop.getIsoMapVerts &lt;Mesh mesh&gt; &lt;Integer mapChannel&gt;

返回一个<BitArray>数组，如果某一位设为 On，表示该序号的贴图顶点为一个孤立的

顶点。

23. `meshop.deleteMapVertSet <Mesh mesh> <Integer mapChannel> \ {<BitArray set> | <Integer index> | <Integer array>}`

删除指定贴图顶点。返回一个`<BitArray>`数组，元素为 Mesh 对象的贴图面序号，这些贴图面有使用删除前的这些顶点。

24. `meshop.freeMapVerts <Mesh mesh> <Integer mapChannel>`

释放分配给贴图顶点数组的内存，并将贴图顶点数设为 0。

25. `meshop.freeMapFaces <Mesh mesh> <Integer mapChannel>`

释放分配给贴图面数组的内存，并将贴图面数设为 0。

26. `meshop.applyUVWMap <TriMesh mesh> \`

```
{<#planar | #cylindrical | #spherical | #ball | #box> | <#face>} \
utile:<Float=1.0> vtile:<Float=1.0> wtile:<Float=1.0> \
uflip:<boolean=False> vflip:<boolean=False> wflip:<boolean=False> \
cap:<boolean=True> tm:<Matrix3=identity matrix> channel:<int=1>
```

为指定贴图通道施加指定贴图类型。

- ◆ 参数 `utile:/vtile:/wtile:` U/V/W 方向的贴图数目；
- ◆ 参数 `uflip/vflip/wflip:` 如果为 True，U/V/W 被镜向；
- ◆ 参数 `cap:` 和 `#cylindrical` 一起使用。如果为 True，那些比较垂直的面法线用平面坐标来进行贴图；
- ◆ 参数 `tm:` 定义贴图空间的转换矩阵；
- ◆ 参数 `channel:` 指定贴图作用的通道，默认值为通道 1。

27. `meshop.buildMapFaces <Mesh mesh> <Integer mapChannel> <Boolean keep>`

使用指定通道的贴图面数重新设置 Mesh 对象的面数，没有被贴图面使用的贴图顶点都会被删除。如果参数`<keep>`为 True，保留已有的贴图数据。

28. `meshop.defaultMapFaces <Mesh mesh> \`

`<Integer mapChannel> <Integer count>`

将指定通道的贴图面数设置为 Mesh 对象的面数，其贴图顶点数设置为 Mesh 对象的顶点数。贴图面顶点的序号与 Mesh 对象面顶点序号相同。

29. `meshop.getNumCPVVerts <Mesh mesh>`

返回 Color-Per-Vertex 贴图通道里的顶点数目。

30. `meshop.setNumCPVVerts <Mesh mesh> <int>`

设置 Color-Per-Vertex 贴图通道里的顶点数目。

31. `meshop.getNumTVerts <Mesh mesh>`

返回纹理贴图通道里的默认顶点数目。

32. `meshop.setNumTVerts <Mesh mesh> <int>`

设置纹理贴图通道里的默认顶点数目。

### 9.1.12 Meshop Editable\_Mesh 用户界面控件方法

下面的方法仅用于 Editable\_Mesh 对象：

1. meshop.getAffectBackfacing <Mesh mesh>  
meshop.setAffectBackfacing <Mesh mesh> <boolean>  
存取 Soft Selection 卷展栏里 Affect Backfacing 复选框的状态。
2. meshop.getBubble <Mesh mesh>  
meshop.setBubble <Mesh mesh> <float>  
存取 Soft Selection 卷展栏里 Bubble 值。
3. meshop.getDisplacementMapping <Mesh mesh>  
meshop.setDisplacementMapping <Mesh mesh> <boolean>  
存取置换贴图值。
4. meshop.getExtrusionType <Mesh mesh>  
meshop.setExtrusionType <Mesh mesh> <int>  
存取拉伸类型：1: Group; 2: Local
5. meshop.getFalloff <Mesh mesh>  
meshop.setFalloff <Mesh mesh> <float>  
存取 Soft Selection 卷展栏里 Falloff 值。
6. meshop.getIgnoreBackfacing <Mesh mesh>  
meshop.setIgnoreBackfacing <Mesh mesh> <boolean>  
存取 Selection 卷展栏里 Ignore Backfacing 复选框的状态。
7. meshop.getIgnoreVisEdges <Mesh mesh>  
meshop.setIgnoreVisEdges <Mesh mesh> <boolean>  
存取 Selection 卷展栏里 Ignore Visible Edges 复选框的状态。
8. meshop.getNormalSize <Mesh mesh>  
meshop.setNormalSize <Mesh mesh> <float>  
存取 Selection 卷展栏里 Show Normals 复选框状态为 True 时的 Scale 值。
9. meshop.getPinch <Mesh mesh>  
meshop.setPinch <Mesh mesh> <float>  
存取 Soft Selection 卷展栏里 Pinch 值。
10. meshop.getPlanarThreshold <Mesh mesh>  
meshop.setPlanarThreshold <Mesh mesh> <float>  
存取 Selection 卷展栏里 Planar Thresh 值。
11. meshop.getSelByVertex <Mesh mesh>  
meshop.setSelByVertex <Mesh mesh> <boolean>  
存取 Selection 卷展栏里 By Vertex 复选框的状态。
12. meshop.getShowFNormals <Mesh mesh>  
meshop.setShowFNormals <Mesh mesh> <boolean>  
当处在 Face、Polygon 或 Element 子对象级时，存取 Selection 卷展栏里 Show Normals

复选框的状态。

13. `meshop.getShowVNormals <Mesh mesh>`

`meshop.setShowVNormals <Mesh mesh> <boolean>`

当处在 Vertex 子对象级时，存取 Selection 卷展栏里 Show Normals 复选框的状态。

14. `meshop.getSoftSel <Mesh mesh>`

`meshop.setSoftSel <Mesh mesh> <boolean>`

存取 Soft Selection 卷展栏里 Use Soft Selection 复选框的状态。

15. `meshop.getSplitMesh <Mesh mesh>`

`meshop.setSplitMesh <Mesh mesh> <boolean>`

当处在 Object 模式时，存取 Surface Properties 卷展栏里 Split Mesh 复选框的状态。

16. `meshop.getSSEdgeDist <Mesh mesh>`

`meshop.setSSEdgeDist <Mesh mesh> <int>`

存取 Soft Selection 卷展栏里 Edge Distance 值。

17. `meshop.getSSUseEdgeDist <Mesh mesh>`

`meshop.setSSUseEdgeDist <Mesh mesh> <boolean>`

存取 Soft Selection 卷展栏里 Edge Distance 复选框的状态。

18. `meshop.getSubdivisionAngle <Mesh mesh>`

`meshop.setSubdivisionAngle <Mesh mesh> <float>`

存取 Surface Properties 卷展栏里 Subdivision Angle 值。

19. `meshop.getSubdivisionDisplacement <Mesh mesh>`

`meshop.setSubdivisionDisplacement <Mesh mesh> <boolean>`

存取 Surface Properties 卷展栏里 Subdivision Displacement 复选框的状态。

20. `meshop.getSubdivisionDistance <Mesh mesh>`

`meshop.setSubdivisionDistance <Mesh mesh> <float>`

存取 Surface Properties 卷展栏里 Subdivision Distance 值。

21. `meshop.getSubdivisionEdge <Mesh mesh>`

`meshop.setSubdivisionEdge <Mesh mesh> <float>`

存取 Surface Properties 卷展栏里 Subdivision Edge 值。

22. `meshop.getSubdivisionMaxLevels <Mesh mesh>`

`meshop.setSubdivisionMaxLevels <Mesh mesh> <int>`

存取 Surface Properties 卷展栏下 Advanced Parameters...对话框里 Maximum Subdivision Levels 值。

23. `meshop.getSubdivisionMaxTriangles <Mesh mesh>`

`meshop.setSubdivisionMaxTriangles <Mesh mesh> <int>`

存取 Surface Properties 卷展栏下 Advanced Parameters...对话框里 Maximum Number of Triangles 值。

24. `meshop.getSubdivisionMethod <Mesh mesh>`

`meshop.setSubdivisionMethod <Mesh mesh> \`

- {#regular | #spatial | #curvature | #spatialAndCurvature}  
存取 Surface Properties 卷展栏里的 Subdivision Method 值。
25. meshop.getSubdivisionMinLevels <Mesh mesh>  
meshop.setSubdivisionMinLevels <Mesh mesh> <int>  
存取 Surface Properties 卷展栏下 Advanced Parameters...对话框里 Minimum Subdivision Levels 值。
26. meshop.getSubdivisionSteps <Mesh mesh>  
meshop.setSubdivisionSteps <Mesh mesh> <int>  
存取 Surface Properties 卷展栏里的 Subdivision Steps 值。
27. meshop.getSubdivisionStyle <Mesh mesh>  
meshop.setSubdivisionStyle <Mesh mesh> {#tree | #grid | #delaunay }  
存取 Subdivision Style 值。
28. meshop.getSubdivisionView <Mesh mesh>  
meshop.setSubdivisionView <Mesh mesh> <boolean>  
存取 Surface Properties 卷展栏里 View-Dependent 复选框的状态。
29. meshop.getWeldPixels <Mesh mesh>  
meshop.setWeldPixels <Mesh mesh> <int>  
存取 Edit Geometry 卷展栏里 Target Weld pixels 的值。
30. meshop.getWeldThreshold <Mesh mesh>  
meshop.setWeldThreshold <Mesh mesh> <float>  
存取 Edit Geometry 卷展栏里 Weld Selected Threshold 的值。

### 9.1.13 Mesh Texture Vertex 方法

下面是一些存取 UVW 贴图通道的基本方法。

1. getNumTVerts <mesh>  
返回一个整数, 表示指定 Mesh 对象的纹理顶点个数, 等于属性<mesh>.numtverts 的值。
2. setNumTVerts <mesh> <number> [ <boolean> ]  
为指定 Mesh 对象设置纹理顶点个数, 如果可选参数<boolean>为 True, Mesh 对象现有的纹理顶点结构会被保留。
3. getTVert <mesh> <tvert\_index\_Integer>  
返回指定纹理顶点的 UVW 贴图坐标。
4. setTVert <mesh> <tvert\_index\_Integer> (<point3> | <x> <y> <z>)  
为指定纹理顶点设置 UVW 贴图坐标, 坐标格式可以为一个 Point3 数据, 也可以是 3 个 Float 类型数据, 分别对应 U、V、W 坐标。
5. getTVFace <mesh> <face\_index\_Integer>  
setTVFace <mesh> <face\_index\_Integer> <vert\_indices\_point3>  
setTVFace <mesh> <face\_index\_Integer> <tvert\_index\_Integer> \  
<tvert\_index\_Integer> <tvert\_index\_Integer>

方法 `getTVFace()` 和 `setTVFace()` 分别对应方法 `getFace()` 和 `setFace()`，用于为纹理顶点获取/设置贴图结构。

#### 6. `buildTVFaces <mesh> [ <boolean> ]`

本方法仅用于手工向 `Mesh` 对象添加纹理顶点以创建纹理面数组的情况。那些由 `3ds max` 施加的贴图坐标，系统会自动对其进行维护；而如果用户自己向 `Mesh` 对象添加纹理顶点，当用户改变纹理顶点的数量时，在构造纹理面之前，需要显式地调用本方法。如果可选参数`<boolean>`为 `True`，`Mesh` 对象现有的纹理贴图结构会被保留；否则，所有的纹理面会被重建，纹理面的所有三个顶点都将使用纹理顶点 1。

#### 7. `numMapsUsed <mesh>`

返回一个整数，表示指定 `Mesh` 对象正在使用的纹理贴图通道上限，其最小值不小于 1。本方法可用在下面的情况：如果想对所有贴图通道执行某些操作，只要循环到本方法的返回值即可，而不用一直循环到 99（最高可能的纹理贴图通道）。本方法不能用于 `TriMesh` 类对象。

**注意**

`Mesh` 类对象包含一个纹理顶点列表，这个列表完全独立于 `Mesh` 对象常规的顶点。这两者之间也没有任何关联。由纹理顶点组成纹理面，`Mesh` 对象的每一个常规面必须有一个纹理面与之对应。

每一个纹理面有三个纹理顶点，纹理顶点的坐标系为 `UVW` 坐标系，其数据类型为 `Point3`；坐标分量分别为 `U`、`V`、`W`；其取值范围为 0~1。在本书 9.1.17 节里的示例演示了如何为一个对象创建平面纹理贴图。

### 9.1.14 Mesh Color-Per-Vertex 方法

下面是一些存取 `Color-Per-Vertex` 通道的基本方法。

#### 1. `getNumCPVVerts <mesh>`

返回一个整数，表示指定 `Mesh` 对象的 C-P-V 顶点个数，等于属性`<node>.numcpvverts` 的返回值。

#### 2. `setNumCPVVerts <mesh> <Integer> [ <boolean> ]`

为指定 `Mesh` 对象设置 C-P-V 顶点个数，如果可选参数`<boolean>`为 `True`，`Mesh` 对象现有的 C-P-V 顶点结构会被保留。

#### 3. `buildVCFaces <mesh> [ <boolean> ]`

在设置 C-P-V 顶点数目后，用本方法来构造 C-P-V 面数组。如果可选参数 `<boolean>` 为 `True`，`Mesh` 对象现有的 C-P-V 贴图结构会被保留。

#### 4. `defaultVCFaces <mesh>`

为指定 `Mesh` 对象设置 C-P-V 顶点和 C-P-V 面的数目，以精确匹配 `Mesh` 对象的几何结构，然后调用方法 `buildVCFaces()`。

#### 5. `getVertColor <mesh> <CPVvert_index_Integer>`

返回一个 `Color` 数据，表示指定 C-P-V 顶点的颜色。

#### 6. `getVCFace <mesh> <CPVface_index_Integer>`

返回一个 Point3 数据，包含三个 C-P-V 顶点，表示指定 C-P-V 面的拓扑结构。

7. setVCFace <mesh> <CPVface\_index\_Integer> <CPVvert\_indices\_point3>

用一个 Point3 数据（包含三个 C-P-V 顶点）为指定 C-P-V 面设置拓扑结构。

8. setVCFace <mesh> <CPVface\_index\_Integer> <CPVvert\_index\_Integer> \<CPVvert\_index\_Integer> <CPVvert\_index\_Integer>

用三个 C-P-V 顶点为指定 C-P-V 面设置拓扑结构。

C-P-V 顶点和面的序号均从 1 开始。要记住：在对 Mesh 对象进行修改后，必须调用方法 update() 来刷新 Mesh 对象，这样你所作的修改才可以反映到场景里去。

**注意** Mesh 类对象包含一个 C-P-V 顶点列表，这个列表完全独立于 Mesh 对象常规的顶点。这两者之间也没有任何关联。由 C-P-V 顶点组成 C-P-V 面，Mesh 对象的每一个常规面必须有一个 C-P-V 面与之对应。

每一个 C-P-V 面有三个 C-P-V 顶点，C-P-V 顶点的坐标系为 RGB 坐标系，其数据类型为 Color。存取、构造 C-P-V 顶点和面的方法实际上和存取、构造纹理顶点和面的方法一致。

### 9.1.15 Subdivision Displacement Surface 方法

下面的方法用来为 Mesh 对象设置 Subdivision Displacement Surface（细分位移曲面）属性，这些方法不能用于 TriMesh 对象。

1. setSplitMesh <mesh> <boolean>

打开或关闭指定 Mesh 对象的 Split Mesh Subdivision Displacement 设置。如果<boolean> 为 False，会打开 Subdivision Displacement。

2. getSplitMesh <mesh>

如果 Split Mesh Subdivision Displacement 设置被打开，返回 True；否则，返回 False。

3. setSubdivisionDisplacement <mesh> <boolean>

打开或关闭指定 Mesh 对象的 Subdivision Displacement 设置。

4. getSubdivisionDisplacement <mesh>

如果 Subdivision Displacement 设置被打开，返回 True；否则，返回 False。

5. displacementToPreset <mesh> (#low | #medium | #high)

将网格细分参数设置为指定值。

6. setDisplacementMapping <mesh> <boolean>

设置是否对指定 Mesh 对象执行置换贴图操作。本方法在用户界面里没有相应的界面控件。

7. getDisplacementMapping <mesh>

如果执行置换贴图操作，返回 True；否则，返回 False。

### 9.1.16 Editable\_Mesh Modify 面板命令的操作方法

有一组方法可以让用户在脚本里调用 Modify 命令面板下 Editable\_Mesh 对象和

Edit\_Mesh Modifier 的操作，并和各 Mesh 子对象级下的按钮相对应。这些方法都带一个前缀.meshOps，每一个方法都相当于按下 3ds max 用户界面 Modify 命令面板里的一个按钮。调用 meshOps 类方法有几个前提条件：一是 Modify 命令面板被打开，且被操作的对象（可以是 Editable\_Mesh 类基本对象，也可以是 Edit\_Mesh Modifier）当前要被选择。如果这些条件不能满足，调用 meshOps 类方法不会执行任何操作。用户可以调用 MAXScript 函数（如：max modify mode 或 setCommandPanelTaskMode mode:#modify）来打开 Modify 命令面板；用 select <obj>方法来选择对象；用 subObjectLevel = <n>方法来设置子对象级别；最后还要用 modPanel.setCurrentObject <obj\_or\_Modifier> 来为新函数建立要操作的子对象集合。

在下面这些方法的描述中，第一个变量<editable\_mesh\_Node\_or\_Modifier>可以是一个类型为 Editable\_Mesh 的对象，同时其 Modifier Stack 处于 Base 级；也可以是一个 Edit\_Mesh Modifier，同时其 Modifier Stack 里当前 Modifier 为 Edit Mesh。下面方法均工作在当前子对象级，当然，该子对象级必须适合于要调用的操作。

如果在脚本里对 Mesh 对象进行了修改，别忘了在调用这些方法之前调用 update() 方法。但却不必在调用这些方法后调用 update() 方法，3ds max 系统会自动进行刷新。

调用下面的方法相当于按下 Editable\_Mesh Modify 面板下的一个按钮，并将当前被选择的子对象作为操作对象。调用这些命令时，相应的按钮会高亮显示，就像用鼠标按下该按钮一样，并开始完成指定的指令。

下面逐一列出了这些方法：

1. meshOps.startAttach <editable\_mesh\_Node\_or\_Modifier>

相当于按下 Attach 按钮，适用于 Vertex、Face、Polygon 和 Element 子对象级，以及处在非子对象模式下时。

2. meshOps.startBevel <editable\_mesh\_Node\_or\_Modifier>

相当于按下 Bevel 按钮，适用于 Edge、Face、Polygon 和 Element 子对象级。

3. meshOps.startChamfer <editable\_mesh\_Node\_or\_Modifier>

相当于按下 Chamfer 按钮，适用于 Vertex 和 Edge 子对象级。

4. meshOps.startCreate <editable\_mesh\_Node\_or\_Modifier>

相当于按下 Create 按钮，适用于 Vertex、Face、Polygon 和 Element 子对象级。

5. meshOps.startCut <editable\_mesh\_Node\_or\_Modifier>

相当于按下 Cut 按钮，适用于 Vertex、Face、Polygon 和 Element 子对象级。

6. meshOps.startDivide <editable\_mesh\_Node\_or\_Modifier>

将选择的每一条边分成两条边，选择的每一面分成三个面，适用于 Edge、Face、Polygon 和 Element 子对象级。

7. meshOps.startExtrude <editable\_mesh\_Node\_or\_Modifier>

相当于按下 Extrude 按钮，适用于 Face、Polygon 和 Element 子对象级。

8. meshOps.startFlipNormalMode <editable\_mesh\_Node\_or\_Modifier>

相当于按下 Flip Normal 按钮，适用于 Face、Polygon 和 Element 子对象级。

9. meshOps.startSlicePlane <editable\_mesh\_Node\_or\_Modifier>

相当于按下 Slice Plane 按钮，适用于所有子对象级，以及处在非子对象模式下时。

10. meshOps.startTurn <editable\_mesh\_Node\_or\_Modifier>  
相当于按下 Turn 按钮，适用于 Edge 子对象级。
11. meshOps.startWeldTarget <editable\_mesh\_Node\_or\_Modifier>  
相当于按下 Weld Target 按钮，适用于 Vertex 子对象级。  
下面的方法调用 Editable\_Mesh Modify 面板下的一些“杂项”按钮，同样将当前选择的子对象作为操作对象。
12. meshOps.autoEdge <editable\_mesh\_Node\_or\_Modifier>  
为当前选择的子对象设置边的可见性，适用于 Edge 子对象级。
13. meshOps.break <editable\_mesh\_Node\_or\_Modifier>  
为每一个与当前被选择的子对象连接的面创建一个新的顶点，适用于 Vertex 子对象级。
14. meshOps.collapse <editable\_mesh\_Node\_or\_Modifier>  
将选择的子对象塌陷，适用于 Vertex、Edge、Face、Polygon 和 Element 子对象级。
15. meshOps.createShapeFromEdges <editable\_mesh\_Node\_or\_Modifier>  
显示 Create Shape 对话框。适用于 Edge 子对象级。
16. meshOps.delete <editable\_mesh\_Node\_or\_Modifier>  
删除选择的子对象，适用于 Vertex、Face、Polygon 和 Element 子对象级。
17. meshOps.detach <editable\_mesh\_Node\_or\_Modifier>  
显示 Detach 对话框。适用于 Vertex、Face、Polygon 和 Element 子对象级。
18. meshOps.explode <editable\_mesh\_Node\_or\_Modifier>  
显示 Explode to Objects 对话框。适用于 Vertex、Face、Polygon 和 Element 子对象级。
19. meshOps.flipNormal <editable\_mesh\_Node\_or\_Modifier>  
反转选择面的法线，适用于 Face、Polygon 和 Element 子对象级。
20. meshOps.gridAlign <editable\_mesh\_Node\_or\_Modifier>  
将选择的子对象与构造平面对齐，适用于 Vertex、Edge、Polygon 和 Element 子对象级。
21. meshOps.hide <editable\_mesh\_Node\_or\_Modifier>  
隐藏选择的子对象，适用于 Vertex、Face、Polygon 和 Element 子对象级。
22. meshOps.invisibleEdge <editable\_mesh\_Node\_or\_Modifier>  
将选择的边的可见性设为 False，适用于 Edge 子对象级。
23. meshOps.makePlanar <editable\_mesh\_Node\_or\_Modifier>  
强制使选择的子对象共面，适用于 Vertex、Edge、Face、Polygon 和 Element 子对象级。
24. meshOps.removeIsolatedVerts <editable\_mesh\_Node\_or\_Modifier>  
删除对象里的所有孤立的顶点。
25. meshOps.selectOpenEdges <editable\_mesh\_Node\_or\_Modifier>  
选择开放边，适用于 Edge 子对象级。
26. meshOps.slice <editable\_mesh\_Node\_or\_Modifier>  
基于切片平面对边和面进行细分。在调用本命令之前必须调用 meshOps.startSlicePlane() 方法来指定切片平面，适用于所有子对象级，以及处在非子对象模式下时。
27. meshOps.tessellate <editable\_mesh\_Node\_or\_Modifier>

细分选择的子对象，适用于 Face、Polygon 和 Element 子对象级。

28. `meshOps.unhideAll <editable_mesh_Node_or_Modifier>`

取消当前子对象级别下隐藏子对象的隐藏，适用于 Vertex、Face、Polygon 和 Element 子对象级。

29. `meshOps.unifyNormal <editable_mesh_Node_or_Modifier>`

统一选择面的法线，适用于 Face、Polygon 和 Element 子对象级。

30. `meshOps.viewAlign <editable_mesh_Node_or_Modifier>`

将所有选择的子对象对齐到活动视窗的 0,0,0 平面，适用于 Vertex、Edge、Face、Polygon 和 Element 子对象级。

31. `meshOps.visibleEdge <editable_mesh_Node_or_Modifier>`

将选择边的可见性设为 True，适用于 Edge 子对象级。

32. `meshOps.weld <editable_mesh_Node_or_Modifier>`

焊接选择顶点，适用于 Vertex 子对象级。

示例：

```
s=sphere()          --先创建一个 sphere 对象
ConvertToMesh s      --将其转换成 Editable_Mesh 对象
s.selectedfaces=#{1..112}  --选择一半的面
max modify mode      --打开 Modify 命令面板
select s              --选择 sphere 对象
modPanel.setCurrentObject s --将 Modifier Stack 设为基本对象
                         --(在本例里并不需要这样)
subObjectLevel = 4      --将子对象级别设为 polygon
meshOps.startExtrude s    --开始面 extrude 命令，用户可以看到
                           --Edit Geometry rollout 下的按钮被亮显
```

下面是另一个例子，会隐藏当前对象：

```
meshOps.Hide(modPanel.getCurrentObject())
```

### 9.1.17 使用 Editable\_Mesh 的示例

示例一：下面函数会反转指定 Mesh 对象的法线。

```
fn FlipObjNormal obj =
( for f =1 to obj.numFaces do --对 Mesh 对象每一面循环
  ( verts=getface obj f      --获取面的 3 个顶点
    local tmp=verts.x        --将第 1 个顶点与第 3 个顶点交换
    verts.x=verts.z          --反转法线
    verts.z=tmp
    setface obj f verts     --将修改后的面存回 Mesh 对象
  )
  update obj                --刷新对象
)
```

示例二：下面脚本是一个使用纹理顶点和面的示例。

```
--本函数用来给对象 obj 指定一个平面贴图
```

```

--参数 direction 用来指定贴图垂直于哪一轴
--可以为下面的值: #x、#y、或 #z
--
fn ApplyPlanarMap obj direction =
( local oldcoordsys, normalize_pos
--在本例中纹理顶点的数目和 Mesh 对象的顶点数相等
    obj.numverts=obj.numverts
--建立纹理顶点面
    buildTVFaces obj
--下面的操作在 Objects 坐标系下进行,
--先存储现有坐标系, 以便操作完成后恢复它。
    oldcoordsys=set coordsys local
--对 Mesh 对象的每一顶点的.position 属性进行标准化运算: 其取值范围
--为 [0,0,0] 到 [1,1,1], 其结果为平面 UVW 坐标系
    for v = 1 to obj.numverts do
        ( normalize_pos=((getvert obj v)-obj.min)/(obj.max-obj.min)
--反转.position 各元素, 使其方向
--垂直于平面贴图
        case direction of
            (#x:( tmp=normalize_pos.x
                normalize_pos.x=normalize_pos.y
                normalize_pos.y=normalize_pos.z
                normalize_pos.z=tmp
            )
            #y:( tmp=normalize_pos.y
                normalize_pos.y=normalize_pos.z
                normalize_pos.z=tmp
            )
        )
--设置相应纹理顶点的位置
        settvert obj v normalize_pos
    )
--完成位置纹理顶点后, 建立纹理顶点面,
--因为本例中 Mesh 和纹理顶点之间存在一对一
--的关系, Mesh 对象的面上顶点的序号与
--纹理面上顶点的序号相同
    for f = 1 to obj.numfaces do
        setTVFace obj f(getface obj f)
--完成所有操作后, 恢复坐标系
    set coordsys oldcoordsys
--刷新 Mesh 对象
    update obj
)
--
--检验程序。创建一个 Sphere 对象, 施加一种 Material,
--为 2x3 平铺 Checker 类贴图, 打开视窗的 Cecker Texture Map
--显示开关, 调用 ApplyPlanarMap() 函数
--
g=geosphere()
convertToMesh g

```

```

g.Material=standard()
g.Material.maps[2]=checker()
g.Material.maps[2].coordinates.u_tiling=2
g.Material.maps[2].coordinates.v_tiling=3
showTextureMap g.Material g.Material.maps[2] True
ApplyPlanarMap g #x

```

输出：

```

ApplyPlanarMap()          --5 到 45 行函数定义输出
$GeoSphere:GeoSphere02 @ [0.0,0.0,0.0]      --第 51 行输出
$Editable_Mesh:GeoSphere02 @ [0.0,0.0,0.0]  --第 52 行输出
Standard:Standard          --第 53 行输出
Checker:Checker            --第 54 行输出
2                          --第 55 行输出
3                          --第 56 行输出
OK                         --第 57 行输出
OK                         --第 58 行输出

```

## 9.2 SplineShape: Shape

SplineShape 类是所有 Shape 类对象的通用可编辑版本（General Editable Version），两者之间的关系类似于 Editable\_Mesh 类和 Geometry 对象之间的关系：如果给一个 Shape 类基本对象赋给一个 Edit Spline Modifier，然后将 Modifier Stack 塌陷，得到的结果就是一个 SplineShape 类对象。在 MAXScript 里，可以直接构造 SplineShape 类对象，并向它添加单独的曲线或控制点，还可以先把一个 Shape 类基本对象转换成 SplineShape 类对象，然后用本节所述的方法对它进行编辑。

### 9.2.1 SplineShape 属性

下面是 SplineShape 类对象的属性：

属性名称	数据类型	默认值	说明
<SplineShape>.angle	Float	0.0	设置剖面图形沿路径轴向旋转的角度。可动画
<SplineShape>.thickness	Float	1.0	设置可渲染二维图形剖面的直径。可动画
<SplineShape>.sides	Float	12.0	设置可渲染二维图形剖面的边数。可动画
<SplineShape>.viewport_thickness	Float	1.0	设置二维图形在视窗中的剖面直径
<SplineShape>.viewport_sides	Integer	12	设置二维图形在视窗中的剖面边数
<SplineShape>.viewport_angle	Float	0.0	设置剖面图形在视窗中的沿路径轴向旋转的角度
<SplineShape>.DisplayRenderMesh	Boolean	False	当设置为 On 时，将可渲染的二维图形对象显示为网格对象
<SplineShape>.UseViewportSettings	Boolean	False	当设置为 On 时，二维图形对象可以被渲染

(续表)

属性名称	数据类型	默认值	说明
<SplineShape>.DisplayRenderSettings	Boolean	True	当设置为 On 时, 依据视图设置将可渲染的二维图形对象显示为网格对象
<SplineShape>.numSplines	Integer		在二维图形对象上独立的样条曲线数量。只读
<splineshape>.steps	Integer	6	设置每两个顶点之间短直线的数量
<splineshape>.optimize	Boolean	True	当设置为 On 时, 自动检查并移除样条曲线上多余的步数设置, 以减小样条曲线的复杂程度
<splineshape>.adaptive	Boolean	False	当设置为 On 时, 将依据样条曲线的复杂程度自动指定步数, 对于直线自动将步数设定为 0; 对于复杂的弯曲曲线, 自动加大步数使其更光滑

### 9.2.2 使用 SplineShape 方法的注意事项

在使用 SplineShape 下面这些方法之前应注意以下一些事项:

- ◆ 首先, 必须先把 Shape 类基本对象转换成 SplineShape 类对象。可以用 convertToSplineShape()方法来转换。
- ◆ 只有用 updateShape()方法才能刷新 Shape 类对象的内部缓存和 3ds max 视窗。一般来说, 刷新操作需要大量的运算, 比较费时, 没有必要在每一次方法调用后都执行刷新。但必须确保在进行一系列修改后, 在 shape 对象被 3ds max 系统使用前或在 MAXScript 被其他方法调用前执行刷新操作。
- ◆ Bezier 曲线控制点的 in 和 out 矢量格式为矢量控制柄坐标, 而不是真正的矢量。
- ◆ 所有方法使用的坐标系是 MAXScript 工作坐标系, 如果读者熟悉 SDK 里相应的 Spline 函数, 尤其应该注意这一点, 因为在 SDK 里, 经常使用 Object-Local 坐标系。
- ◆ 当命令面板处在 Modify 模式时, 对一个对象调用 updateShape()方法, 系统会自动取消对该对象的选择, 这样可以避免 3ds max 系统的崩溃, 因为当 updateShape()方法运行时, 不能在 Modify 模式下用脚本对对象进行修改。
- ◆ 如果在 3ds max 界面里创建或修改一个 Spline 对象后, 还没有对它执行 updateShape()方法, 此时如果将 3ds max 窗口最小化, 当再次将窗口最大化时, 3ds max 系统会崩溃, 这是因为在执行 updateShape()方法之前, spline 对象的系统内部数据结构尚处在不稳定的状态。
- ◆ 对一个 SplineShape 类对象, 共有 3\*N 个坐标值(顶点坐标、in 矢量、out 矢量)可以作为属性进行存取, 其中 N 为该对象的顶点数目。一旦一个顶点被赋给一个 Controller, 它的顶点和切线坐标属性就可以进行存取。可以用 animateVertex()方法将 Controller 赋给顶点。例如, 一个 Circle 类对象被塌陷成 SplineShape 类对象, 并且它的某些顶点被赋给 Controller 后, 我们可以用下面的格式对这些顶点和切线坐标进行存取:

```
$Circle01.Spline_1_InVec_1 Point3 value: [-75,33,0] --可动画
$Circle01.Spline_1_Vertex_1 Point3 value: [-75,33,0] --可动画
```

```
$Circle01.Spline_1_OutVec_1 Point3 value: [-50,0,0] --可动画
$Circle01.Spline_1_InVec_2 Point3 value: [-20,-33,0] --可动画
$Circle01.Spline_1_Vertex_2 Point3 value: [7,-66,0] --可动画
$Circle01.Spline_1_OutVec_2 Point3 value: [32,-95,0] --可动画
```

### 9.2.3 Shape 方法

#### 1. updateShape <shape>

刷新<shape>对象的内部缓存和视窗显示以反映此前对其所做的修改。

`updateShape()`方法会将在脚本里对一个基本对象所做的修改传送至当前 Modifier Stack 的底部。注意：即使这些修改会导致堆栈里已有的 Modifier 无效，系统也不会给出警告信息。

#### 2. resetShape <shape>

清除<shape>对象所有的样条曲线。

#### 3. numSplines <shape>

返回一个整数，表示<shape>对象里包含的样条曲线数目，与属性<shape>.numSplines 的值相等。

#### 4. setFirstSpline <shape> <spline\_index\_Integer>

对<shape>对象里的样条曲线重新排序，使指定样条曲线的序号成为 1。在 Loft 和 Surface Tool 等对样条曲线排序非常敏感的 Modify 里，当操作对象为包含多条样条曲线的 Shape 对象时，本方法非常有用。

#### 5. hideselectedsplines <shape>

隐藏<shape>对象里当前被选择的样条曲线。

#### 6. hideselectedsegments <shape>

隐藏<shape>对象里当前被选择的线段。

#### 7. hideselectedverts <shape>

隐藏<shape>对象里与当前被选择的顶点相连的线段。

#### 8. unhidesegments <shape>

恢复显示<shape>对象里的所有线段。

#### 9. addAndWeld <to\_shape> <from\_shape> <weldthreshold\_Float>

将<from\_shape>的样条曲线添加到 Bezier 曲线<to\_shape>里。参数<weldthreshold\_Float>表示如果新样条曲线的端点和原样条曲线的端点之间的距离小于该指定的值，系统会将它们焊接在一起。

#### 10. bindKnot <shape> <isEnd\_boolean> <splineId\_Integer> \

<segIndex\_Integer> <splineSegId\_Integer>

将指定样条曲线的起点或终点约束到指定线段的中点上。各参数说明如下：

- ◆ <shape> 要设定约束的 Shape 对象；
- ◆ <isEnd\_boolean> 指定约束哪一点：如果为 True，约束指定样条曲线终点；否则约束指定样条曲线起点；
- ◆ <splineId\_Integer> 指定要约束的终点或起点所在样条曲线的序号；

- ◆ <segIndex\_Integer> 指定约束点的线段；
- ◆ <splineSegId\_Integer> 指定约束点的样条曲线。

11. unBindKnot <shape> <spline\_index\_Integer> <isEnd\_boolean>

取消指定样条曲线的终点或起点的约束。

12. updateBindList <shape>

当拓扑结构改变（如从约束样条曲线或被约束样条曲线删除顶点）时，调用本方法以刷新约束列表。

13. MaterialID <shape> <spline\_index\_Integer> <segment\_index\_Integer>

返回<shape>对象里指定线段的材质 ID 码。

14. animateVertex <shape> <vertex\_spec>

给指定顶点或切线赋给 Controller，参数<vertex\_spec>可以是以下几种格式：

<Integer\_index>

<Integer\_index\_array>

#all

在给顶点和切线赋给 Controller 以后，这些顶点和切线都会被添加到 Master subAnim，并会在 Track View 视窗里显示为 animatable，可以用脚本对这些顶点和切线进一步设置动画。如：

```
animateVertex $Circle01 #all
```

为所有顶点赋给 Controller。

#### 9.2.4 Spline 方法

1. addNewSpline <shape>

向指定 Splineshape 添加新样条曲线，并返回一个整数，表示新添加的样条曲线的序号。样条曲线的序号从 1 开始。本方法将新的样条曲线添加到 Shape 对象原有样条曲线的末尾。

2. getSplineSelection <shape>

返回一个整数数组，表示<shape>对象里当前被选择样条曲线的序号。

3. setSplineSelection <shape> <spline\_index\_array> [ keep:<boolean> ]

选择指定 Splineshape 里由数组<spline\_index\_array>指定的样条曲线系列。如果可选参数[ keep:<boolean> ] 为 keep:True，之前选择的样条曲线会被保留；否则之前的选择会被取消选择。

4. deleteSpline <shape> <spline\_index\_Integer>

从指定 Splineshape 里删除指定样条曲线。剩下的曲线会重新进行排序。

5. numSegments <shape> <spline\_index\_Integer>

返回一个整数，表示指定样条曲线里线段的数目。如果是一个封闭样条曲线，本方法返回值与顶点数相等，如果是一个开放样条曲线，本方法返回值为顶点数减 1。

6. numKnots <shape> [ <spline\_index\_Integer> ]

返回一个整数，表示指定样条曲线里的结点数。如果没有指定样条曲线序号，返回值

为整个 Shape 对象上的结点数。

7. `isClosed <shape> <spline_index_Integer>`

如果指定样条曲线为封闭曲线，返回 True；否则，返回 False。

8. `close <shape> <spline_index_Integer>`

封闭指定样条曲线。

9. `open <shape> <spline_index_Integer>`

在指定样条曲线的最后一个结点和第一个结点之间打开指定样条曲线。

10. `reverse <shape> <spline_index_Integer>`

将指定样条曲线的结点排序逆转。

11. `setFirstKnot <shape> <spline_index_Integer> <knot_index_Integer>`

对指定样条曲线的结点重新排序，使指定结点序号为 1。

12. `getSegLengths <splineShape> <spline_index> [cum:<boolean>] \`

`[byVertex:<boolean>] [numArcSteps:<Integer>]`

返回一个数组，表示各线段长度或顶点之间的距离。

如果参数 cum: 为 True，其结果为长度累加；否则为每个单独线段的长度。

如果参数 byVertex: 为 True，数组为按顶点逐个排列；否则按线段逐个排列。

可选参数的默认值为：

cum: False

byVertex: False

numArcSteps: 100

13. `subdivideSegment <splineShape> <spline_index> <seg_index> <divisions>`

将样条曲线的指定线段分成指定数目的小线段，双精度计算，非常精确。

14. `interpCurve3D <splineShape> <spline_index> \`

`<param_Float> [pathParam:<boolean>]`

返回一个 Point3 坐标点，表示指定曲线上指定点的坐标。如果指定参数 pathParam:False (默认值)，参数<param\_Float>为样条曲线长度的分数；否则为路径长度的分数。

15. `tangentCurve3D <splineShape> <spline_index> \`

`<param_Float> [pathParam:<boolean>]`

返回一个 Point3 坐标点，表示指定曲线上的一个指定点的切线。如果指定 pathParam:False (默认值)，参数<param\_Float>为样条曲线长度的分数；否则为路径长度的分数。

### 9.2.5 Segment 方法

1. `getSegmentType <shape> <spline_index_Integer> <seg_index_Integer>`

返回指定样条曲线上指定线段的类型。

2. `setSegmentType <shape> <spline_index_Integer> \`

`<seg_index_Integer> (#curve | #line)`

设置指定样条曲线上指定线段的类型。

3. refineSegment <shape> <spline\_index\_Integer> \  
     <seg\_index\_Integer> <seg\_interp\_param\_Float>

在指定位置给指定线段添加一个新结点。参数<seg\_interp\_param\_Float>为一个 Float 数据，取值范围 0.0~1.0，指定了插入点处在沿线段的长度比例。新插入结点的坐标、in 矢量和 out 矢量都会自动计算，以保持现有曲线的弯曲形状。本方法返回值为插入点的序号。

4. getSegSelection <shape> <spline\_index\_Integer>

返回一个整数数组，表示指定样条曲线上当前被选择线段的所有顶点序号。

5. setSegSelection <shape> <spline\_index\_Integer> \  
     <segment\_index\_array> [ keep: <boolean> ]

返回指定样条曲线上由数组<segment\_index\_array>指定的所有线段。

6. setMaterialID <splineShape> <spline\_index> <seg\_index> <matID>

为指定样条曲线上指定线段设置材质 ID 码。

7. getMaterialID <splineShape> <spline\_index> <seg\_index>

返回指定样条曲线上指定线段的材质 ID 码。

### 9.2.6 Knot 方法

1. addKnot <shape> <spline\_index\_Integer> \  
     (#smooth | #corner | #bezier | #bezierCorner) \  
     (#curve | #line) <position\_point3> \  
     [invec\_point3 outvec\_point3] [where\_Integer]

向指定样条曲线添加一个结点（控制点），并返回一个整数，表示新点的序号。

第三个参数指定新点的类型，第四个参数指定了线段在该点的离开类型，第五个参数指定了新点的坐标（坐标系为当前工作坐标系），如果结点类型指定为#bezier 或 #bezierCorner，还必须指定一个 in 矢量和一个 out 矢量作为第六和第七个参数。最后一个参数指定了新结点的序号要插入到哪一个旧结点前面，默认值将新结点作为样条曲线的最后一个结点。

2. deleteKnot <shape> <spline\_index\_Integer> <knot\_index\_Integer>

删除指定样条曲线上指定结点。剩下的结点会重新排序。

3. getKnotType <shape> <spline\_index\_Integer> <knot\_index\_Integer>

返回指定样条曲线上指定结点的类型，返回值可能为以下值：#smooth、#corner、#bezier 或 #bezierCorner。

4. setKnotType <shape> <spline\_index\_Integer> <knot\_index\_Integer> \  
     (#smooth | #corner | #bezier | #bezierCorner)

为指定样条曲线上指定结点设置类型。在 3ds max 窗口下，可以在 Edit\_Spline Modifier 下通过用鼠标右击结点来为结点设置类型。

5. getKnotPoint <shape> <spline\_index\_Integer> <knot\_index\_Integer>

返回一个 Point3 值，表示指定结点的坐标值，坐标系为当前工作坐标系。

6. setKnotPoint <shape> <spline\_index\_Integer> <knot\_index\_Integer> <point3>

为指定结点设置坐标值，坐标系为当前工作坐标系。

7. `getInVec <shape> <spline_index_Integer> <knot_index_Integer>`

返回一个 Point3 值，表示指定结点的 in 矢量控制柄坐标，坐标系为当前工作坐标系。

8. `setInVec <shape> <spline_index_Integer> <knot_index_Integer> <point3>`

为指定结点的 in 矢量控制柄设置坐标，坐标系为当前工作坐标系。

9. `getOutVec <shape> <spline_index_Integer> <knot_index_Integer>`

返回一个 Point3 值，表示指定结点 out 矢量控制柄的坐标，坐标系为当前工作坐标系。

10. `setOutVec <shape> <spline_index_Integer> <knot_index_Integer> <point3>`

为指定结点 out 矢量控制柄设置坐标，坐标系为当前工作坐标系。

11. `getKnotSelection <shape> <spline_index_Integer>`

返回一个整数数组，表示<shape>样条曲线上当前被选择的结点序号。

12. `setKnotSelection <shape> <spline_index_Integer> \`

`<knot_index_array> [keep: <boolean> ]`

选择<shape>样条曲线上由数组<knot\_index\_array>指定的所有结点。

### 9.2.7 Editable\_Spline Modify 面板命令的操作方法

有一组方法可以让用户在脚本里调用 Modify 命令面板下 Editable\_Spline、Line 和 Edit Spline 的操作，并和各 Shape 子对象级下的按钮相对应。这些方法都带一个前缀.splineOps，每一个方法都相当于按下 3ds max 用户界面 Modify 命令面板里的一个按钮。如果要使用这些方法，必须先选择要操作的对象，并打开 Modify 命令面板。在这些方法的描述中，第一个变量<editable\_spline\_or\_line\_Node\_or\_Modifier>可以是一个类型为 Editable\_Spline 或 Line 的对象，同时其 Modifier Stack 处于 Base 级；也可以是一个 Edit\_Spline Modifier，同时其 Modifier Stack 里当前 Modifier 为 Edit\_Spline。所有方法均工作在当前子对象级，当然，该子对象级必须适合于要调用的操作。

如果在脚本里对 Shape 对象进行了修改，别忘了在调用这些方法之前调用 update()方法。但却不必在调用这些方法后调用 update()方法，3ds max 系统会自动进行刷新。

下面的方法会按下 Editable\_Spline Modify 面板下的一个按钮，并将当前被选择的子对象作为操作对象。调用这些命令时，相应的按钮会高亮显示，就像用鼠标按下该按钮一样，并开始完成指定的指令。

1. `splineOps.startUnion <editable_spline_or_line_Node_or_Modifier>`

相当于按下 Boolean Union 按钮，适用于 Spline 子对象级。调用本命令时只能选择一个封闭的样条曲线。

2. `splineOps.startSubtract <editable_spline_or_line_Node_or_Modifier>`

相当于按下 Boolean Subtract 按钮，适用于 Spline 子对象级。调用本命令时只能选择一个封闭的样条曲线。

3. `splineOps.intersect <editable_spline_or_line_Node_or_Modifier>`

相当于按下 Boolean Intersect 按钮，适用于 Spline 子对象级。调用本命令时只能选择一个封闭的样条曲线。

4. splineOps.startAttach <editable\_spline\_or\_line\_Node\_or\_Modifier>  
相当于按下 Attach 按钮，适用于所有子对象级和非子对象级的模式。
5. splineOps.startBind <editable\_spline\_or\_line\_Node\_or\_Modifier>  
相当于按下 Bind 按钮，适用于 Vertex 子对象级。
6. splineOps.startBreak <editable\_spline\_or\_line\_Node\_or\_Modifier>  
相当于按下 Break 按钮，适用于 Vertex 和 Segment 子对象级。
7. splineOps.startChamfer <editable\_spline\_or\_line\_Node\_or\_Modifier>  
相当于按下 Chamfer 按钮，适用于 Vertex 子对象级。
8. splineOps.startConnect <editable\_spline\_or\_line\_Node\_or\_Modifier>  
相当于按下 Connect 按钮，适用于 Vertex 子对象级。
9. splineOps.startCreateLine <editable\_spline\_or\_line\_Node\_or\_Modifier>  
相当于按下 Create Line 按钮，适用于所有子对象级和非子对象级的模式。
10. splineOps.startCrossInsert <editable\_spline\_or\_line\_Node\_or\_Modifier>  
相当于按下 CrossInsert 按钮，适用于 Vertex 子对象级。
11. splineOps.startExtend <editable\_spline\_or\_line\_Node\_or\_Modifier>  
相当于按下 Extend 按钮，适用于 Spline 子对象级。
12. splineOps.startFillet <editable\_spline\_or\_line\_Node\_or\_Modifier>  
相当于按下 Fillet 按钮，适用于 Vertex 子对象级。
13. splineOps.startInsert <editable\_spline\_or\_line\_Node\_or\_Modifier>  
相当于按下 Insert 按钮，适用于所有子对象级和非子对象级的模式。
14. splineOps.startOutline <editable\_spline\_or\_line\_Node\_or\_Modifier>  
相当于按下 Outline 按钮，适用于 Spline 子对象级。
15. splineOps.startRefine <editable\_spline\_or\_line\_Node\_or\_Modifier>  
相当于按下 Refine 按钮，适用于 Vertex 和 Segment 子对象级。本方法不会关闭 Connect 复选框，其功能与方法 startRefineConnect() 相同。
16. splineOps.startRefineConnect <editable\_spline\_or\_line\_Node\_or\_Modifier>  
相当于按下 Refine 按钮，适用于 Vertex 和 Segment 子对象级。本方法不会打开 Connect 复选框，其功能与方法 startRefine() 相同。
17. splineOps.startTrim <editable\_spline\_or\_line\_Node\_or\_Modifier>  
相当于按下 Trim 按钮，适用于 Spline 子对象级。  
下面方法调用 Editable\_Shape Modify 面板下的一些“杂项”按钮，同样将当前选择的子对象作为操作对象。
18. splineOps.attachMultiple <editable\_spline\_or\_line\_Node\_or\_Modifier>  
显示 Attach Multiple 对话框，可以选择多个 Shape 对象附加到指定对象上。
19. splineOps.close <editable\_spline\_or\_line\_Node\_or\_Modifier>  
封闭选择的样条曲线，适用于 Spline 子对象级。
20. splineOps.cycle <editable\_spline\_or\_line\_Node\_or\_Modifier>  
在 Spline 各顶点之间循环，依次选择每一个顶点。在调用本方法之前，必须至少选择

一个顶点。适用于 Vertex 子对象级。

21. splineOps.delete <editable\_spline\_or\_line\_Node\_or\_Modifier>

删除选择的子对象，适用于所有子对象级。

22. splineOps.detach <editable\_spline\_or\_line\_Node\_or\_Modifier>

显示 Detach 对话框，可以指定一个对象名，适用于所有子对象级。

23. splineOps.divide <editable\_spline\_or\_line\_Node\_or\_Modifier>

将选择线段拆分，适用于 Segment 子对象级。

24. splineOps.explode <editable\_spline\_or\_line\_Node\_or\_Modifier>

将选择的 Spline 的每一线段炸开成单独的 spline 或别的对象，适用于 Spline 子对象级。

25. splineOps.hide <editable\_spline\_or\_line\_Node\_or\_Modifier>

隐藏选择的子对象，适用于所有子对象级。

26. splineOps.makeFirst <editable\_spline\_or\_line\_Node\_or\_Modifier>

将选择的顶点的序号调整为 1，适用于 Vertex 子对象级。

27. splineOps.mirrorBoth <editable\_spline\_or\_line\_Node\_or\_Modifier>

将选择的 Spline 在水平方向和垂直方向同时镜像，适用于 Spline 子对象级。

28. splineOps.mirrorHoriz <editable\_spline\_or\_line\_Node\_or\_Modifier>

将选择的 Spline 在水平方向镜像，适用于 Spline 子对象级。

29. splineOps.mirrorVert <editable\_spline\_or\_line\_Node\_or\_Modifier>

将选择的 Spline 在垂直方向镜像，适用于 Spline 子对象级。

30. lineOps.reverse <editable\_spline\_or\_line\_Node\_or\_Modifier>

将选择的 Spline 顶点序号反转，适用于 Spline 子对象级。

31. splineOps.unbind <editable\_spline\_or\_line\_Node\_or\_Modifier>

将选择的顶点解除约束，适用于 Vertex 子对象级。

32. splineOps.unhideAll <editable\_spline\_or\_line\_Node\_or\_Modifier>

解除所有隐藏对象的隐藏，适用于所有子对象级和非子对象级的模式。

33. splineOps.weld <editable\_spline\_or\_line\_Node\_or\_Modifier>

焊接选择顶点，适用于 Vertex 子对象级。

## 9.3 Patch: GeometryClass

Editable\_Patch 类是所有 Patch 类对象的通用可编辑版本，两者之间的关系类似于 Editable\_mesh 类和 Geometry 对象之间的关系：如果给一个 Patch 类基本对象赋给一个 Edit\_Patch Modifier，然后将 Modifier Stack 塌陷，得到的结果就是一个 Editable\_Patch 类对象。在 MAXScript 里，可以直接构造 Editable\_Patch 类对象，但是目前还没有办法存取或修改 Patch 类对象的子对象（如顶点、边、面）。

### 构造函数

1. Quadpatch ...

**Tripatch ...**

创建一个 Patch 类场景对象，如下面的语句创建了一个空 Patch 对象：

```
patch pos:[10,10,10] name: "foo"
```

2. convertTo <node> Editable\_Patch        mapped

将指定的场景对象转换成一个 Editable\_Patch 对象。如果指定对象不能被转换（比如对那些非 Geometry 类对象），本方法返回 undefined。被转换对象原有的 Modifier 都会被塌陷。也可以用方法 collapseStack() 或 3ds max 用户界面上 Modifier Stack 对话框边上的 Collapse 按钮来实现转换，但在 Modifier Stack 里至少要有一个 Modifier。

**属性**

一旦一个控制点被赋给一个 Controller，它的顶点和切线坐标属性就可以进行存取。一个控制点的切线数目等于与该顶点相连的边数目。在 3ds max 里可以用 MAXScript 函数将 Controller 赋给控制点。如果一个 TriPatch 类对象被塌陷成 SplineShape 类对象，并且它的某些顶点被赋给 Controller，我们可以用下面的格式对这些控制点和切线进行存取：

```
$TriPatch01.Vertex_1                    --Point3 value: [-21.5,-6.4,0], 可动画
$TriPatch01.Vertex_1_Vector_1    --Point3 value: [-21.5,-2.1,0], 可动画
$TriPatch01.Vertex_1_Vector_2    --Point3 value: [-7.1,-6.4,0], 可动画
$TriPatch01.Vertex_2                    --Point3 value: [21.5,-6.4,0], 可动画
$TriPatch01.Vertex_2_Vector_1    --Point3 value: [7.1,-6.4,0], 可动画
$TriPatch01.Vertex_2_Vector_2    --Point3 value: [7.1,-2.1,0], 可动画
```

**方法**

1. getPatchSteps <editable\_patch\_Node>

返回指定 Patch 对象在视窗里使用的步数，与 Modify panel 的 spinner 类控件 View Steps 的值相等。

2. setPatchSteps <editable\_patch\_Node> <Integer>

将 Modify 面板下控件 View Steps 的值设为指定值。

### 9.3.1 Editable\_Patch Modify 面板命令的操作方法

有一组方法可以让用户在脚本里调用 Modify 命令面板下 Editable\_Patch 和 Edit\_Mesh 的操作，并和各 Mesh 子对象级下的按钮相对应。这些方法都带一个前缀.patchOps，每一个方法都相当于按下 3ds max 用户界面 Modify 命令面板里的一个按钮。如果要使用这些方法，必须先选择要操作的对象，并打开 Modify 命令面板。在这些方法的描述中，第一个变量<editable\_patch\_Node\_or\_Modifier>可以是一个类型为 Editable\_Patch 的对象，同时其 Modifier Stack 处于 Base 级；也可以是一个 Edit\_Patch Modifier，同时其 Modifier Stack 里当前 Modifier 为 Edit\_Patch。所有方法均工作在当前子对象级，当然，该子对象级必须适合于要调用的操作。

下面的方法相当于按下 Editable\_Patch Modify 面板下的一个按钮，并将当前被选择的子对象作为操作对象。调用这些命令时，相应的按钮会高亮显示，就像用鼠标按下该按钮

一样，并开始完成指定的指令。

1. patchOps.startAttach <editable\_patch\_Node\_or\_Modifier>

相当于按下 Attach 按钮，适用于所有子对象级和非子对象级的模式。

2. patchOps.startBevel <editable\_patch\_Node\_or\_Modifier>

相当于按下 Bevel 按钮，适用于 Patch 子对象级。

3. patchOps.startBind <editable\_patch\_Node\_or\_Modifier>

相当于按下 Bind 按钮，适用于 Vertex 子对象级。

4. patchOps.startCreate <editable\_patch\_Node\_or\_Modifier>

相当于按下 Create 按钮，适用于 Vertex、Patch 和 Element 子对象级。

5. patchOps.startExtrude <editable\_patch\_Node\_or\_Modifier>

相当于按下 Extrude 按钮，适用于 Patch 子对象级。

6. patchOps.startFlipNormalMode <editable\_patch\_Node\_or\_Modifier>

相当于按下 Flip Normal 按钮，适用于 Patch 和 Element 子对象级。

7. patchOps.startWeldTarget <editable\_patch\_Node\_or\_Modifier>

相当于按下 Weld Target 按钮，适用于 Vertex 子对象级。

下面的方法调用 Editable\_Patch Modify 面板下的一些“杂项”按钮，同样将当前选择的子对象作为操作对象。

1. patchOps.addTri <editable\_patch\_Node\_or\_Modifier>

给选择的边添加一个三角形面片，适用于 Edge 子对象级。

2. patchOps.addQuad <editable\_patch\_Node\_or\_Modifier>

给选择的边添加一个四边形面片，适用于 Edge 子对象级。

3. patchOps.break <editable\_patch\_Node\_or\_Modifier>

为每一个与选择子对象相连的 Patch 对象创建一个新顶点，适用于 Vertex 和 Edge 子对象级。

4. patchOps.clearAllSG <editable\_patch\_Node\_or\_Modifier>

为当前选择的 Patch 对象清除光滑组，适用于 Face、Polygon 和 Element 子对象级。

5. patchOps.createShapeFromEdges <editable\_patch\_Node\_or\_Modifier>

显示 Create Shape 对话框来指定一个对象名，系统用它来命名用当前选择边创建的 Shape 对象，适用于 Edge 子对象级。

6. patchOps.delete <editable\_patch\_Node\_or\_Modifier>

删除选择的子对象，适用于所有子对象级。

7. patchOps.detach <editable\_patch\_Node\_or\_Modifier>

显示 Detach 对话框，可以指定一个对象名，适用于 Patch 子对象级。

8. patchOps.flipNormal <editable\_patch\_Node\_or\_Modifier>

反转 Patch 对象的法线，适用于 Patch 和 Element 子对象级。

9. patchOps.hide <editable\_patch\_Node\_or\_Modifier>

隐藏选择的子对象，适用于所有子对象级。

10. patchOps.selectByID <editable\_patch\_Node\_or\_Modifier>

显示 Select By Material ID 对话框，可以指定一个材质 ID 码，来选择指定材质 ID 码的 Patch 对象，适用于 Patch 和 Element 子对象级。

11. patchOps.selectBySG <editable\_patch\_Node\_or\_Modifier>

显示 Select By Smooth Groups 对话框，可以指定一个光滑组 ID 码，来选择指定光滑组 ID 码的 Patch，适用于 Patch 和 Element 子对象级。

12. patchOps.selectOpenEdges <editable\_patch\_Node\_or\_Modifier>

选择所有只与一个 Patch 对象相连的边，适用于 Edge 子对象级。

13. patchOps.subdivide <editable\_patch\_Node\_or\_Modifier>

将选择的子对象分开，适用于 Edge 和 Patch 子对象级。

14. patchOps.unbind <editable\_patch\_Node\_or\_Modifier>

将选择的顶点解除约束，适用于 Vertex 子对象级。

15. patchOps.unifyNormal <editable\_patch\_Node\_or\_Modifier>

将选择 Patch 对象的法线统一，适用于 Patch 和 Element 子对象级。

16. patchOps.unhideAll <editable\_patch\_Node\_or\_Modifier>

解除所有隐藏子对象的隐藏，适用于所有子对象级和非子对象级的模式。

17. patchOps.weld <editable\_patch\_Node\_or\_Modifier>

焊接选择顶点，适用于 Vertex 子对象级。

下面的方法可用来存取 Patch 对象的一些全局性的参数。在 setNumXXX 系列方法里，设置 keep:True 会保留 Patch 对象当前的 Vertex/Vector/ Patch/Edge 数据，默认值为 False，表示不保留这些信息。

1. patch.getNumVerts <obj>

patch.setNumVerts <obj> <num> [keep:<boolean>]

存取 Patch 对象的顶点数。

2. patch.getNumVecs <obj>

patch.setNumVecs <obj> <num> [keep:<boolean>]

存取 Patch 对象的矢量数。

3. patch.getNumPatches <obj>

patch.setNumPatches <obj> <num> [keep:<boolean>]

存取 Patch 对象的网格数。

4. patch.getNumEdges <obj>

patch.setNumEdges <obj> <num> [keep:<boolean>]

存取 Patch 对象的边数。

下面这些方法用来存取单个顶点和矢量控制柄的坐标位置。这些顶点和矢量控制柄的坐标值在 MAXScript 里都是在当前工作坐标系下。与 MAXScript 里其他序号一样，Vertex/Patch/Vector/Edge 等的序号都从 1 开始。

1. patch.getVert <obj> <index>

patch.setVert <obj> <index> <point3>

存取 Patch 对象指定顶点的坐标。

2. patch.getVec <obj> <index>  
 patch.setVec <obj> <index> <point3>

存取 Patch 对象指定矢量的坐标。

下面两个方法主要用来在 Patch Mesh 里创建单个 Patch 对象。

1. patch.makeQuadPatch <obj> <index> <va> <vab> <vba> <vb> <vbc> <vc> <vc> <vcd> <vdc> <vd> <vda> <vad> <i1> <i2> <i3> <i4> <sm>

参数说明如下：

- ◆ Index 要创建 Patch 的序号，取值范围为[0 numPatches]
- ◆ Va 第一个顶点 a
- ◆ Vab 矢量 ab
- ◆ vba 矢量 ba
- ◆ vb 第二个顶点 b
- ◆ vbc 矢量 bc
- ◆ vcb 矢量 cb
- ◆ vc 第三个顶点 c
- ◆ vcd 矢量 cd
- ◆ vdc 矢量 dc
- ◆ vdT 第四个顶点 d
- ◆ vda 矢量 da
- ◆ vad 矢量 ad
- ◆ i1 内部顶点 1
- ◆ i2 内部顶点 2
- ◆ i3 内部顶点 3
- ◆ i4 内部顶点 4
- ◆ sm 光滑组蒙版

2. patch.makeTriPatch <obj> <index> <va> <vab> <vba> <vb> <vbc> <vc> <vc> <vca> <vac> <i1> <i2> <i3> <sm>

参数说明同上。

下面的方法强制所有 Geometry 和拓扑结构进行重建，同时刷新视窗。在完成对 Patch 对象的修改后，退出 MAXScript 前必须调用本方法；否则可能导致系统的崩溃。方法 patch.update()与 Mesh 对象的 update()方法类似。

patch.update <obj>

下面的方法可以用来存取 Patch 对象的现有拓扑结构。这些方法返回单个序号或序号数组。

patch.getVertVecs <obj> <index>	返回值类型： Integer_array
patch.getVertPatches <obj> <index>	返回值类型： Integer_array
patch.getVertEdges <obj> <index>	返回值类型： Integer_array
patch.getVecVert <obj> <index>	返回值类型： Integer

patch.getVecPatches <obj> <index>      返回值类型: Integer\_array

patch.getEdgeVert1 <obj> <index>      返回值类型: Integer

patch.getEdgeVert2 <obj> <index>      返回值类型: Integer

patch.getEdgeVec12 <obj> <index>      返回值类型: Integer

patch.getEdgeVec21 <obj> <index>      返回值类型: Integer

下面的方法可以用来存取 Patch 对象的贴图信息。

patch.setNumMaps <obj> <num> [keep:<bool>]

patch.getNumMaps <obj>: Integer

patch.setMapSupport <obj> <map\_chan> [init:<boolean>]

patch.getMapSupport <obj> <map\_chan>: boolean

patch.maxMapChannels <obj>

patch.setNumMapVerts <obj> <map\_chan> <num> [keep:<boolean>]

patch.getNumMapVerts <obj> <map\_index>

patch.setNumMapPatches <obj> <map\_chan> <num> [keep:<boolean>]

patch.getMapVert <obj> <map\_chan> <index>

patch.setMapVert <obj> <map\_chan> <index> <point3>

patch.getMapPatch <obj> <map\_chan> <index>

patch.setMapPatch <obj> <map\_chan> <index> <index\_array>

下面的方法可以用来存取单个或全部 Patch 对象的材质 ID 码。

patch.getMtlID <obj>      返回值类型: Integer

patch.setMtlID <obj>      返回值类型: <mtl\_id>

patch.getPatchMtlID <obj> <index>      返回值类型: Integer

patch.setPatchMtlID <obj> <patch\_index>      返回值类型: <mtl\_id>

下面的方法可以用来存取、控制视窗和渲染器的可见性。

patch.setMeshSteps <obj> <steps>

patch.getMeshSteps <obj>

patch.setMeshStepsRender <obj> <steps>

patch.getMeshStepsRender <obj>

patch.setShowInterior <obj> <bool>

patch.getShowInterior <obj>

patch.setAdaptive <obj> <bool>

patch.getAdaptive <obj>

下面的方法可以用来存取指定子对象的拓扑信息，返回值类型都为 Integer\_array。

patch.getEdges <obj> <v1> [<v2>]

patch.getPatches <obj> <v1> [<v2>]

patch.getVectors <obj> <vert>

下面的方法可以用来对 Patch 对象的所有顶点坐标进行转换，转换在对象的 Local 空间下进行。

`patch.transform <obj> <matrix3>`

下面的方法用来对 Patch 对象执行一些高级操作。

`patch.weldVerts <obj> <threshold> <weldIdentical_bool> <startVert>`

`patch.weld2Verts <obj> <v1> <v2>`

`patch.weldEdges <obj>`

`patch.deletePatchParts <obj> <del_vert_bitarray> <del_patches_bitarray>`

`patch.clonePatchParts <obj> <patch_bit_array>`

`patch.subdivideEdges <obj> <propogate>`

`patch.subdividePatches <obj> <propogate>`

`patch.addTriPatch <obj>`

`patch.addQuadPatch <obj>`

下面的方法用来获取、操作法线信息。同样，在 MAXScript 里法线矢量也在当前工作坐标系下。

`patch.patchNormal <obj> <patch>`

`patch.edgeNormal <obj> <edge>`

`patch.updatePatchNormals <obj>`

`patch.flipPatchNormal <obj> <patch>`

`patch.unifyNormals <obj>`

`patch.autoSmooth <obj> <angle> <useSel_bool> <preventIndirectSmoothing_bool>`

下面方法用来存取、改变顶点和内部顶点的类型。

`patch.changeVertType <obj> <vert> #corner | #coplanar`

`patch.getVertType <obj> <vert> #corner | #coplanar`

`patch.changePatchInteriorType <obj> <patch> #automatic | #manual`

`patch.getPatchInteriorType <obj> <patch> #automatic | #manual`

下面方法返回指定 Patch 对象的当前网格细分。

`patch.getMesh <obj>` 返回一个 Mesh 值

下面方法用来插入单个 Patch 面。插入一个三角面片 U、V、W 三个 Barycentric 坐标值，插入一个四边形面片需要 U、V 两个参数。返回点都是在 MAXScript 的当前工作坐标系下。

`patch.interpTriPatch <obj> <patch> <u> <v> <w>` 返回 Point3 值

`patch.interpQuadPatch <obj> <patch> <u> <v>` 返回 Point3 值

## 9.4 Editable\_Poly: GeometryClass

Editable\_Poly 为所有可编辑多边形对象的通用可编辑版本，两者之间关系类似于 Editable\_mesh 类和 Geometry 对象之间的关系。

## 构造函数

Editable\_Poly 类实例不能直接用 MAXScript 创建，但是将 Geometry 其他类的对象转化成 Editable\_Poly。

例如：

```
ep = editable_mesh name: (uniquename "EPoly") --创建一个 EMesh
convertTo ep (Editable_Poly) --转化成 Editable_Poly
```

## 属性

属性名称	数据类型	默认值	说明
<Editable_Poly>.autoSmoothThreshold <Editable_Poly>.Autosmooth_angle_threshold	Float	45.0	指定相邻多边形的法线之间的最大角度，用于确定这些多边形是否可包含在同一光滑组中
<Editable_Poly>.ignoreBackfacing <Editable_Poly>.Ignore_Backfacing_in_Selections	Boolean	False	当设置为 On 时，选择子对象将只影响朝向用户的那些对象
<Editable_Poly>.fullyInteractive <Editable_Poly>.Fully_Interactive	Boolean	True	
<Editable_Poly>.selByVertex <Editable_Poly>.Select_By_Vertex	Boolean	False	当设置为 On 时，只需选择子对象使用的顶点，即可选择子对象
<Editable_Poly>.split	Boolean	False	设置是否使用分割
<Editable_Poly>.bevelHeight <Editable_Poly>.Bevel_Height	Float	10.0	以场景为单位指定挤出的范围
<Editable_Poly>.bevelOutline <Editable_Poly>.Bevel_Outline	Float	-1.0	设置倒角的轮廓量
<Editable_Poly>.bevelType <Editable_Poly>.Type_of_Bevel	Integer	0	设置倒角的类型： 0: Group (组) 1: Local Normal (局部法线) 2: by Polygon (按多边形)
<Editable_Poly>.connectEdgeSegments <Editable_Poly>.Connect_Edge_Segments	Integer	1	设置连接边分段数量
<Editable_Poly>.constrainType <Editable_Poly>.Vertex_Constraint_Type	Integer	0	设置约束类型： 0: None (无约束) 1: Edge (约束顶点到边界的变换) 2: Face (约束顶点到曲面的变换)
<Editable_Poly>.cutEndCoords <Editable_Poly>.Cut_End_Coordinates	Point3	[0,0,0]	设置切割的终点坐标
<Editable_Poly>.cutNormal <Editable_Poly>.Cut_Normal_Direction	Point3	[0,0,0]	设置切割法线的向量值

(续表)

属性名称	数据类型	默认值	说明
<Editable_Poly>.cutStartCoords <Editable_Poly>. Cut_Start_Coordinates	Point3	[0,0,0]	设置切割的起点坐标
<Editable_Poly>.cutstartIndex <Editable_Poly>.Cut_Start_Index	Integer	1	设置切割开始点的序号
<Editable_Poly>.cutStartLevel <Editable_Poly>. Cut_Start_SO_Level	Integer	1	设置切割开始的级别
<Editable_Poly>.edgeChamfer <Editable_Poly>.Edge_Chamfer	Float	1.0	设置边倒角量值
<Editable_Poly>. edgeExtrudeHeight <Editable_Poly>. Edge_Extrude_Height	Float	10.0	设置边挤出高度
<Editable_Poly>.edgeExtrudeWidth <Editable_Poly>.Edge_Extrude_Width	Float	3.0	设置边挤出基面宽度
<Editable_Poly>.edgeWeldThreshold <Editable_Poly>. Edge_Weld_Threshold	Float	0.1	设置焊接阈值
<Editable_Poly>. extrudeAlongSplineAlign <Editable_Poly>. Extrude_Along_Spline_Align	Boolean	False	设置是否对齐到面法线
<Editable_Poly>. extrudeAlongSplineNode <Editable_Poly>. Extrude_Along_Spline_Node	undefined	undefined	指定样条曲线
<Editable_Poly>. extrudeAlongSplineRotation <Editable_Poly>. Extrude_Along_Spline_Rotation	Float	0.0	设置挤出的旋转
<Editable_Poly>. extrudeAlongSplineSegments <Editable_Poly>. Extrude_Along_Spline_Segments	Integer	6	设置沿多边形数挤出的分段数
<Editable_Poly>. extrudeAlongSplineTaper <Editable_Poly>. Extrude_Along_Spline_Taper_Amount	Float	0.0	设置挤出沿着其长度变小或变大的范围。锥化挤出的负设置越小，锥化挤出的正设置就越大
<Editable_Poly>. extrudeAlongSplineTaperCurve <Editable_Poly>. Extrude_Along_Spline_Taper_Curve	Float	0.0	设置继续进行的锥化率。低设置会产生渐变更大的锥化，而高设置会产生更突然的锥化
<Editable_Poly>. extrudeAlongSplineTwist <Editable_Poly>. Extrude_Along_Spline_Twist	Float	0.0	设置沿着挤出的长度应用扭曲的数量

(续表)

属性名称	数据类型	默认值	说明
<Editable_Poly>.extrusionType <Editable_Poly>.Type_of_Extrusion	Integer	0	设置挤出的类型： 0: Group (组) 1: Local Normal (局部法线) 2: by Polygon (按多边形)
<Editable_Poly>.faceExtrudeHeight <Editable_Poly>.Face_Extrude_Height	Float	10.0	以场景为单位指定挤出的数
<Editable_Poly>.hinge	Integer	0	设置是否启用从边旋转
<Editable_Poly>.hingeAngle <Editable_Poly>. Hinge_From_Edge_Angle	Float	30.0	沿着转枢旋转的数量值
<Editable_Poly>.hingeSegments <Editable_Poly>. Hinge_From_Edge_Segments	Integer	1	将多边形数指定到每个细分的挤出侧中
<Editable_Poly>.Inset_Type <Editable_Poly>.Inset_Amount	Float	1.0	以场景为单位指定插入的数
<Editable_Poly>.Inset_Type <Editable_Poly>.Inset_Type	Integer	0	设置影插入类型： 0: Group (组, 沿着多个连续的多边形进行插入) 1: By Polygon (按多边形, 独立插入每个多边形)
<Editable_Poly>.outlineAmount <Editable_Poly>.Outline_Amount	Float	-1.0	设置轮廓数量值
<Editable_Poly>.isolineDisplay <Editable_Poly>.Isoline_Display	Boolean	True	设置是否显示等值线
<Editable_Poly>.Affect_Backfacing <Editable_Poly>.Affect_Backfacing	Boolean	True	当设置为 On 时, 那些法线方向与选定子对象平均法线方向相反的、取消选择的面就会受到软选择的影响
<Editable_Poly>.bubble	Float	0.0	设置在软选择卷展栏里的曲线膨胀值。可动画
<Editable_Poly>.falloff	Float	20.0	设置在软选择卷展栏里的曲线衰减值。可动画
<Editable_Poly>.pinch	Float	0.0	设置在软选择卷展栏里的曲线收缩值。可动画
<Editable_Poly>.useSoftSel <Editable_Poly>.Use_Soft_Selection	Boolean	False	设置在软选择卷展栏里的是否使用软选择
<Editable_Poly>.ssEdgeDist <Editable_Poly>.Edge_Distance	Integer	1	设置在软选择卷展栏里的曲线边距离值。可动画
<Editable_Poly>.ssUseEdgeDist <Editable_Poly>.Use_Edge_Distance	Boolean	False	设置在软选择卷展栏里的是否使用边距离来控制选择范围
<Editable_Poly>.iterations	Integer	0	设置平滑多边形对象时所用的迭代次数。可动画

(续表)

属性名称	数据类型	默认值	说明
<Editable_Poly>.renderIterations <Editable_Poly>.Render_Iterations	Integer	0	用于选择不同的平滑迭代次数，以便在渲染时应用于对象
<Editable_Poly>.renderSmoothness <Editable_Poly>.Render_Smoothness	Float	1.0	用于选择不同的平滑度值，以便在渲染时应用于对象。可动画
<Editable_Poly>.separateByMaterial <Editable_Poly>.Separate_by_Material_ID	Boolean	False	设置是否按材质 ID 码来创建新多边形的分隔方式
<Editable_Poly>.separateBySmoothing <Editable_Poly>.Separate_by_Smoothing_Groups	Boolean	False	设置是否按光滑组来创建新多边形的分隔方式
<Editable_Poly>.sepByMats <Editable_Poly>.Separate_by_Material_ID	Boolean	False	设置是否按材质 ID 码来创建新多边形的分隔方式
<Editable_Poly>.sepBySmGroups <Editable_Poly>.Separate_by_Smoothing_Groups	Boolean	False	设置是否按光滑组来创建新多边形的分隔方式
<Editable_Poly>.showCage <Editable_Poly>.Show_Cage	Boolean	True	设置是否启用 Show Cage
<Editable_Poly>.smoothness	Float	1.0	设置视窗显示的光滑度值
<Editable_Poly>.subdivSmoothing <Editable_Poly>.Smooth_Subdivision_Result	Boolean	True	设置是否通过 NURMS 方法应用光滑
<Editable_Poly>.surfaceSmoothness <Editable_Poly>.Surface_Smoothness	Float	1.0	可动画
<Editable_Poly>.surfSubdivide <Editable_Poly>.Subdivide_Surface	Boolean	False	设置是否通过 NURMS 方法应用光滑
<Editable_Poly>.update <Editable_Poly>.Update_Type	Integer	0	设置更新的方式： 0: Always (在地形合成操作调整过程中总是自动更新) 1: When Rendering (在场景最终渲染输出时才进行场景更新) 2: Manually (手动更新)
<Editable_Poly>.useRenderIterations <Editable_Poly>.Use_Render_Iterations	Boolean	False	当设置为 On 时，用于选择不同的光滑迭代次数，以便在渲染时应用于对象
<Editable_Poly>.useRenderSmoothness <Editable_Poly>.Use_Render_Smoothness	Boolean	False	当设置为 On 时，用于选择不同的光滑度值，以便在渲染时应用于对象
<Editable_Poly>.tesselateBy <Editable_Poly>.Tessellate_By	Integer	0	设置细分类型：0: Edge; 1: Face

(续表)

属性名称	数据类型	默认值	说明
<Editable_Poly>.tessTension <Editable_Poly>.Tessellation_Tension	Float	0.0	设置细分的张力值
<Editable_Poly>.displaceAngle	Float	10.0	设置细分位移的角度。可动画
<Editable_Poly>.displaceDistance	Float	20.0	设置细分位移的距离。可动画
<Editable_Poly>.displaceEdge <Editable_Poly>.Displacement_Edge	Float	20.0	设置细分位移的边。可动画
<Editable_Poly>.displaceMaxLevels <Editable_Poly>.Displacement_Max_Levels	Integer	2	用于选择低、中或高质量的预设曲面近似值
<Editable_Poly>.displaceMaxTris <Editable_Poly>.Displacement_Max_Triangles	Integer	20000	设置最大细分级别
<Editable_Poly>.displaceMethod <Editable_Poly>.Displacement_Method	Integer	3	设置细分的方法： 0: Regular (规则) 1: Spatial (空间) 2: Curvature (曲率) 3: Spatial and Curvature (空间和曲率)
<Editable_Poly>.displaceMinLevels <Editable_Poly>.Displacement_Min_Levels	Integer	0	设置最小细分级别
<Editable_Poly>.displaceSplitMesh <Editable_Poly>.Displacement_Split_Mesh	Boolean	True	设置是否启用分割网格
<Editable_Poly>.displaceSteps <Editable_Poly>.Displacement_Steps	Integer	2	在规则细分方式下，设置细分的置换步数
<Editable_Poly>.displaceStyle <Editable_Poly>.Displacement_Style	Integer	1	设置用于细分曲面的方法： 0: Grid (使用常规栅格来细分曲面) 1: Tree (使用二进制树来细分曲面) 2: Delaunay (使用近似等边三角形来细分曲面)
<Editable_Poly>.useSubdivisionDisplacement <Editable_Poly>.Use_Displacement_Subdivision	Boolean	False	当设置为 On 时，使用细分预设和细分方法组框中指定的方法和设置，细分网格面以精确地位移贴图。否则，修改器通过移动网格中的顶点应用贴图
<Editable_Poly>.viewDependent <Editable_Poly>.View_Dependent	Boolean	False	设置是否启用 view-Dependent
<Editable_Poly>.vertexChamfer <Editable_Poly>.Vertex_Chamfer_Amount	Float	1.0	设置点的倒角量

(续表)

属性名称	数据类型	默认值	说明
<Editable_Poly>.vertexExtrudeHeight <Editable_Poly>.Vertex_Extrude_Height	Float	10.0	以场景为单位指定挤出的数
<Editable_Poly>.vertexExtrudeWidth <Editable_Poly>.Vertex_Extrude_Width	Float	3.0	以场景为单位指定挤出基面的大小
<Editable_Poly>.vertSelectBy <Editable_Poly>.Select_Vertex_By	Integer	0	设置顶点的选择方式： 0: Color (颜色) 1: Illumination (光照)
<Editable_Poly>.vertSelectionBlueRange <Editable_Poly>.Select_by_color_Blue_Range	Integer	10	指定蓝色匹配的范围
<Editable_Poly>.vertSelectionColor <Editable_Poly>.Vertex_Selection_Color	Color (color 255 255 255)		指定要匹配的颜色
<Editable_Poly>.vertSelectionGreenRange <Editable_Poly>.Select_by_color_Green_Range	Integer	10	指定绿色匹配的范围
<Editable_Poly>.vertSelectionRedRange <Editable_Poly>.Select_by_color_Red_Range	Integer	10	指定红色匹配的范围
<Editable_Poly>.weldPixels <Editable_Poly>.Weld_Pixel_Threshold	Integer	4	指定目标焊接的阈值
<Editable_Poly>.weldThreshold <Editable_Poly>.Vertex_Weld_Threshold	Float	0.1	指定焊接的阈值
<Editable_Poly>.deleteIsolatedVerts <Editable_Poly>.Delete_Isolated_Vertices	Boolean	True	设置是否删除孤立顶点
<Editable_Poly>.preserveUVs <Editable_Poly>.Preserve_UVs	Boolean	False	设置是否移除未使用的贴图顶点
<Editable_Poly>.lockSoftSel <Editable_Poly>.Lock_Soft_Selection	Boolean	False	设置是否锁定软选择
<Editable_Poly>.paintSelSize <Editable_Poly>.Brush_Size	Float	10.0	设置软选择笔刷的尺寸
<Editable_Poly>.paintSelStrength <Editable_Poly>.Brush_Strength	Float	1.0	设置软选择笔刷的强度值
<Editable_Poly>.paintSelValue <Editable_Poly>.Selection_Value	Float	1.0	设置软选择笔刷的选择值

(续表)

属性名称	数据类型	默认值	说明
<Editable_Poly>.paintDeformAxis <Editable_Poly>.Push_Pull_Direction	Integer	1	设置顶点的推或拉依据方式： 1: Original Normals (原始法线) 2: Deformed Normals (变形法线) 3: Transform Axis: X (变换轴 X) 4: Transform Axis: Y (变换轴 Y) 5: Transform Axis: Z (变换轴 Z)
<Editable_Poly>.paintDeformSize <Editable_Poly>.Deformation_Size	Float	20.0	设置圆形笔刷的半径。只有笔刷圆之内的顶点才可以变形
<Editable_Poly>.paintDeformStrength	Float	1.0	设置笔刷应用推/拉值的速率
<Editable_Poly>.paintDeformValue <Editable_Poly>.Deformation_Value	Float	10.0	确定单个推/拉操作应用的方向和最大范围。正值将顶点拉出对象曲面，而负值将顶点推出曲面
<Editable_Poly>.bridgeBias <Editable_Poly>.Bridge_Bias	Float	0.0	决定最大锥化量的位置。偏移值的范围从 -99.0 到 99.0
<Editable_Poly>.bridgeSegments <Editable_Poly>.Bridge_Segments	Integer	1	沿着桥连接的长度指定多边形的数目
<Editable_Poly>.bridgeSelected <Editable_Poly>.Bridge_Selected	Integer	1	
<Editable_Poly>.bridgeSmoothThreshold <Editable_Poly>.Bridge_Smooth_Threshold	Float	45.0	设置桥接光滑阈值。可动画
<Editable_Poly>.bridgeTaper <Editable_Poly>.Bridge_Taper	Float	0.0	设置桥宽度距离其中心变大变小的程度。负值设定将桥中心锥化的更小；正值设定将其锥化的更大
<Editable_Poly>.bridgeTarget1 <Editable_Poly>.Bridge_Target_1	Integer	0	设置桥接第一个指定边的序号
<Editable_Poly>.bridgeTarget2 <Editable_Poly>.Bridge_Target_2	Integer	0	设置桥接第二个指定边的序号
<Editable_Poly>.bridgeTwist1 <Editable_Poly>.Bridge_Twist_1	Integer	0	设置桥接的第一个扭曲量
<Editable_Poly>.bridgeTwist2 <Editable_Poly>.Bridge_Twist_2	Integer	0	设置桥接的第二个扭曲量
<Editable_Poly>.relaxAmount <Editable_Poly>.Relax_Amount	Float	0.5	设置松弛的数量值
<Editable_Poly>.relaxHoldBoundaryPoints <Editable_Poly>.Relax_Hold_Boundary_Points	Boolean	True	设置是否保留边界点
<Editable_Poly>.relaxHoldOuterPoints <Editable_Poly>.Relax_Hold_Outer_Points	Boolean	False	设置是否保留外点

(续表)

属性名称	数据类型	默认值	说明
<Editable_Poly>.relaxIterations <Editable_Poly>.Relax_Iterations	Integer	1	设置重复松弛过程的次数
<Editable_Poly>.selectAngle <Editable_Poly>.Selection_Angle	Float	45.0	设置 Select By Angle 的角度值
<Editable_Poly>.selectByAngle <Editable_Poly>.Select_by_Angle	Boolean	False	设置是否启用 Select By Angle

### 9.4.1 Editable\_Poly 方法

#### 贴图专用的方法

**注意** 通道编号从 0 开始。

1. `polyOp.setNumMaps <Poly poly> <int count> keep:<boolean=True>`

设置可用的贴图通道数目。取值范围为 0~100，贴图通道 0 和 1 分别为顶点颜色通道（Vertex Color）和默认纹理贴图通道（Default Texture Map）。如果参数 `keep:` 为 False，旧的通道信息会被丢失；如果为 True，旧的通道信息会被保留。

2. `polyOp.getNumMaps <Poly poly>`

返回一个整数，表示可用的贴图通道数目。

3. `polyOp.setMapSupport <Poly poly> <int mapChannel> <Boolean support>`

设置是否支持指定贴图通道。如果参数 `<support>` 为 True，而当前的通道支持状态为 False，系统会创建一个新的贴图面数组，其长度与 Mesh 对象的面数相同，但不会创建顶点数组；如果参数 `<support>` 为 False，而当前的通道支持状态为 True，现有的贴图通道面数组和顶点数组会从内存里清除；如果参数 `<support>` 与当前的通道支持状态相同，不执行任何操作；如果指定的通道当前还不可用，系统会自动调用方法 `SetNumMaps()`。

4. `polyOp.getMapSupport <Poly poly> <int mapChannel>`

返回指定贴图通道的支持状态。

5. `polyOp.setNumMapVerts <Poly poly> <int mapChannel> \ <int count> keep:<boolean=False>`

为指定贴图通道设置顶点数目、初始化贴图顶点数组。如果参数 `keep:` 为 False，旧的顶点信息会被丢失；如果为 True，旧的顶点信息会被保留。

6. `polyOp.getNumMapVerts <Poly poly> <int mapChannel>`

返回一个整数，表示指定贴图通道的顶点数目。

7. `polyOp.setNumMapFaces <Poly poly> <int mapChannel> \ <int count> keep:<boolean=False>`

为指定贴图通道设置面数目。如果参数 `keep:` 为 False，旧的面信息会被丢失；如果为 True，旧的面信息会被保留。建议不要使用贴图通道 0 和 1。

8. `polyOp.getNumMapFaces <Poly poly> <int mapChannel>`

返回一个整数，表示指定贴图通道的面数目。

9. polyOp.setMapVert <Poly poly> <int mapChannel> <int index> <Point3 uvw>

为指定贴图通道设置坐标系。

10. polyOp.getMapVert <Poly poly> <int mapChannel> <int index>

返回一个 Point3 值，表示指定贴图通道的坐标系。

11. polyOp.setMapFace <Poly poly> <int mapChannel> \  
                   <int map face index> <map vertex array>

为指定贴图面设置贴图顶点。

12. polyOp.getMapFace <Poly poly> <int mapChannel> <int index>

返回一个数组，表示指定贴图面的顶点序号。

13. polyOp.defaultMapFaces <Poly poly> <int mapChannel>

为顶点颜色通道分配并初始化一个基本平面贴图或白色贴图。

14. polyOp.applyUVWMap <Poly poly> \  
                   <#planar | #cylindrical | #spherical | #ball | #box> | <#face> \  
                   utile:<Float=1.0> vtile:<Float=1.0> wtile:<Float=1.0> \  
                   uflip:<boolean=False> vflip:<boolean=False> wflip:<boolean=False> \  
                   cap:<boolean=True> tm:<Matrix3=identity matrix> channel:<Integer=1>

给指定贴图通道应用指定贴图类型。参数说明如下：

- ◆ utile/vtile/wtile 表示 U/V/W 方向铺贴数目；
- ◆ uflip/vflip/wflip 当设置为 On 时，表示 U/V/W 方向被镜向；
- ◆ cap 当设置为 On 时，表示所有面法线轴角度大于 45 度的面使用平面坐标。只有当贴图类型为#cylindrical 时，该选项才有用；
- ◆ Tm 表示定义贴图空间。当每一点被贴图时，先乘以这个矩阵，然后才贴图。
- ◆ Channel 表示要贴图的贴图通道，默认值为 1。

### 与顶点颜色有关的方法

1. polyOp.getVertsByColor <Poly poly> <Color color> \

                  <Float red\_thresh><Float green\_thresh>\

                  <Float blue\_thresh>channel:<int=0>

返回一个<BitArray>值，元素为 Poly 对象的顶点序号，这些顶点的颜色在指定范围，颜色范围值应该在 0~255 之间。

2. polyOp.getVertsByColor <Poly poly> <Point3 uvw> <Float u\_thresh> \

                  <Float v\_thresh> <Float w\_thresh> channel:<int=0>

返回一个<BitArray>值，元素为 Poly 对象的顶点序号，这些顶点的 UVW 在指定范围，UVW 范围值应该在 0~1 之间。

3. polyOp.setVertColor <Poly poly> <int mapChannel> <vertlist> <Color color>

为指定通道的指定顶点设置颜色。

4. polyOp.SetFaceColor <Poly poly> <int mapChannel> <facelist> <Color color>

为使用指定面的指定通道顶点设置颜色。

### 与顶点数据通道有关的方法

1. `polyOp.setNumVDataChannels <Poly poly> <int count> keep:<boolean=True>`

设置可用的顶点数据通道数。其取值范围为 0~100。如果参数 `keep`: 为 False 或没有被指定，旧的数据通道会被丢弃；如果为 True，旧的数据通道会被保留。前面 10 个通道被系统使用，预定义的几个通道有：

- ◆ channel 1 Soft Selection (软选择)
- ◆ channel 2 Vertex weight (顶点权重，用于 NURMS MeshSmooth)
- ◆ channel 3 Vertex Alpha value (顶点 Alpha 通道数据)
- ◆ channel 4 Cornering value (用于细分时的角点数据)

2. `polyOp.getNumVDataChannels <Poly poly>`

返回一个整数，表示可用的顶点数据通道数。

3. `polyOp.setVDataChannelSupport <Poly poly>\<int vdChannel> <boolean support>`

设置是否支持指定的顶点数据通道。通道索引`<vdChannel>`从 1 开始。

4. `polyOp.getVDataChannelSupport <Poly poly> <int vdChannel>`

测试指定 Poly 对象是否支持指定顶点数据通道。

5. `polyOp.getVDataValue <Poly poly> <int vdChannel> <int vertex_index>`

返回一个浮点数，表示指定数据通道`<Integer vdChannel>`里与指定顶点`<Integer vert_index>`对应的通道数据值。

6. `polyOp.setVDataValue <Poly poly> <int vdChannel> <vertlist> <Float>`

设置指定数据通道`<Integer vdChannel>`里与指定顶点`<Integer vert_index>`对应的通道数据值。

7. `polyOp.freeVData <Poly poly> <int vdChannel>`

`polyOp.resetVData <Poly poly> <int vdChannel>`

取消指定顶点数据通道数组并关闭该顶点数据通道。

### 与 Edge Data 通道有关的方法

1. `polyOp.setNumEDataChannels <Poly poly> <int count> keep:<boolean=True>`

设置可用的 Edge Data 通道数。其取值范围为 0~10。如果参数 `keep`: 为 False 或没有指定，旧的数据通道被丢弃，如果为 True，旧的数据通道被保留。前面两个通道被系统使用：

- ◆ channel 1 Edge Kot Dta (边节点数据)
- ◆ channel 2 Edge Cease Dta (边折缝数据)

2. `polyOp.getNumEDataChannels <Poly poly>`

返回一个整数，表示可用 Edge Data 通道数。

3. `polyOp.setEDataChannelSupport <Poly poly>\<int edChannel> <Boolean support>`

设置是否支持指定的 Edge Data 通道。通道序号`<edChannel>`从 1 开始。

## 4. polyOp.getEDataChannelSupport &lt;Poly poly&gt; &lt;int edChannel&gt;

测试指定 Poly 对象是否支持指定 Edge Data 通道。

## 5. polyOp.getEDataValue &lt;Poly poly&gt; &lt;int edChannel&gt; &lt;int edge\_index&gt;

返回一个浮点数,表示指定通道<Integer edChannel>里与指定边<edge\_index>对应的通道数据值。

## 6. polyOp.setEDataValue &lt;Poly poly&gt; &lt;int edChannel&gt; &lt;edgelist&gt; &lt;Float&gt;

设置指定通道<Integer vdChannel>里与指定边<edge\_index>对应的通道数据值。

## 7. polyOp.freeEData &lt;Poly poly&gt; &lt;int edChannel&gt;

## resetEData &lt;Poly poly&gt; &lt;int edChannel&gt;

取消指定边数据通道数组并关闭该边数据通道。

#### 9.4.2 Editable\_Poly Modify 面板命令的操作方法

有一组方法可以让用户在脚本里调用 Modify 命令面板下 Editable\_Poly 对象的操作，并和各子对象级下的按钮相对应。这些方法都带一个前缀.polyops，每一个方法都相当于按下 3ds max 用户界面 Modify 命令面板里的一个按钮。如果要使用这些方法，必须先选择要操作的对象，并打开 Modify 命令面板。在这些方法的描述中，第一个变量<editable\_poly\_node\_or\_modifier>可以是一个类型为 Editable\_Poly 的对象，同时其 Modifier Stack 处于 Base 级；也可以是一个 Edit\_Poly Modifier，同时其 Modifier Stack 里当前选择为 Edit\_Poly。所有方法均工作在当前子对象级，当然，该子对象级必须适合于要调用的操作。

下面的方法会按下 Editable\_Poly Modify 面板下的一个按钮，并将当前被选择的子对象作为操作对象。调用这些命令时，相应的按钮会高亮显示，就像用鼠标按下该按钮一样，并开始完成指定的指令。

**注意** 一旦开始这些指令，这些方法就会马上返回，而不用等到这些方法执行完毕，所以接下来的脚本要非常小心，不要干扰这些指令的执行。

下面的很多方法都有一个参数<vertlist>、<facelist>或<edgelist>，指定了方法要操作的子对象。这些参数可以有下面几种形式：

- ◆ #all 该类型所有子对象；
- ◆ #selection 当前该类型被选择的子对象；
- ◆ #none 没有指定对象；
- ◆ 整数数组 每一个整数表示一个该类型子对象的序号；
- ◆ 一个 BitArray 值 每一个位表示一个该类型子对象的序号；
- ◆ 一个可转换为整数的数值 表示一个该类型子对象的序号；
- ◆ 一个 VertexSelection、EdgeSelection 或 FaceSelection 值 类型必须与子对象类型匹配。

#### 存取选择集

下面所有方法都有一个参数<poly>，指定要操作的 Poly 对象。这个参数既可以指定一

个场景对象，其.baseobject 属性为 Editable\_Poly；也可以指定一个 Editable\_Poly 对象。

1. polyOp.getVertSelection <Poly poly>  
返回一个 BitArray 值，表示当前被选择的顶点。
2. polyOp.setVertSelection <Poly poly> <vertlist>  
选择由参数<vertlist>指定的顶点。
3. polyOp.getEdgeSelection <Poly poly>  
返回一个 BitArray 值，表示当前被选择的边。
4. polyOp.setEdgeSelection <Poly poly> <edgelist>  
选择由参数<edgelist>指定的边。
5. polyOp.getFaceSelection <Poly poly>  
返回一个 BitArray 值，表示当前被选择的面。
6. polyOp.setFaceSelection <Poly poly> <facelist>  
选择由参数<facelist>指定的面。

#### 按标记获取对象

1. polyOp.getVertsByFlag <Poly poly> <int flag> mask:<int maskflag>

Poly 对象的每一顶点都包含一个 32 位标记变量。本方法返回一个 BitArray 值，每一位表示一个顶点的标记位值。

参数<maskflag>的默认值为 0。

顶点各标记位分别为：

- ◆ 位 1 顶点是否被选择
- ◆ 位 2 顶点是否是“死”的
- ◆ 位 3 系统保留位
- ◆ 位 4 指示顶点所在面是否指向当前视窗的“背面”
- ◆ 位 5~24 系统保留位
- ◆ 位 25~32 可由用户使用

示例：

```
theObj = plane isSelected:True
convertto theObj editable_poly
nVerts = theObj.numverts
bit30 = bit.set 0 30 True
bit31 = bit.set 0 31 True
bit32 = bit.set 0 32 True
--为前面约 3/4 顶点设置一个标记位
(
flagsToSet = bit30
flagsToMask = 0
vertexSet = #{1..(3*nVerts/4) }
theObj.setVertexFlags vertexSet flagsToSet \
flagMask:flagsToMask generateUndoRecord:True
ok
```

```

)
--为中间一些顶点设置另一个标记位
(
flagsToSet = bit31
flagsToMask = 0
vertexSet = #{(nVerts/3)..(nVerts-4)}
theObj.setVertexFlags vertexSet flagsToSet \
flagMask:flagsToMask generateUndoRecord:True
ok
)
--为最后两个顶点设置另一个标记位
(
flagsToSet = bit32
flagsToMask = 0
vertexSet = #{(nVerts-2)..nVerts}
theObj.setVertexFlags vertexSet flagsToSet \
flagMask:flagsToMask generateUndoRecord:True
ok
)
(
--
format "30: %\n" (polyOp.getvertsByFlag theObj bit30)
format "31: %\n" (polyOp.getvertsByFlag theObj bit31)
format "32: %\n" (polyOp.getvertsByFlag theObj bit32)
format "30 and !31: %\n" (polyOp.getvertsByFlag \
theObj bit30 mask:(bit30+bit31))
format "!31: %\n" (polyOp.getvertsByFlag theObj bit30 mask:(bit31))
format "30 and 31: %\n" (polyOp.getvertsByFlag theObj(bit30+bit31))
format "! (31 or 32) : %\n" (polyOp.getvertsByFlag \
theObj bit30 mask:(bit31+bit32))
)

```

## 2. polyOp.getVertFlags <Poly poly> <int vert>

返回指定顶点的标记值。

## 3. polyOp.setVertFlags <Poly poly> <vertlist> \

<int flag> mask:<int=0> undoable:<boolean=False>

将指定顶点的指定标记位设置为指定值<flag>。

示例：

```

p=convertToPoly(plane())
p.selectedverts = #{10..20}
i=1      --选择位
j=bit.set 0 30 True  --设置位 30
oldflags = for k = 1 to p.numverts collect polyop.getVertFlags p k
theSet=#{5..15}
polyop.setvertflags p theSet j mask:i
for k = 1 to p.numverts do
(
nf=polyop.getVertFlags p k

```

```

format "%: %: %\n" k nf oldflags[k]
oldflags[k]=nf
)
j=bit.set 0 1 True --设置位 1
polyop.setvertflags p theSet j
for k = 1 to p.numverts do
(
nf=polyop.getVertFlags p k
format "%: %: %\n" k nf oldflags[k]
)

```

#### 4. polyOp.getEdgesByFlag <Poly poly> <int flag> mask:<int maskflag>

Poly 对象的每一边都包含一个 32 位标记变量。本方法返回一个 BitArray 值，每一位表示一个边的标记位值。

参数<maskflag>的默认值为 0。

边各标记位分别为：

- ◆ 位 1 边是否被选择
- ◆ 位 2 边是否是“死”的
- ◆ 位 3 系统保留位
- ◆ 位 4 指示边所在面是否指向当前视窗的“背面”
- ◆ 位 5~24 系统保留位
- ◆ 位 25~32 可由用户使用

#### 5. polyOp.getEdgeFlags <Poly poly> <int edge>

返回指定边的标记位值。

#### 6. polyOp.setEdgeFlags <Poly poly> <edgelist> <int flag> \ mask:<int=0> undoable:<boolean=False>

将指定边的指定标记位设置为指定值<flag>。

#### 7. polyOp.getFacesByFlag <Poly poly> <int flag> mask:<int maskflag>

Poly 对象的每一面都包含一个 32 位标记变量。本方法返回一个 BitArray 值，每一位表示一个面的标记位值。

参数<maskflag>的默认值为 0。

面各标记位分别为：

- ◆ 位 1 面是否被选择
- ◆ 位 2 面是否是“死”的
- ◆ 位 3 系统保留位
- ◆ 位 4 指示面是否指向当前视窗的“背面”
- ◆ 位 5~24 系统保留位
- ◆ 位 25~32 可由用户使用

#### 8. polyOp.getFaceFlags <Poly poly> <int face>

返回指定面的标记位值。

9. polyOp.setFaceFlags <Poly poly> <facelist> <int flag> \  
 mask:<int=0> undoable:<boolean=False>

将指定面的指定标记位设置为指定值<flag>。

#### 获取 Poly 对象的顶点、边和面的数目

1. polyOp.getNumVerts <Poly poly>

返回 Poly 对象里的顶点数目，包括“死”的顶点。

2. polyOp.getNumEdges <Poly poly>

返回 Poly 对象里的边数目，包括“死”的边。

3. polyOp.getNumFaces <Poly poly>

返回 Poly 对象里的面数目，包括“死”的面。

#### 存取 Poly 对象的隐藏状态

1. polyOp.getHiddenVerts <Poly poly>

返回一个 BitArray 值，表示对应序号顶点被隐藏。

2. polyOp.setHiddenVerts <Poly poly> <vertlist>

隐藏 Poly 对象里的指定顶点。

3. polyOp.getHiddenFaces <Poly poly>

返回一个 BitArray 值，表示对应序号的面被隐藏。

4. polyOp.setHiddenFaces <Poly poly> <facelist>

隐藏 Poly 对象里的指定面。

5. polyOp.unHideAllFaces <Poly poly>

显示 Poly 对象里所有隐藏面。

6. polyOp.unHideAllVerts <Poly poly>

显示 Poly 对象里所有隐藏顶点。

#### 指定 Poly 对象边的可见性

1. polyOp.getEdgeVis <Poly poly> <int edge>

如果指定边为可见，返回 True，否则返回 False。

2. polyOp.setEdgeVis <Poly poly> <edgelist> <boolean>

为指定边设置可见性。

#### 获取 Poly 对象的开放边

1. polyOp.getOpenEdges <Poly poly>

返回一个 BitArray 值，表示对应序号的边为开放边（仅被一个面使用）。

2. polyOp.getBorderFromEdge <Poly poly> <int edge>

返回一个 BitArray 值，表示对应序号边通过开放边与指定边相连接。Poly 对象的所有开放边形成一个链，首尾相连，组成了 Poly 对象的边界。

### 按面获取对象

1. `polyOp.getFaceVerts <Poly poly> <int face>`

返回一个数组，包含指定面的所有顶点。数组里顶点的排列顺序与顶点在面里的顺序相同。

2. `polyOp.getFaceEdges <Poly poly> <int face>`

返回一个数组，包含指定面的所有边。数组里边的排列顺序与边在面里的顺序相同。

3. `polyOp.getFaceDeg <Poly poly> <int face>`

返回指定面里的顶点数。

### 按边获取对象

1. `polyOp.getEdgeVerts <Poly poly> <int edge>`

返回一个二元数组，表示指定边的两个顶点。

2. `polyOp.getEdgeFaces <Poly poly> <int edge>`

返回一个一元（开放边）或二元数组，表示使用指定边的面，

### 获取“死”子对象

1. `polyOp.getDeadVerts <Poly poly>`

返回一个 BitArray 值，表示当前“死”顶点的序号，一般情况下返回一个空的 BitArray 值。

2. `polyOp.getDeadEdges <Poly poly>`

返回一个 BitArray 值，表示当前“死”边的序号，一般情况下返回一个空的 BitArray 值。

3. `polyOp.getDeadFaces <Poly poly>`

返回一个 BitArray 值，表示当前“死”面的序号，一般情况下返回一个空的 BitArray 值。

4. `polyOp.GetHasDeadStructs <Poly poly>`

返回一个 BitArray 值，如果位 1 被打开，表示指定 Poly 对象里有“死”的顶点，如果位 2 被打开，表示指定 Poly 对象里有“死”的边，如果位 3 被打开，表示指定 Poly 对象里有“死”的面。

5. `polyOp.isFaceDead <Poly poly> <int face>`

如果指定面为“死”面，返回 True，否则返回 False。

6. `polyOp.isEdgeDead <Poly poly> <int edge>`

如果指定边为“死”边，返回 True，否则返回 False。

7. `polyOp.isVertDead <Poly poly> <int vert>`

如果指定顶点为“死”顶点，返回 True，否则返回 False。

8. `polyOp.CollapseDeadStructs <Poly poly>`

从指定 Poly 对象里删除所有“死”的子对象。

### 存取面材质 ID 码

1. polyOp.getFaceMatID <Poly poly> <int face>  
返回指定面的材质 ID 码。
2. polyOp.setFaceMatID <Poly poly> <facelist> <int MatID>  
给指定面赋材质 ID 码。

### 操作顶点

1. polyOp.getVert <Poly poly> <int vertex> node:<node=unsupplied>  
返回指定顶点的坐标值。如果参数<poly>为一个 Node 对象或 Editable\_Poly 对象，且参数<node>被指定，返回坐标值在当前坐标系下；如果参数<poly>为一个 Editable\_Poly 对象，且参数<node>没有被指定，返回坐标值在 Poly 对象的 Local 坐标系下。

2. polyOp.setVert <Poly poly> <vertlist> \  
 <point3 pos> node:<node=unsupplied>  
为指定顶点设置坐标值。参照坐标系同 polyOp.getVert 方法。

3. polyOp.moveVert <Poly poly> <vertlist> \  
 <point3 offset> node:<node=unsupplied>

将指定顶点移动指定距离<offset>。如果参数<poly>为一个 Node 对象或 Editable\_Poly 对象，且参数<node>被指定，<offset>所处坐标系为当前坐标系；如果参数<poly>为一个 Editable\_Poly 对象，且参数<node>没有被指定，<offset>所处坐标系为 Poly 对象的 Local 坐标系。

### 填充 Mesh

1. polyOp.isMeshFilledIn <Poly poly>  
如果指定 Poly 对象的拓扑链接（如边列表）是完整的，返回 True。
2. polyOp.fillInMesh <Poly poly>  
重新计算 Poly 对象内部拓扑结构。

### 由 A 获取 B

1. polyOp.getEdgesUsingVert <Poly poly> <vertlist>  
返回一个 BitArray 值，元素为 Poly 对象的边序号，表示这些边使用参数<vertlist>里指定的一个顶点。
2. polyOp.getFacesUsingVert <Poly poly> <vertlist>  
返回一个 BitArray 值，元素为 Poly 对象的面序号，表示这些面使用参数<vertlist>里指定的一个顶点。
3. polyOp.getVertsUsingEdge <Poly poly> <edgelist>  
返回一个 BitArray 值，元素为 Poly 对象的顶点序号，表示这些顶点使用参数<edgelist>里指定的一个边。
4. polyOp.getFacesUsingEdge <Poly poly> <edgelist>  
返回一个 BitArray 值，元素为 Poly 对象的面序号，表示这些面使用参数<edgelist>里指

定的一个边。

5. `polyOp.getvertsUsingFace <Poly poly> <facelist>`

返回一个 BitArray 值，元素为 Poly 对象的顶点序号，表示这些顶点使用参数`<facelist>`里指定的一个面。

6. `polyOp.getedgesUsingFace <Poly poly> <facelist>`

返回一个 BitArray 值，元素为 Poly 对象的边序号，表示这些边使用参数`<facelist>`里指定的一个面。

7. `polyOp.getelementsUsingFace <Poly poly> <facelist> fence:<fence_facelist>`

返回一个 BitArray 值，元素为 Poly 对象的面序号，表示这些面与参数`<facelist>`里指定的面处在同一个“元件”里。

如果指定参数`<fence_facelist>`，该参数里指定的面会跳过这种运算。比如参数`<facelist>`里包含序号为 i 的面，如果它周边的面被指定在参数`<fence_facelist>`里，本方法运行时会跳过这些面。

8. `polyOp.getvertsUsedOnlyByFaces <Poly poly> <facelist>`

返回一个 BitArray 值，元素为 Poly 对象的顶点序号，表示这些顶点仅仅被参数`<facelist>`里的面使用。

### 获取面的中心、法线和面积

1. `polyOp.getFaceCenter <Poly poly> <int face> node:<node=unsupplied>`

计算指定面的中心坐标，计算方法为其所有顶点的坐标平均值。

如果参数`<poly>`为一个 Node 对象或 Editable\_Poly 对象，且参数`<node>`被指定，坐标值所处坐标系为当前坐标系；如果参数`<poly>`为一个 Editable\_Poly 对象，且参数`<node>`没有被指定，坐标值所处坐标系为 Poly 对象的 Local 坐标系。

2. `polyOp.getSafeFaceCenter <Poly poly> <int face> node:<node=unsupplied>`

计算指定面的安全中心坐标（如果可能的话）。对有凸起的面，用方法 `getFaceCenter()` 返回的面中心在一些场合下不适用，因为该中心可能处在面以外，或处在一个不能看到全部顶点的区域。如果安全中心找不到，返回 `undefined`。

本方法提供了一种计算有凸起外壳的面中心的更好方法。

如果参数`<poly>`为一个 Node 对象或 Editable\_Poly 对象，且参数`<node>`被指定，返回值所处坐标系为当前坐标系；如果参数`<poly>`为一个 Editable\_Poly 对象，且参数`<node>`没有被指定，返回值所处坐标系为 Poly 对象的 Local 坐标系。

3. `polyOp.getFaceNormal <Poly poly> <int face> node:<node=unsupplied>`

返回指定面的法线。

如果参数`<poly>`为一个 Node 对象或 Editable\_Poly 对象，且参数`<node>`被指定，返回的方向矢量所处坐标系为当前坐标系；如果参数`<poly>`为一个 Editable\_Poly 对象，且参数`<node>`没有被指定，返回的方向矢量所处坐标系为 Poly 对象的 Local 坐标系。

4. `polyOp.getFaceArea <Poly poly> <int face>`

返回指定面的面积。

**Attach (附加)**

```
polyOp.attach <Poly poly> <source> \
```

```
targetNode:<node=unsupplied> sourceNode:<node=unsupplied>
```

将源对象<source>（如有必要先将其转化为一个 Polymesh 对象）连接到指定的对象<poly>上。如果<source>或<poly>不是 Node 对象，必须指定相应的 targetNode 或 sourceNode 参数。源对象<source>在连接操作完成会后被删除。

**Delete (删除)**

1. polyOp.deleteVerts <Poly poly> <vertlist>

删除指定顶点。

2. polyOp.deleteFaces <Poly poly> <facelist> delIsoverts:<boolean=True>

删除指定面。如果可选参数 delIsoverts: 为 True，所有被这些面使用的顶点同时被删除。

3. polyOp.deleteEdges <Poly poly> <edgelist> delIsoverts:<boolean=True>

删除指定边。如果参数 delIsoverts: 为 True，所有被这些边使用的顶点同时被删除。

从 3ds max 5 开始，执行上面的三个方法，会同时删除所有使用要被删除元件的子对象。

示例：

```
fn polyop_deleteEdges obj which delIsoverts:=
(
if(MaxVersion())[1] >= 5000 then
(
local bit30 = bit.set 0 30 True
polyop.setEdgeFlags obj which bit30
obj.EditablePoly.remove selLevel:#edge flag:bit30
if delIsoverts == True do polyop.deleteIsoverts obj
)
else
polyop.deleteEdges obj which delIsoverts:delIsoverts
)
```

4. polyOp.deleteIsoverts <Poly poly>

删除孤立的顶点。

**Weld (顶点焊接)**

1. polyOp.weldVertsByThreshold <Poly poly> <vertlist>

将指定顶点中彼此距离小于指定阈值的顶点焊接在一起，距离阈值由 Editable\_Poly 对象的 WeldThreshold 属性指定。只有处在对象边沿的顶点可被焊接。

2. polyOp.weldVerts <Poly poly> <int vert1> <int vert2> \

```
<point3 location> node:<node=unsupplied>
```

焊接指定的两个顶点，生成的顶点坐标为参数<location>指定的值。只有处在对象边沿的顶点可被焊接。

如果参数<poly>为一个 Node 对象或 Editable\_Poly 对象，且参数<node>被指定，参数

`location` 所处坐标系为当前坐标系；如果参数`<poly>`为一个 `Editable_Poly` 对象，且参数`<node>`没有被指定，参数 `location` 所处坐标系为 `Poly` 对象的 Local 坐标系。

### 3. `polyOp.weldEdgesByThreshold <Poly poly> <edgelist>`

将参数`<edgelist>`指定边使用的所有顶点中彼此距离小于指定阈值的顶点焊接在一起。只有处在对象边沿的顶点可被焊接。

### 4. `polyOp.weldEdges <Poly poly> <int edge1> <int edge2>`

将指定边使用的所有顶点焊接在一起。只有处在对象边沿的顶点可被焊接。

## Create (创建)

### 1. `polyOp.createVert <Poly poly> <point3 pos> node:<node=unsupplied>`

在指定位置创建一个顶点。

如果参数`<poly>`为一个 `Node` 对象或 `Editable_Poly` 对象，且参数`<node>`被指定，参数`pos` 所处坐标系为当前坐标系；如果参数`<poly>`为一个 `Editable_Poly` 对象，且参数`<node>`没有被指定，参数`pos` 所处坐标系为 `Poly` 对象的 Local 坐标系。返回值为新顶点的序号，如果顶点创建不成功，返回 `undefined`。

### 2. `polyOp.createEdge <Poly poly> <int vert1> <int vert2>`

用指定的两个顶点创建一条边。指定顶点必须被同一个面使用。返回值为新边的序号，如果创建不成功，返回 `undefined`。

### 3. `polyOp.createPolygon <Poly poly> <vertex array>`

用指定顶点创建一个面。顶点的顺序与参数`<vertex array>`里顶点的顺序相同。返回值为新面的序号，如果创建不成功，返回 `undefined`。

## Auto-Smooth (自动光滑)

### `polyOp.autoSmooth <Poly poly>`

为指定 `Poly` 对象执行一次自动光滑操作。面的光滑角阈值由 `Poly` 对象的`.autoSmoothThreshold` 属性指定。

## Triangulation (三角划分)

### 1. `polyOp.flipNormals <Poly poly> <facelist>`

反转指定面的法线。

### 2. `polyOp.retriangulate <Poly poly> <facelist>`

对指定面重新进行三角划分。

### 3. `polyOp.setDiagonal <Poly poly> <int face> <int face_vert1> <int face_vert2>`

将指定面的一个对角线设为指定的两顶点的连线。指定面的级数必须大于 3，且指定两顶点不能有边连接。

## Subdivisions (细分)

### 1. `polyOp.forceSubdivision <Poly poly>`

在下一次视窗刷新时强制对 `Poly` 对象的曲面进行一次细分刷新。

## 2. polyOp.meshSmoothByVert &lt;Poly poly&gt; &lt;vertlist&gt;

对使用指定顶点的面进行光滑处理。

## 3. polyOp.meshSmoothByFace &lt;Poly poly&gt; &lt;facelist&gt;

对指定的面进行光滑处理。

## 4. polyOp.meshSmoothByEdge &lt;Poly poly&gt; &lt;edgelist&gt;

对使用指定边的面进行光滑处理。

**Face Smoothing Groups (面光滑组)**

## 1. polyOp.setFaceSmoothGroup &lt;Poly poly&gt; &lt;facelist&gt; \

<int smoothing\_group> add:<boolean=False>

为指定面设置光滑组。如果参数<add>为 True, <smoothing\_group>被添加到指定面现有的光滑组数据之后；如果为 False, 现有的光滑组数据被新的光滑组数据所覆盖。

## 2. polyOp.getFaceSmoothGroup &lt;Poly poly&gt; &lt;int face&gt;

返回一个整数，表示指定面的光滑组数据。

**Break/Divide/Split (断开/拆分/分割)**

## 1. polyOp.breakVerts &lt;Poly poly&gt; &lt;vertlist&gt;

对参数<vertlist>里的每一个顶点，在相同位置创建 N-1 个新顶点（N 为使用该顶点的面数目），每一个面使用一个不同的顶点。

## 2. polyOp.divideEdge &lt;Poly poly&gt; &lt;int edge&gt; &lt;Float fraction&gt;

沿指定边的指定比例处拆分边。返回值为新顶点的序号。如果没有顶点被创建，返回 undefined。

## 3. polyOp.divideFace &lt;Poly poly&gt; &lt;int face&gt; \

<point3 pos> node:<node=unspecified>

拆分指定面。在面上最接近参数 pos 指定的位置会创建新顶点。如果指定位置在指定面以外，新顶点在指定面的中心。

如果参数<poly>为一个 Node 对象或 Editable\_Poly 对象，且参数<node>被指定，参数 pos 所处坐标系为当前坐标系；如果参数<poly>为一个 Editable\_Poly 对象，且参数<node>没有被指定，参数 pos 所处坐标系为 Poly 对象的 Local 坐标系。

返回值为新顶点的序号。

## 4. polyOp.splitEdges &lt;Poly poly&gt; &lt;edgelist&gt;

将指定边在中间分割。

**Collapse (塌陷)**

## 1. polyOp.collapseVerts &lt;Poly poly&gt; &lt;vertlist&gt;

将<vertlist>里每一簇顶点塌陷为单个的顶点。一簇顶点指一群互相连接的顶点。

## 2. polyOp.collapseFaces &lt;Poly poly&gt; &lt;facelist&gt;

将<facelist>里每一簇面的顶点塌陷为单个的顶点。一簇面指一群互相连接的面。

## 3. polyOp.collapseEdges &lt;Poly poly&gt; &lt;edgelist&gt;

将`<facelist>`里每一簇边的顶点塌陷为单个的顶点。一簇边指一群互相连接的边。

### 标记传送

```
polyOp.propagateFlags <Poly poly> <toSOLevel> <toFlag_int> \
<fromSOLevel> <fromFlag_int> ampersand:<boolean=False> \
set:<boolean=True> undoable:<boolean=True>
```

在子对象之间传送标记。

其中参数 `toSOLevel/fromSOLevel = {#object | #vertex | #edge | #face}`

### Tessellate（细分）

1. `polyOp.tessellateByVert <Poly poly> <vertlist>`

细分使用指定顶点的面。使用 Poly 对象的`.tessellateBy` 和`.tessTension` 属性。

2. `polyOp.tessellateByFace <Poly poly> <facelist>`

细分指定面。使用 Poly 对象的`.tessellateBy` 和`.tessTension` 属性。

3. `polyOp.tessellateByEdge <Poly poly> <edgelist>`

细分使用指定边的面。使用 Poly 对象的`.tessellateBy` 和`.tessTension` 属性。

4. `polyOp.detachFaces <Poly poly> <facelist> \`

`delete:<boolean=True> asNode:<boolean=False> \`

`name:<string="Object01"> node:<node=unsupplied>`

从 Poly 对象分开指定面。如果操作成功，返回 True。

如果参数 `delete:` 为 True，指定面被分开后被删除；如果为 False，指定面被分开后不会被删除。

如果参数 `asNode:` 为 True，分开的面作为一个独立的 Node 对象；如果为 False，分开的面作为 Poly 对象的一个子对象。

如果参数 `asNode:` 为 True，参数 `name:` 指定新 Node 对象的`.name` 属性。

如果参数 `asNode:` 为 True，且`<poly>` 为 Editable\_Poly 对象（不是 Node 对象），必须指定参数`<node>`。

下面脚本将一个 EditablePoly 对象分解成一些元件：

```
macroScript PolyToElements category: "Help_Examples"
(
on isEnabled return
selection.count==1 and \
classof selection[1].baseobject == Editable_Poly
on execute do
(
obj = selection[1]
for p = polyop.getNumFaces obj to 1 by -1 do
polyOp.detachFaces obj #{p}
)
)
```

**Detach (分离)**

```
1. polyOp.detachEdges <Poly poly> <edgelist> delete:<boolean=True> \
    asNode:<boolean=False> name:<string="Object01"> \
    node:<node=unsupplied>
```

将被指定边使用的面分离。

- ◆ 如果参数 delete: 为 True， 面在分开后被删除；如果 delete: 为 False， 面不会被删除；
- ◆ 如果参数 asNode: 为 True， 面在分开后会作为一个独立的 Node 对象；如果 asNode: 为 False， 面在分开后仍作为 Poly 对象的一个元件；
- ◆ 如果参数 asNode: 为 True， 参数 name: 指定了新 Node 对象的.name 属性；
- ◆ 如果参数 asNode: 为 True， 且<poly>为 Editable\_Poly (不是 Node 对象)， 必须指定参数<node>；
- ◆ 如果操作成功， 返回 True。

```
2. polyOp.detachVerts <Poly poly> <vertlist> delete:<boolean=True> \
    asNode:<boolean=False> name:<string="Object01"> \
    node:<node=unsupplied>
```

将被指定顶点使用的面分离。

参数说明同 detachEdges 方法。

**Slice Plane (切片平面)**

```
1. polyOp.resetSlicePlane <Poly poly>
```

重置切片平面。

```
2. polyOp.getSlicePlane <Poly poly> size:<&size_Float_var> node:<node=unsupplied>
```

返回一个 Ray 值，表示切片平面的坐标和方向矢量。如果有指定可选参数 size:，切片平面的大小会在变量<&size\_Float\_var>里返回。

如果<poly>为一个 Node 对象，或如果<poly>为一个 Editable\_Poly 对象且有指定参数<node>，返回的 Ray 值处在当前坐标系下；如果<poly>为一个 Editable\_Poly 对象而没有指定参数<node>，返回的 Ray 值处在 Poly 对象的 Local 坐标系下。

```
3. polyOp.setSlicePlane <Poly poly> <ray plane_and_dir> \
    <Float size>node:<node=unsupplied>
```

将切片平面的坐标和方向矢量设置为参数<plane\_and\_dir>指定的值，大小设置为参数<size>指定的值。

如果<poly>为一个 Node 对象，或如果<poly>为一个 Editable\_Poly 且有指定参数<node>，参数<plane\_and\_dir>处在当前坐标系下；如果<poly>为一个 Editable\_Poly 而没有指定参数<node>，参数<plane\_and\_dir>处在 Poly 对象的 Local 坐标系下。

```
4. polyOp.slice <Poly poly> <facelist> <ray plane_and_dir> \
    node:<node=unsupplied>
```

用参数<plane\_and\_dir>指定的平面和方向矢量对 Poly 对象进行切片操作。如果切片成

功，返回 True。

如果<poly>为一个 Node 对象，或如果<poly>为一个 Editable\_Poly 且有指定参数<node>，参数<plane\_and\_dir>处在当前坐标系下；如果<poly>为一个 Editable\_Poly 而没有指定参数<node>，参数<plane\_and\_dir>处在 Poly 对象的 Local 坐标系下。

#### 5. polyOp.inSlicePlaneMode <Poly poly>

如果 Poly 对象正处在切片（Slice）模式下，返回 True，否则返回 False。

### Cut（剪切）

#### 1. polyOp.cutVert <Poly poly> <int start\_vert> <point3 destination> \ <point3 projdir> node:<node=unsupplied>

剪切指定顶点。如果顶点被创建，返回值为新顶点的序号；否则，返回 undefined。

如果<poly>为一个 Node 对象，或如果<poly>为一个 Editable\_Poly 且有指定参数<node>，参数 destination 和 projdir 处在当前坐标系下；如果<poly>为一个 Editable\_Poly 而没有指定参数<node>，参数 destination 和 projdir 处在 Poly 对象的 Local 坐标系下。

#### 2. polyOp.cutFace <Poly poly> <int face> <point3 start> \ <point3 destination> <point3 projdir> node:<node=unsupplied>

剪切指定面。如果顶点被创建，返回值为新顶点的序号；否则，返回 undefined。

如果<poly>为一个 Node 对象，或如果<poly>为一个 Editable\_Poly 且有指定参数<node>，参数 start、destination 和 projdir 处在当前坐标系下；如果<poly>为一个 Editable\_Poly 而没有指定参数<node>，参数 start、destination 和 projdir 处在 Poly 对象的 Local 坐标系下。

#### 3. polyOp.cutEdge <Poly poly> <int edge1> <Float prop1> <int edge2> \ <Float prop2> <point3 projdir> node:<node=unsupplied>

剪切指定边。如果顶点被创建，返回值为新顶点的序号；否则，返回 undefined。

如果<poly>为一个 Node 对象，或如果<poly>为一个 Editable\_Poly 且有指定参数<node>，参数 projdir 处在当前坐标系下；如果<poly>为一个 Editable\_Poly 而没有指定参数<node>，参数 projdir 处在 Poly 对象的 Local 坐标系下。

### Cap Hole（补漏洞）

#### 1. polyOp.capHolesByVert <Poly poly> <vertlist>

由指定顶点<vertlist>的边界来创建面以执行补漏洞操作。如果操作成功，返回 True。

#### 2. polyOp.capHolesByFace <Poly poly> <facelist>

由关联指定面<facelist>的边界来创建面以执行补漏洞操作。如果操作成功，返回 True。

#### 3. polyOp.capHolesByEdge <Poly poly> <edgelist>

由关联指定边<edgelist>的边界来创建面以执行补漏洞操作。如果操作成功，返回 True。

### Make Planar（平面化）

#### 1. polyOp.makeVertsPlanar <Poly poly> <vertlist>

移动指定顶点<vertlist>，使其在同一个平面上。

#### 2. polyOp.moveVertsToPlane <Poly poly><vertlist> <Point3 planeNormal> \ <float distance>

<Float planeOffset> node:<node>

将指定顶点<vertlist>移动到指定平面上。

3. polyOp.makeEdgesPlanar <Poly poly>

移动关联指定边<edgelist>的顶点，使其在同一个平面上。

4. polyOp.moveEdgesToPlane <Poly poly> <edgelist> <Point3 planeNormal>\

<Float planeOffset> node:<node=unsupplied>

将关联指定边<edgelist>的顶点移动到指定平面上。

5. polyOp.makeFacesPlanar <Poly poly> <facelist>

移动关联指定面<facelist>的顶点，使其在同一个平面上

6. polyOp.moveFacesToPlane <Poly poly> <facelist> <Point3 planeNormal>\

<Float planeOffset> node:<node=unsupplied>

将关联指定边面<facelist>的顶点移动到指定平面上。

### 从边创建 Shape 对象

polyOp.createShape <Poly poly> <edgelist> smooth:<boolean=False>\

name:<string="Shape01"> node:<node=unsupplied>

从指定的边<edgelist> 创建 shape 对象。

如果参数 smooth: 为 True， Shape 对象为一条光滑曲线，新对象的名字由参数 name: 指定。

如果<poly>为一个 Editable\_Poly 对象（非 Node 对象），参数 node: 必须被指定。

### Extrude/Bevel/Chamfer（挤出/倒角/切角）

1. polyOp.extrudeFaces <Poly poly> <facelist> <Float amount>

对指定面<facelist>挤出指定数量。

2. polyOp.bevelFaces <Poly poly> <facelist> <Float height> <Float outline>

为指定面<facelist>执行倒角操作。首先挤出指定的高度，然后由<outline>缩放新的顶面。

3. polyOp.chamferVerts <Poly poly> <vertlist> <Float amount>

对指定顶点<vertlist>使用指定的数量执行切角操作。

4. polyOp.chamferEdges <Poly poly> <edgelist> <Float amount>

对指定边<edgelist>使用指定的数量执行切角操作。

# 第10章 Modifier（对象空间修改器）和 Spacewarp（世界空间修改器）

## 10.1 Modifier: MAXWrapper

可以用方法 `addModifier()` 或 `modPanel.addModToSelection()` 创建 `Modifier` 和 `Spacewarp` `Modifier` 类并将它们添加到 `Modifier` 堆栈里，除非特别说明，单词 `Modifier` 的意思包含这两大类。

通过创建一个 `Modifier` 并将它添加到几个对象里，用户可以在对象间共享一个 `Modifier`。就像在 3ds max 界面里，我们可以将一个 `Modifier` 赋给一个对象集合。

有两种方法可以存取现有的 `Modifier`:

第一种方法：作为一个属性，格式如`<Node>.。`当用户存取一个场景对象的一个属性，如`.name` 或`.position` 时，MAXScript 也会将 `Modifier` 视为对象的一个属性，如：

<code>\$Box1.heightsegs</code>	-- 获取创建参数
<code>\$Box1.twist</code>	-- 获取 twist <code>Modifier</code>
<code>\$Box1.twist.angle</code>	-- twist 角度

如果在 `Modifier` 的名称里有空格，可以用连字符“-”代替。如：

<code>\$Box1.uvw_map</code>	-- 获取 UVW Map <code>Modifier</code>
-----------------------------	-------------------------------------

这种方法比较简单，但经常存在下面的情况：几个 `Modifier` 有着相同的名称，或 `Modifier` 的名称与创建参数同名。如果仍然使用这种方法，系统总是返回第一个 `Modifier`。这时可以用第二种方法。

另一种方法是：使用数组机制存取现有的 `Modifier`。先获取一个 `Modifier` 数组，然后用序号对 `Modifier` 进行索引。如：

<code>\$Box1.Modifiers</code>	-- 获取 <code>Modifier</code> 数组
<code>\$Box1.Modifiers[3]</code>	-- 获取第三个 <code>Modifier</code>
<code>\$Box1.Modifiers[#twist]</code>	-- 获取名称为 twist 的 <code>Modifier</code>
<code>\$Box1.Modifiers["ffd 4x4x4"]</code>	-- 获取 FFD <code>Modifier</code>

从上面例子可以看出，我们既可以用 `Name` 值，也可以用字符串作为 `Modifier` 的名称来引用 `Modifier`，这一点是用第一种方法做不到的。

`Modifier` 在数组里的排序可以在 `Modify` 面板的 `Modify stack` 里看到，排序从 1 开始。

`Modifier` 类和 `SpacewarpModifier` 类是直接由 `MAXWrapper` 类派生而来，并继承了该类的属性和方法，这些属性和方法参见 7.1 节“`MAXWrapper` 的通用属性和方法”。

## 10.2 Modifier 通用属性和方法

### 属性

所有的 Modifier 都有下面的属性：

1. <Modifier>.name: string

存取 Modifier 的名称。

2. <Modifier>.enabled: Boolean

用来控制指定 Modifier 的 Enabled/Disabled 状态，可读/写属性，如：

```
$foo.Modifiers[2].enabled = False --关闭第二个Modifier
```

3. <Modifier>.enabledInViews : Boolean

用来控制指定 Modifier 在视窗和渲染时是否显示。

### 相关方法

一个场景对象的 Modifier 堆栈可以用下面的方法进行操作：

1. validModifier(<Node> | <objectset>) <Modifier>

测试指定的 Modifier 是否可以被添加到指定对象或对象集合，如果可以，返回 True，否则返回 False。

2. addModifier <Node> <Modifier> [before:index] mapped

向指定对象添加指定 Modifier 的实例 (instance)。可选参数 before: 用来指定插入到 Modifier 堆栈指定的 Modifier 之前的对象 (从堆栈顶部开始计数)。如果指定对象当前正被选择并在 Modify 面板的某一子对象级打开，添加的 Modifier 会被应用到子对象集合上。用户可以用 3ds max 系统变量 subObjectLevel 来测试、设置当前的子对象级，如：

```
max modify mode --打开Modify面板
select $Box01 --在Modify面板下选择Box01对象
subObjectLevel = 3 --将子对象级设为Face
addModifier $Box01(ffd_2x2x2()) --给Face添加FFD Modifier
```

如果参数 before: 指定的位置为无效位置，该 Modifier 会被放到离指定位置最近的有效位置。比如：如果用户试图将一个 SpacewarpModifier 类对象放到 Modifier 类对象的下面，系统会将 SpacewarpModifier 类对象放在所有其他 SpacewarpModifier 类对象的下面，但是在所有 Modifier 类对象的上面。

如果参数 <Node> 为一个对象集合，该集合里每一个对象都会放置一个指定 Modifier 的实例 (instance)。如果要像在 3ds max 界面里一样将一个 Modifier 施加给一个对象集合，可以使用函数 modPanel.addModToSelection()。

函数 addModifier() 的一个缺点是：它有时不能正确地将 Modifier 赋给子对象集合，建议用户在这种情况下使用函数 modPanel.addModToSelection()。

3. deleteModifier <Node> <Modifier\_or\_index>

从 Modifier 堆栈里删除指定 Modifier。

## 4. collapseStack &lt;Node&gt; mapped

塌陷 Modifier 堆栈，得到与指定对象相对应的可编辑的基对象，如 Editable Mesh、Editable Spline、NURBS Surface 等。如果当调用本函数时 Modifier 堆栈里还没有 Modifier，不会执行任何操作。如果用户想强制将对象的堆栈塌陷或转换成某一指定类，应该使用转换函数，如 convertToMesh()、convertToSplineShape()等。

## 5. getModContextTM &lt;Node&gt; &lt;Modifier&gt;

返回一个 Matrix3 值，表示指定 Modifier 子对象 gizmo 使用的转换矩阵。

## 6. getModContextBBoxMin &lt;Node&gt; &lt;Modifier&gt;

## getModContextBBoxMax &lt;Node&gt; &lt;Modifier&gt;

返回 Point3 值，表示指定 Modifier 绑定框的坐标最小点和最大点，坐标系为 World 坐标系。

### 10.3 Modifier 子对象转换属性

在 MAXScript 里，Modifier 子对象转换属性（如 gizmo 的中心、位置、转角、缩放比例等）所处的坐标系并不是当前工作坐标系，而是由它作用的对象定义的 Local 坐标系。这一点与场景对象的转换属性是不同的。在 MAXScript 里用这些属性获取的值与在 Track View 视窗里相应轨道里看到的是一样的。如果要将这些值的坐标系转换到 World 坐标系，可以将这些值乘以对象的 objecttransform 矩阵；如果要将一个坐标值从 World 坐标系转换到对象的 Local 空间，可以将 World 坐标乘以对象的 objecttransform 矩阵的转置矩阵（Reverse）。如：

```
obj=Box pos:[10,20,30]
addModifier obj(affect_region())
objTM=obj.objecttransform
modTM=getModContextTM obj obj.affect_region
--计算终点的World坐标
obj.affect_region.end_point *(inverse modTM) * objTM
--将终点的World坐标设为 [20,20,0]
obj.affect_region.end_point = [20,20,0] * modTM *(inverse objTM)
```

FFD Modifier 的控制点子对象属性在 MAXScript 里也可以进行存取。控制点属性名与在 Track View 里看到的一样，但要把空格换成“-”符，如：

```
$fooffd_2x2x2.control_point_1
$bazffd_3x3x3.control_point_32 = [1,2,3]
```

存取 FFD Modifier 控制点时，有几点需要注意：

1. 控制点只有被用户手工或用 MAXScript 函数 animateVertex() 或 animateAll() 赋给 Controller 以后，才可以被存取。只有在被赋给 Controller 之后，控制点才会被创建，此后才可以给控制点设置关键帧。

2. 控制点坐标的坐标系为 FFD Lattice 空间，这是一个标准化的空间，原点[0,0,0]在左下角（控制点编号为 1），点[1,1,1]为右上角。所以用户在设置控制点坐标时要特别小心。

如果用户使用表达式控制器（Expression Controller）来控制 FED，也必须采用 FFD Lattice 坐标。系统函数 getModContextBBoxMin() 和 getModContextBBoxMax() 可以返回 FFD Lattice 坐标在 World 坐标系的位置。

下面的示例演示了如何存取、转换 FFD 控制点的坐标：

```
b=Box pos:[10,20,0]
ffd=ffdbox()
addModifier b ffd
animateVertex ffd 64
cp64posL=ffd.control_point_64
objTM=b.objecttransform
modTM=(getModContextTM b ffd)*ffd.lattice_transform.value
modBBMin=getModContextBBoxMin b ffd
modBBMax=getModContextBBoxMax b ffd
cp64posW=(modBBMin+(cp64posL*(modBBMax-modBBMin)))* \
(inverse modTM)* objTM
```

## 10.4 对象空间修改器分类

Modifier（对象空间修改器）有以下类型：

- ◆ Affect\_Region（影响区域修改器）
- ◆ Bend（弯曲修改器）
- ◆ Bevel（倒角修改器）
- ◆ Bevel\_Profile（倒角截面修改器）
- ◆ CameraMap（摄影机贴图修改器）
- ◆ Cap\_Holes（补洞修改器）
- ◆ CrossSection（交叉连线修改器）
- ◆ DeleteMesh（删除网格修改器）
- ◆ DeletePatch（删除面片修改器）
- ◆ DeleteSpline（删除样条线修改器）
- ◆ Disp\_Approx（置换近似修改器）
- ◆ Displace\_Mesh（位移修改器）
- ◆ Edit\_Mesh（编辑网格修改器）
- ◆ Edit\_Patch（编辑面片修改器）
- ◆ Edit\_Poly（编辑多边形修改器）
- ◆ Edit\_Spline（编辑样条线修改器）
- ◆ Extrude（挤出修改器）
- ◆ Face\_Extrude（面挤出修改器）
- ◆ FFDBox（自由形式变形长方体修改器）
- ◆ FFDCyl（自由形式变形圆柱体修改器）

- ◆ FFD\_2x2x2 (自由形式变形)
- ◆ FFD\_3x3x3 (自由形式变形)
- ◆ FFD\_4x4x4 (自由形式变形)
- ◆ FFD\_Select (自由形式变形选择修改器)
- ◆ Fillet\_Chamfer (圆角/切角修改器)
- ◆ Flex (柔体修改器)
- ◆ HSDS\_Modifier (HSDS 修改器)
- ◆ HSDSObject (HSDS 对象修改器)
- ◆ Lathe (车削修改器)
- ◆ Lattice (晶格修改器)
- ◆ Linked\_XForm (链接变换修改器)
- ◆ LS\_Mesh (LS 网格修改器)
- ◆ MaterialByElement (按元素分配材质修改器)
- ◆ Material (材质修改器)
- ◆ Melt (融化修改器)
- ◆ MeshSmooth (网格平滑修改器)
- ◆ Mesh\_Select (网格选择修改器)
- ◆ Mirror (镜像修改器)
- ◆ Morpher (变形器修改器)
- ◆ MultiRes (多分辨率修改器)
- ◆ NCurve\_Sel (NURBS 曲线选择修改器)
- ◆ Noise (噪波修改器)
- ◆ Normalize\_Spl (规格化样条线修改器)
- ◆ NormalModifier (法线修改器)
- ◆ NSurf\_Sel (NURBS 曲面选择修改器)
- ◆ Optimize (优化修改器)
- ◆ PatchDeform (面片变形修改器)
- ◆ Patch\_Select (面片选择修改器)
- ◆ PathDeform (路径变形修改器)
- ◆ Point\_Cache (点缓存修改器)
- ◆ Poly\_Select (多边形选择修改器)
- ◆ Preserve (保留修改器)
- ◆ Push (推动修改器)
- ◆ Relax (松弛修改器)
- ◆ Ripple (涟漪修改器)
- ◆ Skew (倾斜修改器)
- ◆ Skin (蒙皮修改器)
- ◆ Skin\_Morph (蒙皮变形修改器)

- ◆ Skin\_Wrap（蒙皮包裹修改器）
- ◆ Skin\_Wrap\_Patch（蒙皮包裹面片修改器）
- ◆ Slice（切片修改器）
- ◆ Smooth（平滑修改器）
- ◆ Spherify（球形化修改器）
- ◆ Spline\_IK\_Control（样条线 IK 控制修改器）
- ◆ SplineSelect（样条线选择修改器）
- ◆ Squeeze（挤压修改器）
- ◆ STL\_Check（STL 检查修改器）
- ◆ Stretch（拉伸修改器）
- ◆ Subdivide（细分修改器）
- ◆ Substitute（替代修改器）
- ◆ Surface（曲面修改器）
- ◆ SurfDeform（曲面变形修改器）
- ◆ Symmetry（对称修改器）
- ◆ Taper（锥化修改器）
- ◆ Tesselate（细化修改器）
- ◆ Trim\_Extend（修剪/延伸修改器）
- ◆ TurboSmooth（涡轮平滑修改器）
- ◆ Turn\_to\_Mesh（转化为网格修改器）
- ◆ Turn\_to\_Patch（转化为面片修改器）
- ◆ Turn\_to\_Poly（转化为多边形修改器）
- ◆ Twist（扭曲修改器）
- ◆ Unwrap\_UVW（展开 UVW 修改器）
- ◆ UVWMap（UVW 贴图修改器）
- ◆ UVW\_XForm（UVW 变换修改器）
- ◆ Vertex\_Colors（顶点颜色修改器）
- ◆ Vertex\_Weld（顶点焊接修改器）
- ◆ VertexPaint（顶点绘制修改器）
- ◆ VolumeSelect（体积选择修改器）
- ◆ Wave（波浪修改器）
- ◆ XForm（变换修改器）

#### 10.4.1 Affect\_Region: Modifier（影响区域修改器）

##### 构造函数

Affect\_Region...

## 属性

属性名称	数据类型	默认值	说明
<Affect_Region>.falloff	Float	20.0	从 gizmo 箭头的底部设置影响顶点半径的单位数目。可动画
<Affect_Region>.Pinch	Float	0.0	在箭头尖端与曲线的切线的相交位置对切线产生影响。正的值会产生一个凸起的点而负的值会产生一个陷窝。可动画
<Affect_Region>.Bubble	Float	0.0	改变受影响顶点的曲率。值为 1.0 会产生一个半圆。减小此值时，圆的边会更加倾斜。负的值会将 gizmo 箭头底部下面的曲线降低。可动画
<Affect_Region>.ignoreBackfacing	Boolean	Off	当设置为 On 时，仅影响面法线指向 gizmo 箭头所指方向的顶点。可动画
<Affect_Region>.Start_point	Point3	[0,0,0]	设置影响区域修改编辑器方向箭头的基准点。可动画
<Affect_Region>.end_point	Point3	[0,0,25]	设置影响区域修改编辑器方向箭头的尖端点。可动画

## 方法

AffectRegionVal <distance> <falloff> <pinch> <bubble>

标准 affect region 函数，本函数为一个三次曲线，如果<distance>为 0，返回 1；如果<distance>大于<falloff>，返回 0；如果<distance>值在 0 和<falloff>之间，返回一个 0 到 1 之间的数。

### 10.4.2 Bend: Modifier (弯曲修改器)

#### 构造函数

Bend ...

BendMod ...

#### 属性

属性名称	数据类型	默认值	说明
<Bend>.angle	Float	0.0	指定弯曲的角度。可动画
<Bend>.direction	Float	0.0	指定弯曲相对于水平面的方向。可动画
<Bend>.axis	Integer	2	指定弯曲操作依据的轴向，该轴向是弯曲网格对象的局部坐标系轴向，不是选择实体的坐标轴向： 0: X; 1: Y; 2: Z

(续表)

属性名称	数据类型	默认值	说明
<Bend>.limit	Boolean	False	设置是否使用限定效果
<Bend>.upperlimit <Bend>.Upper_Limit	Float	0.0	指定弯曲操作的上限。可动画
<Bend>.lowerlimit <Bend>.Lower_Limit	Float	0.0	指定弯曲操作的下限。可动画
<Bend>.center	Point3	[0,0,0]	可以移动、旋转、放缩 gizmo 中心，还可以为 gizmo 中心指定变换动画。可动画
<Bend>.gizmo	SubAnim		可以移动、旋转、放缩 gizmo，还可以为 gizmo 指定变换动画
<Bend.gizmo>.position	Point3	[0,0,0]	设置 gizmo 的位置。可动画
<Bend.gizmo>.rotation	Quat	(quat 0 0 0 1)	设置 gizmo 的旋转角度。可动画
<Bend.gizmo>.scale	Point3	[1,1,1]	设置 gizmo 的放缩因子。可动画

### 10.4.3 Bevel: Modifier（倒角修改器）

#### 构造函数

Bevel ...

#### 属性

属性名称	数据类型	默认值	说明
<Bevel>.Cap_Bottom	Integer	1	设置是否在结束端加底盖: 0: Off; 1: On
<Bevel>.Cap_Top	Integer	1	设置是否在开始端加顶盖: 0: Off; 1: On
<Bevel>.Cap_Type	Integer	0	设置封顶类型: 0: Morph (端面类型适合于变形编辑) 1: Grid (创建网格类型的端面, 其渲染效果比较好)
<Bevel>.Side_Type	Integer	0	设置边的类型: 0: Linear Sides (指定在两个级别间的线段插补采用线性计算方式) 1: Curved Sides (指定在两个级别间的线段插补采用贝塞尔曲线的计算方式)
<Bevel>.segments	Integer	1	指定在每个层级之间的分段精度。可动画
<Bevel>.Smooth_Levels	Integer	0	当设置为 On 时, 对倒角对象的边进行光滑处理, 但是端面不被同时光滑处理: 0: Off; 1: On
<Bevel>.Produce_Mapping_Coords	Integer	0	设置是否为倒角编辑后的对象指定贴图坐标: 0: Off; 1: On
<Bevel>.Keep_Lines_From_Crossing	Integer	0	当设置为 On 时, 可以防止尖锐的边角与其自身相交, 该操作在尖角部位插入一个节点使尖角变平。0: Off; 1: On
<Bevel>.Separation_Between_Lines	Float	1.0	设置边线之间保持的距离。可动画

(续表)

属性名称	数据类型	默认值	说明
<Bevel>.Starting_Outline	Float	0.0	指定与原始图形的偏移距离, 设置为 0, 则以原始图形作为基础, 正值使轮廓变大; 负值使轮廓缩小。可动画
<Bevel>.Level_1_Height	Float	0.0	指定第一层与基准层之间的距离。可动画
<Bevel>.Level_1_Outline	Float	0.0	指定第一层与基准层的偏移距离。正值使轮廓变大; 负值使轮廓缩小。可动画
<Bevel>.Use_Level_2	Integer	0	设置是否加入第二层: 0: Off; 1: On
<Bevel>.Level_2_Height	Float	0.0	指定第一层与第二层之间的距离。可动画
<Bevel>.Level_2_Outline	Float	0.0	指定第一层与第二层的偏移距离。正值使轮廓变大; 负值使轮廓缩小。可动画
<Bevel>.Use_Level_3	Integer	0	设置是否加入第三层: 0: Off; 1: On
<Bevel>.Level_3_Height	Float	0.0	指定第三层与与第二层之间的距离。可动画
<Bevel>.Level_3_Outline	Float	0.0	指定第三层与第二层的偏移距离。正值使轮廓变大; 负值使轮廓缩小。可动画

#### 10.4.4 Bevel\_Profile: Modifier (倒角截面修改器)

##### 构造函数

Bevel\_Profile...

##### 属性

属性名称	数据类型	默认值	说明
<Bevel_Profile>.Produce_Mapping_Coords	Integer	0	设置是否为轮廓倒角编辑后的对象指定贴图坐标: 0: Off; 1: On
<Bevel_Profile>.Cap_Bottom	Integer	1	设置是否在结束端加底盖: 0: Off; 1: On
<Bevel_Profile>.Cap_Top	Integer	1	设置是否在开始端加顶盖: 0: Off; 1: On
<Bevel_Profile>.Cap_Type	Integer	0	设置封顶类型: Morph: 端面类型适合于变形编辑 Grid: 创建网格类型的端面, 其渲染效果比较好
<Bevel_Profile>.Keep_Lines_From_Crossing	Integer	0	当设置为 On 时, 可以防止尖锐的边角与其自身相交, 该操作在尖角部位插入一个节点使尖角变平: 0: Off; 1: On
<Bevel_Profile>.Separation_Between_Lines	Float	1.0	设置边线之间保持的距离。可动画
<Bevel_Profile>.profile_gizmo	SubAnim		可以移动、旋转、放缩 gizmo, 还可以为 gizmo 指定变换动画
<Bevel_Profile>.Profile_Gizmo>.position	Point3	[0,0,0]	设置 gizmo 的位置。可动画
<Bevel_Profile>.Profile_Gizmo>.rotation	Quat	(quat 0 0 0 1)	设置 gizmo 的旋转角度。可动画
<Bevel_Profile>.Profile_Gizmo>.scale	Point3	[1,1,1]	设置 gizmo 的放缩因子。可动画

#### 10.4.5 CameraMap: Modifier（摄影机贴图修改器）

##### 构造函数

`CameraMap...`

##### 属性

`.cameraNode: node:` 可读写

`.channel: Integer:` 可读写

#### 10.4.6 Cap\_Holes: Modifier（补洞修改器）

##### 构造函数

`Cap_Holes ...`

##### 属性

属性名称	数据类型	默认值	说明
<code>&lt;Cap_Holes&gt;.Smooth_New_Faces</code>	Integer	1	设置是否将所有新创建的面指定为一个光滑组。0: Off; 1: On
<code>&lt;Cap_Holes&gt;.Smooth_With_Old_Faces</code>	Integer	0	当设置为 On 时，将新建面与相邻的已有面指定为一个光滑组，如果漏洞过大，应当同时打开 Smooth New Faces 复选框。 0: Off; 1: On
<code>&lt;Cap_Holes&gt;.Make_All_New_Edges_Visible</code>	Integer	0	设置新建面的边缘是否可见： 0: Off; 1: On

#### 10.4.7 CrossSection: Modifier（交叉连线修改器）

##### 构造函数

`CrossSection ...`

##### 属性

除通用属性外，CrossSection 类没有额外的属性。

#### 10.4.8 DeleteMesh: Modifier（删除网格修改器）

##### 构造函数

`DeleteMesh ...`

##### 属性

除通用属性外，DeleteMesh 类没有额外的属性。

### 10.4.9 DeleteSplineModifier: Modifier (删除样条线修改器)

#### 构造函数

`DeleteSplineModifier ...`

#### 属性

除通用属性外, `DeleteSplineModifier` 类没有额外的属性。

### 10.4.10 Disp\_Approx: Modifier (置换近似修改器)

#### 构造函数

`Disp_Approx ...`

#### 属性

除通用属性外, `Disp_Approx` 类没有额外的属性。

### 10.4.11 Displace: Modifier (位移修改器)

#### 构造函数

`Displace ...`

#### 属性

属性名称	数据类型	默认值	说明
<code>&lt;Displace&gt;.strength</code>	Float	0.0	设置位移修改器的强度。如果设置为 0.0, 位移修改器不起作用。可动画
<code>&lt;Displace&gt;.decay</code>	Float	0.0	设置位移修改器作用范围的衰减量。如果设置为 0.0, 位移修改器的作用强度是一致的。可动画
<code>&lt;Displace&gt;.lumCenterEnable</code>	Boolean	False	设置是否指定亮度中心
<code>&lt;Displace&gt;.center</code> <code>&lt;Displace&gt;.lumCenter</code>	Float	0.5	设置明度的中点值。默认位移修改器的明度中心为 50% 的灰度, 依据贴图图像像素点的灰度级别, 明度高于 128 的像素点表现凸起的效果; 明度低于 128 的像素点表现下凹的效果。可动画
<code>&lt;Displace&gt;.bitmap</code>	Bitmap	undefined	指定位移修改器的位图文件
<code>&lt;Displace&gt;.map</code>	TextureMap	undefined	指定位移修改器的纹理贴图文件
<code>&lt;Displace&gt;.blur</code>	Float	0.0	设置虚化或柔化位移修改器的效果。可动画
<code>&lt;Displace&gt;.maptype</code>	Integer	0	设置贴图方式: 0: Planar (平面贴图方式) 1: Cylindrical (柱面贴图方式) 2: Spherical (球面贴图的方式) 3: Shrink Wrap (收缩包裹的贴图方式)

(续表)

属性名称	数据类型	默认值	说明
<Displace>.cap	Boolean	False	如果设置为 On，贴图会从圆柱的端部投影到封底上
<Displace>.length	Float	1.0	设置置换贴图 gizmo 的长度。可动画
<Displace>.width	Float	1.0	设置置换贴图 gizmo 的宽度。可动画
<Displace>.height	Float	1.0	设置置换贴图 gizmo 的高度。可动画
<Displace>.U_Tile	Float	1.0	设置贴图在 U 轴方向的重复次数。可动画
<Displace>.U_Flip	Boolean	False	设置贴图坐标是否在 U 轴方向上反转
<Displace>.V_Tile	Float	1.0	设置贴图在 V 轴方向的重复次数。可动画
<Displace>.V_Flip	Boolean	False	设置贴图坐标是否在 V 轴方向上反转
<Displace>.W_Tile	Float	1.0	设置贴图在 W 轴方向的重复次数。可动画
<Displace>.W_Flip	Boolean	False	设置贴图坐标是否在 W 轴方向上反转
<Displace>.useMap	Boolean	False	当设置为 On 时，使用对象原有的贴图设置，如果原先对象没有贴图设置，则不能设置该参数
<Displace>.applyMap	Boolean	False	当设置为 On 时，指定贴图坐标到关联对象上，可以将材质贴图指定到使用相同贴图坐标设置的对象上
<Displace>.Map_or_Vertex_Color	Integer	0	设置用于位移修改器的通道是使用贴图通道还是使用节点色彩的通道： 0: Map Channel（可以指定 UVW 通道用来贴图） 1: Vertex Color Channel（顶点颜色通道）
<Displace>.Map_Channel	Integer	1	指定用来进行贴图投影的贴图通道
<Displace>.axis	Integer	2	指定位移修改器操作依据的轴向： 0: X; 1: Y; 2: Z
<Displace>.gizmo	SubAnim		可以移动、旋转、放缩 gizmo，还可以为 gizmo 指定变换动画
<Displace.Gizmo>.position	Point3	[0,0,0]	设置 gizmo 的位置。可动画
<Displace.Gizmo>.rotation	Quat	(quat 0 0 -1 0)	设置 gizmo 的旋转角度。可动画
<Displace.Gizmo>.scale	Point3	[1,1,1]	设置 gizmo 的放缩因子。可动画

**注意** 属性.gizmo 只有在 Displace Modifier 被应用到一个对象后才可以被存取。

#### 10.4.12 Edit\_Mesh: Modifier（编辑网格修改器）

##### 构造函数

Edit\_Mesh ...

**属性**

除通用属性外，Edit\_Mesh 类没有额外的属性。

**10.4.13 Edit\_Patch: Modifier (编辑面片修改器)****构造函数**

Edit\_Patch ...

**属性**

除通用属性外，Edit\_Patch 类没有额外的属性。

**10.4.14 Edit\_Spline: Modifier (编辑样条线修改器)****构造函数**

Edit\_Spline ...

**属性**

除通用属性外，Edit\_Spline 类没有额外的属性。

**10.4.15 Extrude: Modifier (挤出修改器)****构造函数**

Extrude ...

**属性**

属性名称	数据类型	默认值	说明
<Extrude>.amount	Float	25.0	设置挤压出的厚度。可动画
<Extrude>.segs <Extrude>.segments	Integer	1	设置挤压后厚度方向的精度。可动画
<Extrude>.capStart	Boolean	True	当设置为 On 时，在挤压的开始端创建一个顶盖：0: Off; 1: On
<Extrude>.capEnd	Boolean	True	当设置为 On 时，在挤压的结束端创建一个顶盖：0: Off; 1: On
<Extrude>.capType	Integer	0	设置封顶类型： <b>Morph:</b> 端面类型适合于变形编辑 <b>Grid:</b> 创建网格类型的端面，其渲染效果比较好
<Extrude>.output	Integer	1	设置挤压对象的输出类型： 0: Patch (面片对象) 1: Mesh (网格对象) 2: NURBS (NURBS 对象)

(续表)

属性名称	数据类型	默认值	说明
<Extrude>.matIDs	Boolean	True	当设置为 On 时, 为挤压生成的对象指定 ID 号码, 顶盖的材质 ID 号码为 1; 底面的材质 ID 号码为 2; 侧面的材质 ID 号码为 3
<Extrude>.useShapeIDs	Boolean	False	当设置为 On 时, 使用二维图形的材质 ID 号码作为被挤压生成对象材质 ID 号码
<Extrude>.smooth	Boolean	True	当设置为 On 时, 对挤压生成对象表面进行自动光滑处理
<Extrude>.mapCoords	Boolean	False	当设置为 On 时, 为挤压生成对象自动指定贴图坐标

**注意** 属性.matIDs、.useShapeIDs 和.smooth 在 3ds max 的 MAXScript 里不能进行存取。

#### 10.4.16 FFDBox: Modifier (自由形式变形长方体修改器)

##### 构造函数

FFDBox ...

##### 属性

属性名称	数据类型	默认值	说明
<FFDBox>.dispLattice <FFDBox>.Lattice	Boolean	True	当设置为 On 时, 将绘制连接控制点的线条以形成栅格
<FFDBox>.dispSource <FFDBox>.Source_Volume		False	当设置为 On 时, 控制点和晶格会以未修改的状态显示
<FFDBox>.deformType	Integer	0	设置变形类型: 0: 只有位于源体积内的顶点会变形。源体积外的顶点不受影响 1: 所有顶点都变形
<FFDBox>.falloff	Float	0.0	指定晶格外节点的变形受晶格内节点变形效果影响的衰减度。可动画
<FFDBox>.tension	Float	25.0	调整控制点作用效果的张力。可动画
<FFDBox>.continuity	Float	25.0	调整控制点作用效果的连贯性。可动画
<FFDBox>.inPoints <FFDBox>.Inside_Points	Boolean	True	当设置为 On 时, 仅控制受与“图形一致”影响的对象内部点
<FFDBox>.outPoints <FFDBox>.Outside_Points	Boolean	True	当设置为 On 时, 仅控制受“与图形一致”影响的对象外部点
<FFDBox>.offset	Float	0.05	受“与图形一致”影响的控制点偏移对象曲面的距离。可动画
<FFDBox>.lattice_transform	SubAnim		
<FFDBox.lattice_transform>.position	Point3	[0,0,0]	可动画
<FFDBox.lattice_transform>.rotation	Quat	(quat 0 0 0 1)	可动画
<FFDBox.lattice_transform>.scale	Point3	[1,1,1]	可动画

## 方法

### 1. conformToShape <FFDBox>

使 FFDDBox Modifier 的控制点和被应用对象的节点一致。如果指定 FFD Modifier 被同时赋给多个对象，调用本函数时什么操作也不会发生。

### 2. resetLattice <FFDBox>

将 FFDDBox Modifier 的控制点恢复为默认位置。

### 3. getDimensions <FFDBox>

#### setDimensions <FFDBox> <Point3>

获取/设置 FFDBox Modifier 的控制点的数目。在 setDimensions 函数的<Point3>参数里，第一个分量指定宽度方向控制点的数目，第二个分量指定长度方向控制点的数目，第三个分量指定高度方向控制点的数目，每个分量的最小值为 2。

### 4. animateVertex <FFDBox> <control\_point\_spec>

为 FFDBox Modifier 的指定控制点设置 Controller，参数<control\_point\_spec>可以为下面几种格式：

<Integer\_index>

<Integer\_index\_array>

#all

控制点赋给 Controller 后，会被添加到 Master subAnim 里，在 Track View 视窗里显示为 animatable，并可以对它做进一步的动画设置。如：

```
animateVertex $Box01.Modifier[1] #all
```

### 5. animateAll <FFDBox>

为 FFDBox Modifier 的所有控制点设置 Controller。

**注意** 用 MAXScript 创建的 FFDBox 控制点默认值为 4x4x4。

## 10.4.17 FFDCyl: Modifier (自由形式变形圆柱体修改器)

### 构造函数

FFDCyl ...

### 属性

属性名称	数据类型	默认值	说明
<FFDCyl>.dispLattice <FFDCyl>.Lattice	Boolean	True	当设置为 On 时，将绘制连接控制点的线条以形成栅格
<FFDCyl>.dispSource <FFDCyl>.Source_Volume	Boolean	False	当设置为 On 时，控制点和晶格会以未修改的状态显示
<FFDCyl>.deformType	Integer	0	设置变形类型： 0：只有位于源体积内的顶点会变形。源体积外的顶点不受影响 1：所有顶点都变形

(续表)

属性名称	数据类型	默认值	说明
<FFDCyl>.falloff	Float	0.0	指定晶格外节点的变形受晶格内节点变形效果影响的衰减度。可动画
<FFDCyl>.tension	Float	25.0	调整控制点作用效果的张力。可动画
<FFDCyl>.continuity	Float	25.0	调整控制点作用效果的连贯性。可动画
<FFDCyl>.inPoints <FFDCyl>.Inside_Points	Boolean	True	当设置为 On 时，仅控制受“与图形一致”影响的对象内部点
<FFDCyl>.outPoints <FFDCyl>.Outside_Points	Boolean	True	当设置为 On 时，仅控制受“与图形一致”影响的对象外部点
<FFDCyl>.offset	Float	0.05	受“与图形一致”影响的控制点偏移对象曲面的距离。可动画
<FFDCyl>.lattice_transform	SubAnim		
<FFDCyl.lattice_transform>.position	Point3	[0,0,0]	可动画
<FFDCyl.lattice_transform>.rotation	Quat	(quat 0 0 0 1)	可动画
<FFDCyl.lattice_transform>.scale	Point3	[1,1,1]	可动画

## 方法

### 1. conformToShape <FFDCyl>

使 FFDCyl Modifier 的控制点和被应用对象的节点一致。如果指定 FFD Modifier 被同时赋给多个对象，调用本函数时什么操作也不会发生。

### 2. resetLattice <FFDCyl>

将 FFDCyl Modifier 的控制点恢复为默认位置。

### 3. getDimensions <FFDCyl>

setDimensions <FFDCyl> <Point3>

获取/设置 FFDCyl Modifier 的控制点的数目。在 setDimensions 函数的<Point3>参数里，第一个分量指定 Side 方向控制点的数目，第二个分量指定 Radial 方向控制点的数目，第三个分量指定 height 方向控制点的数目，每个分量的最小值分别为 6、2、2。

### 4. animateVertex <FFDCyl> <control\_point\_spec>

为 FFDCyl Modifier 的指定控制点设置 Controller，参数<control\_point\_spec>可以为下面几种格式：

```
<Integer_index>
<Integer_index_array>
#all
```

控制点赋给 Controller 后，会被添加到 Master subAnim 里，在 Track View 视窗里显示为 animatable（可动画），并可以对它做进一步的动画设置。如：

```
animateVertex $Box01.Modifier[1] #all
```

### 5. animateAll <FFDCyl>

为 FFDCyl Modifier 的所有控制点设置 Controller。

**注意** 用 MAXScript 创建的 FFDCyl 控制点默认值为 4x6x4。

#### 10.4.18 FFD\_2x2x2: Modifier (自由形式变形)

##### 构造函数

FFD\_2x2x2 ...

FFD2x2x2 ...

##### 属性

属性名称	数据类型	默认值	说明
<FFD_2x2x2>.dispLattice <FFD_2x2x2>.Lattice	Boolean	True	当设置为 On 时，将绘制连接控制点的线条以形成栅格
<FFD_2x2x2>.dispSource <FFD_2x2x2>.Source_Volume	Boolean	False	当设置为 On 时，控制点和晶格会以未修改的状态显示
<FFD_2x2x2>.deformType	Integer	0	设置变形类型： 0: 只有位于源体积内的顶点会变形。源体积外的顶点不受影响 1: 所有顶点都变形
<FFD_2x2x2>.inPoints <FFD_2x2x2>.Inside_Points	Boolean	True	当设置为 On 时，仅控制受“与图形一致”影响的对象内部点
<FFD_2x2x2>.outPoints <FFD_2x2x2>.Outside_Points	Boolean	True	当设置为 On 时，仅控制受“与图形一致”影响的对象外部点
<FFD_2x2x2>.offset	Float	0.05	受“与图形一致”影响的控制点偏移对象曲面的距离。可动画
<FFD_2x2x2>.lattice_transform	SubAnim		
<FFD_2x2x2.lattice_transform>.position	Point3	[0,0,0]	可动画
<FFD_2x2x2.lattice_transform>.rotation	Quat	(quat 0 0 0 1)	可动画
<FFD_2x2x2.lattice_transform>.scale	Point3	[1,1,1]	可动画
<FFD_2x2x2.lattice_transform>.scale	Point3	[1,1,1]	可动画

##### 方法

1. conformToShape <FFD\_2x2x2>

使 FFD\_2x2x2 Modifier 的控制点和被应用对象的节点一致。如果指定 FFD Modifier 被同时赋给多个对象，调用本函数时什么操作也不会发生。

2. resetLattice <FFD\_2x2x2>

将 FFDD\_2x2x2 Modifier 的控制点恢复为默认位置。

3. animateVertex <FFD\_2x2x2> <control\_point\_spec>

为 FFD\_2x2x2 Modifier 的指定控制点设置 Controller，参数<control\_point\_spec>可以为下面几种格式：

```
<Integer_index>
<Integer_index_array>
#all
```

控制点赋给 Controller 后，会被添加到 Master subAnim 里，在 Track View 视窗里显示为 animatable（可动画），并可以对它做进一步的动画设置。如：

```
animateVertex $Box01.Modifier[1] #all
```

4. animateAll <FFD\_2x2x2>

为 FFD\_2x2x2 Modifier 的所有控制点设置 Controller。

#### 10.4.19 FFD\_3x3x3: Modifier（自由形式变形）

##### 构造函数

FFD\_3x3x3 ...

FFD3x3x3 ...

##### 属性

属性名称	数据类型	默认值	说明
<FFD_3x3x3>.dispLattice <FFD_3x3x3>.Lattice	Boolean	True	当设置为 On 时，将绘制连接控制点的线条以形成栅格
<FFD_3x3x3>.dispSource <FFD_3x3x3>.Source_Volume	Boolean	False	当设置为 On 时，控制点和晶格会以未修改的状态显示
<FFD_3x3x3>.deformType	Integer	0	设置变形类型： 0：只有位于源体积内的顶点会变形。源体积外的顶点不受影响 1：所有顶点都变形
<FFD_3x3x3>.inPoints <FFD_3x3x3>.Inside_Points	Boolean	True	当设置为 On 时，仅控制受“与图形一致”影响的对象内部点
<FFD_3x3x3>.outPoints <FFD_3x3x3>.Outside_Points	Boolean	True	当设置为 On 时，仅控制受“与图形一致”影响的对象外部点
<FFD_3x3x3>.offset	Float	0.05	受“与图形一致”影响的控制点偏移对象曲面的距离。可动画
<FFD_3x3x3>.lattice_transform	SubAnim		
<FFD_3x3x3>.lattice_transform>.position	Point3	[0,0,0]	可动画
<FFD_3x3x3>.lattice_transform>.rotation	Quat	(quat 0 0 0 1)	可动画

##### 方法

1. conformToShape <FFD\_3x3x3>

使 FFD\_3x3x3 Modifier 的控制点和被应用对象的节点一致。如果指定 FFD Modifier 被同时赋给多个对象，调用本函数时什么操作也不会发生。

2. resetLattice <FFD\_3x3x3>

将 FFD\_3x3x3 Modifier 的控制点恢复为默认位置。

### 3. animateVertex <FFD\_3x3x3> <control\_point\_spec>

为 FFD\_3x3x3 Modifier 的指定控制点设置 Controller，参数<control\_point\_spec>可以为下面几种格式：

```
<Integer_index>
<Integer_index_array>
#all
```

控制点赋给 Controller 后，会被添加到 Master subAnim 里，在 Track View 视窗里显示为 animatable (可动画)，并可以对它做进一步的动画设置。如：

```
animateVertex $Box01.Modifier[1] #all
```

### 4. animateAll <FFD\_3x3x3>

为 FFD\_3x3x3 Modifier 的所有控制点设置 Controller。

## 10.4.20 FFD\_4x4x4: Modifier (自由形式变形)

### 构造函数

FFD\_4x4x4 ...

FFD4x4x4 ...

### 属性

属性名称	数据类型	默认值	说明
<FFD_4x4x4>.dispLattice <FFD_4x4x4>.Lattice	Boolean	True	当设置为 On 时，将绘制连接控制点的线条以形成栅格
<FFD_4x4x4>.dispSource <FFD_4x4x4>.Source_Volume	Boolean	False	当设置为 On 时，控制点和晶格会以未修改的状态显示
<FFD_4x4x4>.deformType	Integer	0	设置变形类型： 0：只有位于源体积内的顶点会变形。源体积外的顶点不受影响； 1：所有顶点都变形
<FFD_4x4x4>.inPoints <FFD_4x4x4>.Inside_Points	Boolean	True	当设置为 On 时，仅控制受“与图形一致”影响的对象内部点
<FFD_4x4x4>.outPoints <FFD_4x4x4>.Outside_Points	Boolean	True	当设置为 On 时，仅控制受“与图形一致”影响的对象外部点
<FFD_4x4x4>.offset	Float	0.05	受“与图形一致”影响的控制点偏移对象曲面的距离。可动画
<FFD_4x4x4>.lattice_transform	SubAnim		
<FFD_4x4x4.lattice_transform>.position	Point3	[0,0,0]	可动画
<FFD_4x4x4.lattice_transform>.rotation	Quat	(quat 0 0 0 1)	可动画
<FFD_4x4x4.lattice_transform>.scale	Point3	[1,1,1]	可动画

## 方法

1. conformToShape <FFD\_4x4x4>

使 FFD\_4x4x4 Modifier 的控制点和被应用对象的节点一致。如果指定 FFD Modifier 被同时赋给多个对象，调用本函数时什么操作也不会发生。

2. resetLattice <FFD\_4x4x4>

将 FFD\_4x4x4 Modifier 的控制点恢复为默认位置。

3. animateVertex <FFD\_4x4x4> <control\_point\_spec>

为 FFD\_4x4x4 Modifier 的指定控制点设置 Controller，参数<control\_point\_spec>可以为下面几种格式：

<Integer\_index>

<Integer\_index\_array>

#all

控制点赋给 Controller 后，会被添加到 Master subAnim 里，在 Track View 视窗里显示为 animatable，并可以对它做进一步的动画设置。如：

```
animateVertex $Box01.Modifier[1] #all
```

4. animateAll <FFD\_4x4x4>

为 FFD\_4x4x4 Modifier 的所有控制点设置 Controller。

### 10.4.21 FFD\_Select: Modifier（自由形式变形选择修改器）

#### 构造函数

FFD\_select ...

#### 属性

除通用属性外，FFD\_Select 没有额外的属性。

### 10.4.22 Face\_Extrude: Modifier（面挤出修改器）

#### 构造函数

Face\_Extrude ...

#### 属性

属性名称	数据类型	默认值	说明
<Face_Extrude>.amount	Float	0.0	设置挤压出的厚度。可动画
<Face_Extrude>.scale	Float	100.0	设置挤压端面的中心对挤压面进行放缩。可动画
<Face_Extrude>.extrude_center <Face_Extrude>.extrudeFromCenter	Point3	[0,0,0]	设置从中心点的放射方向进行面挤压。可动画

### 10.4.23 Fillet\_Chamfer: Modifier (圆角/切角修改器)

#### 构造函数

Fillet\_Chamfer ...

#### 属性

除通用属性外, Fillet\_Chamfer 没有额外的属性。

### 10.4.24 Flex: Modifier (柔体修改器)

#### 构造函数

Flex ...

#### 属性

属性名称	数据类型	默认值	说明
<Flex>.flex	Float	1.0	设置柔体量和弯曲量。可动画
<Flex>.strength	Float	3.0	设置跟随弹力的整体弹力强度。值为 100 时为刚体。可动画
<Flex>.sway	Float	7.0	为跟随弹力设置对象停止移动的时间。较低值会增加对象停止移动所需的时间。可动画
<Flex>.chase	Boolean	True	开/关跟随弹力
<Flex>.center	Boolean	True	开/关权重
<Flex>.solver	Integer	0	设置解算器类型。处理器类型越高级结果越可靠和精细, 但是计算时间越长。 0: Euler Solver 1: Mid-point solver 2: Runnge Kutta
<Flex>.samples	Float	5	设置帧中按相等时间间隔运行柔软度模拟的次数。采样数越多, 模拟越精确和稳定。在使用中点或Runge-Kutta4 解算器时, 可能不需要与 Euler 一样多的采样数。可动画
<Flex>.stretch	Float	5.0	确定对象边的拉伸长度。可动画
<Flex>.stiffness	Float	0.1	确定对象的刚度。可动画
<Flex>.paintStrength	Float	0.1	设置绘制更改权重值的量。值越高, 更改权重的速度越快。当强度为 0 时, 绘制不更改权重值
<Flex>.paintRadius	Float	36.0	以世界单位数设置笔刷大小
<Flex>.paintFeather	Float	0.7	设置从笔头中心到边缘的衰减度
<Flex>.absolute	Boolean	False	当设置为 On 时, 将绝对的权重数值赋予选定的节点; 当设置为 Off 时, 在节点原先的权重上加上或减去权重
<Flex>.forceNode <Flex>.Force_Nodes	Array-Parameter	#()	为伸缩编辑修改器指定协同作用的动力空间扭曲。数组

(续表)

属性名称	数据类型	默认值	说明
<Flex>.colliderNode	Array-Parameter	#()	为伸缩编辑修改器指定协同作用的碰撞导向对象。数组
<Flex>.referenceFrame <Flex>.Reference_Frame	Integer	0	设置伸缩模拟过程的开始帧
<Flex>.endFrame	Integer	100	设置伸缩模拟过程的结束帧
<Flex>.enableEndFrame	Boolean	False	开/关伸缩模拟过程的结束帧
<Flex>.affectAll	Boolean	False	设置是否忽略当前的次级结构对象选择集，将伸缩变形效果作用于全部的对象
<Flex>.enableAdvanceSprings	Boolean	False	设置是否使高级弹力效果有效
<Flex>.stretchStrength	Float	0.2	控制边缘弹力的强度。可动画
<Flex>.stretchSway	Float	0.2	控制边缘弹力的摆动。可动画
<Flex>.torqueStrength	Float	0.2	控制形态弹力的强度。可动画
<Flex>.torqueSway	Float	0.2	控制形态弹力的摆动。可动画
<Flex>.holdLength	Boolean	False	开/关边缘弹力保持长度
<Flex>.holdLengthPercent	Float	25.0	设置边缘弹力保持长度的最高百分比
<Flex>.addMode	Integer	0	设置加入弹力的方式： 0: 单一边弹力线(在两个选定的顶点之间创建一个边弹力线。如果选中顶点数不是两个，则不创建弹力线) 1: 保持边长度弹力线(任何顶点选择和相邻顶点之间对象的边创建边弹力线) 2: 仅对当前选择保持边长度弹力线(沿所有选定顶点之间对象的边创建边弹力) 3: 保持形弹力线(从选定顶点到保持形半径内的所有其他顶点创建形弹力线) 4: 仅对当前选择保持形弹力线(在保持形半径内的所有选定顶点之间创建形弹力线)
<Flex>.displaySprings	Boolean	False	设置在次级结构层级下是否显示弹力。如果设为True，边缘弹力显示为蓝色线，形态弹力显示为红色线
<Flex>.holdRadius	Float	50.0	指定加入形态弹力的节点区域范围
<Flex>.extraStrength	Array-Parameter	默认值: #(0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2) 控制每个额外弹力的强度。Float 数组	控制每个额外弹力的从开始到停止步进弹力作用的时间。Float 数组
<Flex>.extraSway	Array-Parameter		
<Flex>.lazyEval	Boolean	False	控制是否启用 lazy evaluation，如果设为 True，系统刷新对象的频率降低，但显示精度会降低

(续表)

属性名称	数据类型	默认值	说明
<Flex>.springColors	Array-Parameter	默认值: #([0,0,1], [0.909091,0,0], [0.909091,0,0],[0.818182,0,0], [0.727273,0,0],[0.636364,0,0], [0.545455,0,0],[0.454545,0,0], [0.363636,0,0],[0.272727,0,0], [0.181818,0,0],[0.0909091,0,0], [0,0,0])	为每一个弹力组设置颜色。point3 数组
<Flex>.customSpringDisplay	Array-Parameter	默认值: #(True, True, True, True, True, True, True, True, True, True)	单独为每一个弹力组设置在次级结构层级下是否显示该弹力组。bool 数组
<Flex>.createSpringDepth	Integer	2	指定当创建柔软对象时系统递归计算的次数
<Flex>.createSpringMult	Float	2.0	指定当创建柔软对象时每一次递归计算后伸缩效果的放大倍数

## 方法

### 1. Paint <flex>

相当于按下界面中 Paint 按钮。

### 2. SetReference <flex>

相当于按下界面中 Set Reference 按钮。

### 3. Reset <flex>

相当于按下界面中 Reset 按钮。

### 4. AddForce <flex> <force>

向 Forces 列表中添加指定弹力<force>。

### 5. RemoveForce <flex> <force>

从 Forces 列表中移走指定弹力<force>。参数<force>为要移走的弹力序号。如果 index = -1，当前被选择的弹力会被移走。

### 6. NumberVertices <flex>

返回一个整数，表示指定系统包含的节点数。

### 7. SelectVertices <flex> <sel> <update>

选择指定的节点。参数说明如下：

- ◆ <sel> Bitarray 值，指定需要选择的节点；

- ◆ <update> Boolean 值，指定视窗是否要刷新。

### 8. GetSelectedVertices <flex>

返回一个 BitArray 值，表示当前被选择的节点。

### 9. GetVertexWeight <flex> <index>

返回指定节点的权重。参数说明如下：

- ◆ <index>(Integer) 指定要获取权重的节点序号。

## 10. SetVertexWeight &lt;flex&gt; &lt;Index\_tab&gt; &lt;values\_tab&gt;

为指定节点设置权重。数组<Index\_tab>和<values\_tab>的长度应该相等。参数说明如下：

- ◆ <Index\_tab> Integer 数组，要设置权重的节点序号数组；
- ◆ <values\_tab> Float 数组，权重数组。

## 11. SetEdgeList &lt;flex&gt; &lt;sel&gt; &lt;update&gt;

设置当前被选择的边集合。参数说明如下：

- ◆ <sel> BitArray 值，指定要选择的边；
- ◆ <update> Boolean 值，指定视窗是否要刷新。

## 12. GetEdgeList &lt;flex&gt;

返回一个 BitArray 值，包含当前被选择的边。

## 13. AddSpringFromSelection &lt;flex&gt; &lt;flag&gt; &lt;addDups&gt;

在两个被选择的顶点之间创建一个弹力，该弹力属于指定的弹力组。参数说明如下：

- ◆ <flag>(Integer) 新创建弹力所属弹力组；
- ◆ <addDups>(Boolean) 如果为 True，顶点之间可以创建一个重复的弹力。

## 14. addSpring &lt;flex&gt; &lt;a&gt; &lt;b&gt; &lt;flag&gt; &lt;addDups&gt;

在两个指定的顶点之间创建一个弹力。参数说明如下：

- ◆ <a>(Integer) 弹力起点序号；
- ◆ <b>(Integer) 弹力终点序号；
- ◆ <flag>(Integer) 新创建弹力所属弹力组，取值范围[0 12]；
- ◆ <addDups>(Boolean) 如果为 True，顶点之间可以创建一个重复的弹力组。

## 15. removeAllSprings &lt;flex&gt;

清除系统里所有的弹力。

## 16. addSpringButton &lt;flex&gt;

相当于按下界面里的 Add Spring 按钮。

## 17. RemoveSpringButton &lt;flex&gt;

相当于按下界面里的 Remove Spring 按钮。

## 18. optionsButton &lt;flex&gt;

相当于按下界面里的 Options 按钮。

## 19. createSimpleSoftButton &lt;flex&gt;

相当于按下界面里的 Create Simple Soft Bodies 按钮。

## 20. RemoveSpringByEnd &lt;flex&gt; &lt;a&gt;

清除系统里与指定顶点<a>相连的所有弹力。

## 21. removeSpringByEnds &lt;flex&gt; &lt;a&gt; &lt;b&gt;

清除系统里与指定顶点<a>和<b>相连的所有弹力。

## 22. removeSpringByIndex &lt;flex&gt; &lt;index&gt;

清除界面 LIST 控件 spring 里序号为<index>的弹力。

23. numberSprings <flex>

返回系统里的弹力数。

24. getSpringGroup <flex> <index>

返回指定序号的弹力所属的弹力组号。

25. setSpringGroup <flex> <index> <group>

为指定的弹力设置弹力组。参数说明如下：

- ◆ <index>(Integer) 设置弹力序号；

- ◆ <group>(Integer) 设置弹力组（0~12）。

26. getSpringLength <flex> <index>

返回指定序号弹力的长度。

27. setSpringLength <flex> <index> <length>

为指定序号弹力设置长度。

28. getIndex <flex> <a> <b>

返回与顶点<a>和<b>相连的弹力序号。

### FlexOps 方法

1. flexOps.GetNumberVertices <Flex>

返回被应用 Flex Modifier 对象的顶点数目。

2. flexOps.GetVertexWeight <Flex> <vertex\_index\_Integer>

返回指定顶点的权重。

3. flexOps.SelectVertices <Flex> \

(<vertex\_index\_Integer> | <index\_Integer\_array> | <bitarray>)

选择指定顶点，先前被选择的顶点被清除。

4. flexOps.isEdgeVertex <Flex> <vertex\_index\_Integer>

如果指定顶点为一个边顶点，返回 1，否则返回 0。

5. flexOps.ClearEdgeVertices <Flex> \

(<vertex\_index\_Integer> | <index\_Integer\_array>)

将指定的单个顶点或顶点数组设为非边顶点。

6. flexOps.SetEdgeVertices <Flex> \

(<vertex\_index\_Integer> | <index\_Integer\_array>)

将指定的单个顶点或顶点数组设为边顶点。

7. flexOps.SetVertexWeights <Flex> \

(<vertex\_index\_Integer> | <index\_Integer\_array>)\

(<weight\_Integer> | <weight\_Integer\_array>)

为指定顶点设置权重。如果顶点和权重为数组，其长度必须相等。

### 10.4.25 HSDS\_Modifier: Modifier (HSDS 修改器)

#### 构造函数

HSDS\_Modifier ...

HSDS ...

#### 属性

属性名称	数据类型	默认值	说明
<HSDS_Modifier>.levelOfDetail <HSDS_Modifier>.LOD	Integer	0	显示细分层次的当前级别。当细分一个选中子对象时，它将自动增加。要在不同细节级别上进行编辑，在堆栈中选择此级别。当前级别轮廓变成红色
<HSDS_Modifier>.matId <HSDS_Modifier>.Material_Id	Integer	1	显示指定给当前选中对象的材质 ID 号码。仅在多边形与元素子对象模式中可用。如果选中多个子对象而它们不共享 ID 号码，此字段是空白
<HSDS_Modifier>.ignoreBackface <HSDS_Modifier>.Ignore_Backface	Boolean	False	当设置为 On 时，仅可以选择法线在视口中可见的子对象。否则，选中所有子对象而不考虑其法线方向
<HSDS_Modifier>.useSoftSel <HSDS_Modifier>.Use_Soft_Selection	Boolean	False	在可编辑对象或编辑修改器内影响移动、旋转和缩放功能的操作，如果变形修改器在子对象选择上进行操作，那么也会影响应用到对象上的变形修改器的操作（后者也可以应用到选择修改器）。当设置为 On 时，软件将样条线曲线变形应用到进行变化的选择周围的未选定子对象上
<HSDS_Modifier>.useEdgeDist <HSDS_Modifier>.Use_Edge_Distance	Boolean	False	当设置为 On 时，将软选择限制到指定的面数，该选择在进行选择的区域和软选择的最大范围之间
<HSDS_Modifier>.affectBackface <HSDS_Modifier>.Affect_Backface	Boolean	False	当设置为 On 时，那些法线方向与选定子对象平均法线方向相反的、取消选择的面就会受到软选择的影响。在顶点和边的情况下，这将应用到它们所依附的面的法线上。如果要操纵细对象的面，诸如细长方体，但又不想影响该对象其他侧的面，可以禁用“影响背面”
<HSDS_Modifier>.edgeDistance <HSDS_Modifier>.Edge_Dis	Integer	1	设置影响区域的边距离。空间沿着曲面进行测量，而不是真实空间
<HSDS_Modifier>.falloff	Float	20.0	用以定义影响区域的距离，即用当前单位表示的从中心到球体的边的距离。使用较高的衰减设置以获得更平缓的倾斜，这取决于几何体的比例
<HSDS_Modifier>.pinch	Float	0.0	沿着垂直轴升高或降低曲线的最高点设置区域的相对指向。当该值为负，会产生一个坑而不是一个点；当设置为 0 时，收缩会在该轴上产生平滑的变换
<HSDS_Modifier>.bubble	Float	0.0	沿着垂直轴展开和收缩曲线设置区域的相对饱满。其受收缩的限制，用以设置膨胀的固定开始点。收缩设为 0 并且膨胀设为 1.0 将会产生最为平滑的凸起。负的膨胀值会将曲线的底部移到曲面以下，在区域的地基周围创建了谷地

(续表)

属性名称	数据类型	默认值	说明
<HDSModifier>.vertexType <HDSModifier>.Vertex_Type	Integer	0	在细分过程中，决定如何处理选中顶点。仅在顶点子对象模式中可用。 0: Standard; 1: Conic; 2: Cusp; 3: Corner 确定网格顶点跟随控制栅格顶点移动的距离。标准提供相对移动的最小量，尖锐和折角提供最大量。在细分过程中，折角从圆形一直保持与细分顶点接近
<HDSModifier>.crease	Float	0.0	指定在选中边或边上执行多少折缝
<HDSModifier>.displayLevel <HDSModifier>.Display_Level	Integer	0	返回当前细分级别
<HDSModifier>.onlyCurrentLevel <HDSModifier>.Only_Current_Level	Boolean	False	当设置为 On 时，仅高亮显示当前细节级别的多边形，而不对其平滑。当处理复杂对象时，使用此选项来加速显示
<HDSModifier>.smoothResult <HDSModifier>.Smooth_Result	Boolean	True	当设置为 On 时，对象中的所有面将会进行平滑处理

### 方法

```

subdivide()
deletePolygon()
hide()
showAll()
createNamedSelection <string>name
activateNamedSelection <string>name
addDetail <Float>length <Float>angle
removeDetail <Float>length <Float>angle
setDetailPrecision <Float>p

```

#### 10.4.26 HDSObject: Modifier (HDS 对象修改器)

本类在 3ds max 5 以后已经不可用了，请用 HDSModifier 代替，具体内容参见上一节。

#### 10.4.27 Lathe: Modifier (车削修改器)

##### 构造函数

Lathe ...

## 属性

属性名称	数据类型	默认值	说明
<Lathe>.degrees	Float	360.0	指定对象沿轴车削的角度。单位为度。可动画
<Lathe>.weldCore	Boolean	False	当设置为 On 时，将旋转轴方向的重合节点进行自动焊接，减少造型的几何复杂程度
<Lathe>.flipNormals	Boolean	False	设置是否反转车削对象面法线的方向
<Lathe>.segs <Lathe>.segments	Integer	16	指定在车削开始点与结束点之间的分段精度。可动画
<Lathe>.capStart	Boolean	False	设置是否为车削对象开始端增加顶盖
<Lathe>.capEnd	Boolean	False	设置是否为车削对象结束端增加顶盖
<Lathe>.capType	Integer	0	设置封顶类型： 0: Morph (端面类型适合于变形编辑) 1: Grid (创建网格类型的端面，其渲染效果比较好)
<Lathe>.output	Integer	1	设置车削对象的输出类型： 0: Patch (面片对象) 1: Mesh (网格对象) 2: NURBS (NURBS 对象)
<Lathe>.mapCoords	Boolean	False	当设置为 On 时，为车削对象自动指定贴图坐标
<Lathe>.matIDs	Boolean	True	当设置为 On 时，为车削对象指定 ID 号码，顶盖的材质 ID 号码为 1；底面的材质 ID 号码为 2；侧面的材质 ID 号码为 3
<Lathe>.useShapeIDs	Boolean	False	当设置为 On 时，使用二维图形的材质 ID 号码作为车削对象材质 ID 号码
<Lathe>.smooth	Boolean	True	当设置为 On 时，对车削对象表面进行自动光滑处理
<Lathe>.axis	SubAnim		在次层级结构下可以变换和动画车削对象的轴
<Lathe.axis>.position	Point3	[0,0,0]	设置车削对象的轴位置。可动画
<Lathe.axis>.rotation	Quat	默认值: (quat -0.707107 0 0 0.707107)	
		设置车削对象的轴旋转角度。可动画	
<Lathe.axis>.scale	Point3	[1,1,1]	设置车削对象的轴放缩因子。可动画

## 10.4.28 Lattice: Modifier (晶格修改器)

### 构造函数

Lattice ...

### 属性

属性名称	数据类型	默认值	说明
<Lattice>.Strut_Radius	Float	2.0	设置连接杆的粗细。可动画
<Lattice>.Strut_Segments	Integer	1	设置连接杆长度方向的分段精度。可动画
<Lattice>.Strut_Sides	Integer	4	设置连接杆周长方向的边数。可动画
<Lattice>.Joint_Radius	Float	5.0	设置连接点的大小。可动画
<Lattice>.Joint_Segs	Integer	1	设置连接点造型的分段精度，精度越高越接近球体。可动画

**注意** Geometry、Struts Material ID、Struts Ignore Hidden Edges、Struts End Caps、Struts Smooth、Joints Base Type、Joints Material ID、Joints Smooth 以及 Mapping Coordinates 参数都不被 MAXScript 支持。

#### 10.4.29 Linked\_XForm: Modifier (链接变换修改器)

##### 构造函数

Linked\_XForm ...

##### 属性

属性名称	数据类型	默认值	说明
<Linked_XForm>.Control	Node	undefined	与顶点连接的对象，当它被变形时，顶点会跟着变形
<Linked_XForm>.BackTransform <Linked_XForm>.Back_Transform	Boolean	True	当设置为 On 时，将使用链接变换修改器的对象与控制对象链接起来。通常情况下，移动控制对象将使链接对象移动两次，一次伴随控制对象，一次伴随链接

#### 10.4.30 LS\_Mesh: Modifier (LS 网格修改器)

##### 构造函数

LS\_Mesh ...

LSMesh ...

##### 属性

属性名称	数据类型	默认值	说明
<LS_Mesh>.depth	Integer	0	设置最大细化深度
<LS_Mesh>.limitDepth <LS_Mesh>.Limit_Depth	Boolean	True	当设置为 On 时，允许设置最大细化深度。否则，网格修改器将会下降到细化的底部
<LS_Mesh>.limitSize <LS_Mesh>.Limit_Size	Boolean	False	当设置为 On 时，允许限制细化多边形的尺寸。否则，网格修改器将细化任意尺寸的多边形
<LS_Mesh>.size	Float	19.685	设置限制细化多边形的尺寸。此尺寸是以当前视图单位表示的长度。如果多边形小于此尺寸的平方，修改器将不会对其进行细化

### 10.4.31 MaterialByElement: Modifier（按元素分配材质修改器）

#### 构造函数

MaterialByElement ...

#### 属性

属性名称	数据类型	默认值	说明
<MaterialByElement>.method	Integer	1	设置元素的材质 ID 号码的分配方式： 0: Random Distribution（随机分配子材质 ID 号码） 1: List Frequency（为子材质指定不同的使用权重）
<MaterialByElement>.Material_ID_Count	Integer	2	指定最少使用的材质 ID 号码数量。可动画
<MaterialByElement>.Material_ID_1	Float	0.5	设置第一个材质 ID 的权重。百分比，可动画
<MaterialByElement>.Material_ID_2	Float	0.5	设置第二个材质 ID 的权重。百分比，可动画
<MaterialByElement>.Material_ID_3	Float	0.0	设置第三个材质 ID 的权重。百分比，可动画
<MaterialByElement>.Material_ID_4	Float	0.0	设置第四个材质 ID 的权重。百分比，可动画
<MaterialByElement>.Material_ID_5	Float	0.0	设置第五个材质 ID 的权重。百分比，可动画
<MaterialByElement>.Material_ID_6	Float	0.0	设置第六个材质 ID 的权重。百分比，可动画
<MaterialByElement>.Material_ID_7	Float	0.0	设置第七个材质 ID 的权重。百分比，可动画
<MaterialByElement>.Material_ID_8	Float	0.0	设置第八个材质 ID 的权重。百分比，可动画
<MaterialByElement>.Random_Seed	Integer	12345	为随机的材质 ID 号码指定过程设置种子数

**注意** 属性.Material\_ID\_X 的值在 3ds max 界面上显示为百分比，而实际存储值为小数。

### 10.4.32 MaterialModifier: Modifier（材质修改器）

#### 构造函数

MaterialModifier ...

#### 属性

属性名称	数据类型	默认值	说明
<MaterialModifier>.Materialid <MaterialModifier>.Material_ID	Integer	1	指定材质 ID 号码。如果输入对象为 Face 级子对象选集，指定材质 ID 号码仅应用在被选择的 Face 上，否则应用在整个对象上。可动画

**注意**

由 MaterialModifier Modifier 创建的 Modifier 的默认名为 Material, 如果马上将该 Modifier 作为对象的一个属性来进行存取, 会与对象的材质属性.Material 发生冲突, 为了避免这一点, 建议用户在创建 MaterialModifier Modifier 时为它指定一个名字, 如:

```
addModifier myObj(MaterialModifier MaterialID:5 name: "MaterialMod")
```

## 10.4.33 Melt: Modifier (融化修改器)

## 构造函数

Melt ...

## 属性

属性名称	数据类型	默认值	说明
<Melt>.Melt_Amount	Float	0.0	指定融化的衰减范围。可动画
<Melt>.Spread	Float	19.0	指定对象融化后延展的百分率。可动画
<Melt>.Solidity_Preset	Integer	0	指定融化对象中心的相对高度: 0: Ice (默认固态设置) 1: Glass (使用高固态设置来模拟玻璃) 2: Jelly (产生在中心处显著的下垂效果) 3: Plastic (相对的固体,但是在融化时其中心稍微下垂) 4: Custom: (将固态设置为 0.2~30.0 间的任何值)
<Melt>.Solidity_Custom_Value	Float	1.0	将固态设置为 0.2~30.0 间的任何值。可动画
<Melt>.axis	Integer	0	选择会产生融化的轴: 0: Z; 1: Y; 2: X
<Melt>.Negative_Axis	Integer	0	设置是否反转融化作用的轴向
<Melt>.center	Point3	[0,0,0]	设置融化编辑器 gizmo 的中心
<Melt>.gizmo	SubAnim		在次级结构层级, 可以移动、旋转、放缩 gizmo, 还可以为 gizmo 指定变换动画
<Melt.gizmo>.position	Point3	[0,0,0]	设置 gizmo 的位置。可动画
<Melt.gizmo>.rotation	Quat	默认值: (quat 0 0 0 1)	
			设置 gizmo 的旋转角度。旋转中心为 gizmo 的中心。可动画
<Melt.gizmo>.scale	Point3	[1,1,1]	设置 gizmo 的放缩因子。可动画
接下来的属性被指定给融化编辑器, 但不能被使用:			
<Melt>.cut_OffObsolete	Float	0.0	可动画
<Melt>.Confine_To_Gizmo	Integer	0	

## 10.4.34 MeshSmooth: Modifier (网格平滑修改器)

## 构造函数

MeshSmooth ...

### 属性

属性名称	数据类型	默认值	说明
<MeshSmooth>.subdivMethod	Integer	2	设置细分方式： 0: Classic (通过创建三角形或四边形面光滑对象) 1: Quad Output (仅创建四边形的面) 2: NURMS (减少非均匀有理数网格平滑对象)
<MeshSmooth>.Apply_To_Whole_Mesh	Boolean	True	当设置为 On 时，则对全部网格表面指定光滑操作；反之，可以对次级结构对象选择集指定网格光滑操作
<MeshSmooth>.ignoreSel	Boolean	True	当设置为 On 时，忽略次级结构对象选择集，对整个对象进行网格光滑操作
<MeshSmooth>.oldMapping	Boolean	False	设置是否使用老版本的计算方法指定贴图坐标
<MeshSmooth>.iterations	Integer	0	指定使用已有节点插值创建新面的数量。可动画
<MeshSmooth>.smoothness <MeshSmooth>.smoothness_filter	Float	1.0	指定需要进行光滑处理的拐角的尖锐度。如果为 0，则不会增加新面进行光滑处理；如果为 1，将对所有节点加入新面进行光滑处理。可动画
<MeshSmooth>.UseRenderIterations <MeshSmooth>.use_render_iterations	Boolean	False	设置是否为渲染输出指定一个不同的插值
<MeshSmooth>.renderIterations <MeshSmooth>.render_iterations	Integer	0	设置渲染输出的插值。可动画
<MeshSmooth>.useRenderSmoothness <MeshSmooth>.use_render_smoothness	Boolean	False	设置是否为渲染输出指定一个不同的光滑度
<MeshSmooth>.renderSmoothness <MeshSmooth>.render_smoothness	Float	1.0	设置渲染输出的光滑度。可动画
<MeshSmooth>.ignoreBackfacing	Boolean	False	当设置为 On 时，在选择次级结构对象时忽略背面的对象
<MeshSmooth>.controlLevel	Integer	0	可以观看控制网格，还可以编辑次级结构点或边
<MeshSmooth>.displayCage <MeshSmooth>.display_control_mesh	Boolean	False	当设置为 On 时，显示一个框架对象，该对象形象地显示了光滑效果
<MeshSmooth>.useSoftSel	Boolean	False	设置是否使软选择功能有效
<MeshSmooth>.useEdgeDist	Boolean	False	开/关边距离设置
<MeshSmooth>.edgeDist	Integer	1	设置软选择功能在网格对象表面的作用范围
<MeshSmooth>.affectBackfacing	Boolean	False	当设置为 On 时，指定当前的软选择功能操作同时作用于对象的背面
<MeshSmooth>.falloff	Float	20.0	指定作用范围半径的大小

(续表)

属性名称	数据类型	默认值	说明
<MeshSmooth>.pinch	Float	0.0	指定曲线在箭头尖顶点汇集的形状，负值表现平坦的顶部，正值表现为尖锐的顶部
<MeshSmooth>.bubble	Float	0.0	指定隆起曲线的曲率，指定为 1.0 时产生一个半球体隆起，降低该数值隆起曲线会变得陡峭；负值可以产生下陷的效果
<MeshSmooth>.strength	Float	0.5	设置加入面的尺寸，取值范围是 0.0~1.0。如果趋向于 0.0 则会创建非常小的面；如果趋向于 1.0 则会创建非常大的面。可动画
<MeshSmooth>.Relax	Float	0.0	指定一个正值可以光滑所有的节点。可动画
<MeshSmooth>.limitSurface <MeshSmooth>.project_to_limit_surface	Boolean	False	当设置为 On 时，将光滑结束的所有点放置到限定的面上
<MeshSmooth>.smoothResult <MeshSmooth>.smooth_output	Boolean	False	当设置为 On 时，为所有面指定相同的光滑组
<MeshSmooth>.sepByMats <MeshSmooth>.separate_by_Materials	Boolean	False	当设置为 On 时，利用对象表面不同区域的材质 ID 号码，限定不同的光滑区域
<MeshSmooth>.sepBySmGroups <MeshSmooth>.separate_by_smoothing_group	Boolean	False	当设置为 On 时，利用对象表面不同区域的光滑组，限定不同的光滑区域
<MeshSmooth>.faceType	Integer	1	设置输入转换方式： 0: Faces (将所有三角形的面都作用为一个面，光滑操作将作用于所有边（包括不可见边）) 1: Polygons (只将多边形作为一个面，光滑操作忽略不可见的边)
<MeshSmooth>.keepConvex <MeshSmooth>.keep_input_faces_convex	Boolean	False	设置是否保持所有输入的多边形都是外凸的，所有非外凸的多边形都被作为是由几个面拼接而成的
<MeshSmooth>.update <MeshSmooth>.update_options	Integer	0	设置更新的方式： 0: 自动更新；1: 渲染时更新；2: 手工更新
<MeshSmooth>.reset	Integer	0	设置重置方式： 0: 重置所有级别；1: 重置当前级别

**注意** 边和顶点的权重属性在 3ds max 的 MAXScript 里不能进行存取。

#### 10.4.35 Mesh\_Select: Modifier (网格选择修改器)

##### 构造函数

Mesh\_Select ...

MeshSelect ...

### 属性

属性名称	数据类型	默认值	说明
<Mesh_Select>.Use_Soft_Selection	Integer	0	设置是否使软选择功能有效。可动画 0: Off; 1: On
<Mesh_Select>.falloff	Float	20.0	指定作用范围半径的大小。可动画
<Mesh_Select>.Pinch	Float	0.0	指定曲线在箭头尖顶点汇集的形状，负指值表现平坦的顶部，正值表现为尖锐的顶部。可动画
<Mesh_Select>.Bubble	Float	0.0	指定隆起曲线的曲率，指定为 1.0 时产生一个半球体隆起，降低该数值隆起曲线会变得陡峭；负值可以产生下陷的效果。可动画

有关存取 Mesh\_Select Modifier 的当前选择集的内容，请读者参见第 9 章中的 getVertSelection()、getFaceSelection() 和 getEdgeSelection() 函数。

### 10.4.36 Mirror: Modifier (镜像修改器)

#### 构造函数

Mirror ...

#### 属性

属性名称	数据类型	默认值	说明
<Mirror>.mirror_axis	Integer	0	指定镜像操作所依据的轴向：0: X; 1: Y; 2: Z; 3: XY; 4: YZ; 5: ZX
<Mirror>.Mirror_Offset	Float	0.0	指定对象镜像操作后的偏移距离。可动画
<Mirror>.copy	Boolean	False	当设置为 On 时，镜像并复制对象
<Mirror>.mirror_center	SubAnim		显示镜像效果的轴向，可以移动、旋转、放缩 gizmo
<Mirror.Mirror_Center>.position	Point3	[0,0,0]	设置镜像 gizmo 的位置。可动画
<Mirror.Mirror_Center>.rotation	Quat	默认值: (quat 0 0 0 1)	
			设置镜像 gizmo 的旋转角度。可动画
<Mirror.Mirror_Center>.scale	Point3	[1,1,1]	设置镜像 gizmo 的放缩因子。可动画

### 10.4.37 Morpher: Modifier (变形器修改器)

#### 构造函数

Morpher ...

#### 属性

属性名称	数据类型	默认值	说明
<Morpher>.Use_Limits	Integer	1	当设置为 On 时，所有通道使用最大和最小限定
<Morpher>.Spinner_Minimum	Float	0.0	设置通道最小限定
<Morpher>.Spinner_Maximum	Float	100.0	设置通道最大限定
<Morpher>.Use_Selection	Integer	0	打开将限定设置应用于节点的选择集。节点选择集是修改编辑堆栈中由变形修改编辑器下面的修改编辑器向上传递的节点选择集
<Morpher>.Value_Increments	Integer	1	设置 Spinner_Minimum 控件和 Spinner_Maximum 控件的增量
<Morpher>.Autoload_of_targets	Integer	0	当设置为 On 时，指定可以重新导入变形目标对象

## 方法

1. IsValidMorpherMod <Modifier\_class>Modifier <boolean>

当指定的 Modifier 是一个有效的 Morpher Modifier 时，返回 True；否则返回 False。

2. WM3\_MC\_BuildFromNode <Morpher\_Class>Morpher <boolean>

channel\_index <geometry\_class>target\_object <Integer>

添加一个目标对象到指定 Morpher Modifier 的指定通道。如果成功了，返回 True。

例如：

```
WM3_MC_BuildFromNode $Teapot01.morpher 1 $Teapot02
True
```

3. WM3\_MC\_Delete <Morpher\_Class>Morpher <Integer>channel\_index <boolean>

从指定 Morpher Modifier 的指定通道删除一个目标物体。如果成功了，返回 True。

例如：

```
WM3_MC_Delete $Teapot01.morpher 1
True
```

4. WM3\_MC\_GetLimitMAX <Morpher\_Class>\

Morpher <Integer>channel\_index <Float>

返回指定 Morpher Modifier 的指定通道的最大限定。

例如：

```
WM3_MC_GetLimitMAX $Teapot01.morpher 1
100.0
```

5. WM3\_MC\_GetLimitMIN <Morpher\_Class>\

Morpher <Integer>channel\_index <Float>

返回指定 Morpher Modifier 的指定通道的最小限定。

例如：

```
WM3_MC_GetLimitMIN $Teapot01.morpher 1
0.0
```

6. WM3\_MC\_GetMemUse <Morpher\_Class>Morpher <Integer>channel\_index <Float>  
返回指定 Morpher Modifier 的指定通道的内存使用字节数。

例如：

```
WM3_MC_GetMemUse $Teapot01.morpher 1
10600.0
```

7. WM3\_MC\_GetMorphPoint <Morpher\_Class>Morpher \
<Integer> channel\_index <Integer>index <Point3>

返回指定 Morpher Modifier 的指定通道的索引变形点的位置。

8. WM3\_MC\_GetMorphWeight <Morpher\_Class> \
Morpher <Integer> channel\_index <Integer>index <Float>

返回指定 Morpher Modifier 的指定通道的索引变形点的权重值。

9. WM3\_MC\_GetName <Morpher\_Class>Morpher <Integer>channel\_index <string>  
返回指定 Morpher Modifier 的指定通道名字。

10. WM3\_MC\_GetTarget <Morpher\_Class>Morpher <Integer>channel\_index <node>  
返回指定 Morpher Modifier 的指定通道目标对象。

11. WM3\_MC\_GetUseLimits <Morpher\_Class> \
Morpher<Integer> channel\_index <boolean>  
如果指定 Morpher Modifier 的指定通道使用最大最小限定，返回 True；否则返回 False。

12. WM3\_MC\_GetUseVertexSel <Morpher\_Class> \
Morpher <Integer>channel\_index <boolean>  
如果指定 Morpher Modifier 的指定通道使用顶点选择，返回 True；否则返回 False。

13. WM3\_MC\_GetValue <Morpher\_Class>Morpher <Integer>channel\_index <Float>  
返回指定 Morpher Modifier 的指定通道数值。

14. WM3\_MC\_HasData <Morpher\_Class>Morpher <Integer>channel\_index <boolean>  
如果指定 Morpher Modifier 的指定通道指定了目标对象，返回 True；否则返回 False。

15. WM3\_MC\_HasTarget <Morpher\_Class> \
Morpher <Integer>channel\_index <boolean>  
如果指定 Morpher Modifier 的指定通道指定了目标对象，返回 True；否则返回 False。

16. WM3\_MC\_IsActive <Morpher\_Class> \
Morpher <Integer>channel\_index <boolean>  
如果指定 Morpher Modifier 的指定通道是活动的，返回 True；否则返回 False。

17. WM3\_MC\_IsValid <Morpher\_Class>Morpher <Integer>channel\_index <boolean>  
如果指定 Morpher Modifier 的指定通道是有效的，返回 True；否则返回 False。

18. WM3\_MC\_NumMPts <Morpher\_Class>Morpher <Integer>channel\_index <Integer>  
返回指定 Morpher Modifier 的指定通道的索引变形点的数量。

19. WM3\_MC\_NumPts <Morpher\_Class>Morpher <Integer>channel\_index <Integer>

返回指定 Morpher Modifier 的指定通道的索引变形点的总数量。

20. WM3\_MC\_Rebuild <Morpher\_Class>Morpher <Integer>channel\_index <boolean>  
重新加载指定 Morpher Modifier 的指定通道的目标对象。如果成功了，返回 True。

21. WM3\_MC\_SetActive <Morpher\_Class>Morpher \  
<Integer>channel\_index <boolean>active\_state<boolean>

使指定 Morpher Modifier 的指定通道的有效或无效。如果成功了，返回 True。

22. WM3\_MC\_SetLimitMAX <Morpher\_Class>Morpher \  
<Integer>channel\_index <Float>max\_limit<boolean>

指定 Morpher Modifier 的指定通道的最大限定。如果成功了，返回 True。

23. WM3\_MC\_SetLimitMIN <Morpher\_Class>Morpher \  
<Integer>channel\_index <Float>min\_limit <boolean>

指定 Morpher Modifier 的指定通道的最小限定。如果成功了，返回 True。

24. WM3\_MC\_SetName <Morpher\_Class>Morpher \  
<Integer>channel\_index <string>new\_name <boolean>

指定 Morpher Modifier 的指定通道的名称。如果成功了，返回 True。

25. WM3\_MC\_SetUseLimits <Morpher\_Class>Morpher \  
<Integer>channel\_index <boolean>use\_limits <boolean>

设置指定 Morpher Modifier 的指定通道使用最大最小限定，如果成功了，返回 True。

26. WM3\_MC\_SetUseVertexSel <Morpher\_Class>Morpher \  
<Integer>channel\_index <boolean>use\_vert\_sel <boolean>

设置指定 Morpher Modifier 的指定通道使用顶点选择，如果成功了，返回 True。

27. WM3\_MC\_SetValue <Morpher\_Class>Morpher \  
<Integer>channel\_index <Float>new\_value <boolean>

设置指定 Morpher Modifier 的指定通道的数值，如果成功了，返回 True。

28. WM3\_RebuildInternalCache <Morpher\_Class>Morpher <boolean>

重新加载指定 Morpher Modifier 的内部缓存器，如果成功了，返回 True。

29. WM3\_RefreshChannelListUI <Morpher\_Class>Morpher <boolean>

更新指定 Morpher Modifier 的通道列表，如果成功了，返回 True。

30. WM3\_RefreshChannelParamsUI <Morpher\_Class>Morpher <boolean>

更新指定 Morpher Modifier 的通道参数，如果成功了，返回 True。

31. WM3\_SetChannelPos <Morpher\_Class>Morpher <Integer>new\_pos <boolean>

滚动 Morpher 用户界面到指定的通道，如果成功了，返回 True。

32. WM3\_SetChannelSel <Morpher\_Class>Morpher <Integer>new\_sel <boolean>

选择指定 Morpher Modifier 的指定通道，如果成功了，返回 True。

#### 10.4.38 MultiRes: Modifier (多分辨率修改器)

##### 构造函数

MultiRes ...

MrmMod ...

**属性**

属性名称	数据类型	默认值	说明
<MultiRes>.vertexCount <MultiRes>.Vertex_Count	Integer	0	修改对象中的顶点总数以设置输出网格中的最大顶点数
<MultiRes>.vertexPercent <MultiRes>.Vertex_Percentage	Float	100.0	修改对象的顶点数相对于原始网格中顶点总数的百分比。注意：在输入特定百分比后，例如 30，可能会发现该软件将该值更改为稍小于输入值，例如 29.971。这是由于模型中顶点总数和百分比计算间的关系。这并不是错误，只是对读者要求的最好近似
<MultiRes>.mergeVertex <MultiRes>.Vertex_Merging	Boolean	False	当设置为 On 时，多分辨率修改器可以在模型中的分散元素间合并顶点。例如，如果将多分辨率修改器应用到由四个独立的元素组成的茶壶上，并启用顶点合并，那么当调整顶点分辨率时，这些独立的组件会一起合并到邻近的低分辨率对象上。要控制顶点合并，可以设置合并阈值，该值决定顶点在什么样的单位距离内会以较高比率合并
<MultiRes>.mergeThreshold <MultiRes>.Threshold	Float	0.0	以 3ds max 单位设置顺序顶点间的最大距离，顶点间距小于这一距离的将考虑进行合并。降低网格复杂度时，在这一距离内的元素间顶点以较高比率焊接在一起。只有在启用顶点合并时才有效。注意：要只减少重叠的顶点，那么将阈值设置为 0.0
<MultiRes>.mergeWithinMesh <MultiRes>.Merge_Within	Boolean	False	当设置为 On 时，多分辨率修改器合并相邻元素的边界以及元素内部的顶点。许多对象可以含有不共享连接的多个顶点组。一个简单的例子是茶壶对象。它由四个不同的元素组成：壶身、壶柄、壶嘴和壶盖。通常，多分辨率修改器在其自身的网格中优化每个分散的元素
<MultiRes>.BoundaryMetric <MultiRes>.Boundary_Metric	Boolean	False	当设置为 On 时，多分辨率修改器保留指定给选中模型的材质。由材质 ID 定义的材质边界尽可能长的保留，并且在降低顶点数时是最后消除的部分
<MultiRes>.baseVertices <MultiRes>.Maintain_Base_Vertices	Boolean	False	当设置为 On 时，覆盖多分辨率优化算法並將在多分辨率修改器顶点子对象层级选择的任何顶点作为关键顶点保留。使用该功能以保留对象或角色的关键特征点，例如其手指或爪，或者使用该功能以保留其他几何体，如果简化的太多，这些几何体就可能变得无法识别
<MultiRes>.multiVertNorm <MultiRes>.Multiple_Normals_Per_Vertex	Boolean	True	当设置为 On 时，让多分辨率修改器为每个顶点指定多条法线。默认情况下，多分辨率修改器为每个顶点生成一条法线。启用多顶点法线后，当顶点附近的几何体更改时，多分辨率修改器生成法线更新。必须以度为单位（范围为 0.0~180.0）指定折缝角度。折缝角度是在面法线间的角度。该值用于决定法线应当何时在两个面间跨越边缘共享。例如，在定义为 10 x 10 的面的网格栅格的平面上，两个相邻面的折缝角度为 0。在立方体中，相邻面的折缝角度为 90 度。通常，接近 0 的折缝角度会产生更为平滑的着色。接近 180 度的折缝角度会产生更明显的角

(续表)

属性名称	数据类型	默认值	说明
<MultiRes>.creaseAngle <MultiRes>.Crease_Angle	Float	75.0	用于生成多条法线所需要的折痕值。只有在启用每个顶点有多个法线时才有效。最佳折缝角度取决于模型；交互式的设置该值、检查视口并为着色效果渲染图像。虽然使用多项点法线可以更精确地着色，但是同时这也需要更多的内部数据
<MultiRes>.reqGenerate <MultiRes>.Generate	Boolean	False	将所有设置都设置为与上一次使用生成时的设置值相同。只有在更改了这些设置中的一个或多个时该选项才有效。使用重置来检查生成参数是否与上一次生成网格时的设置相同
<MultiRes>.resetParams <MultiRes>.Reset	Boolean	False	

例如：

```
modPanel.addModToSelection(MultiRes())
$.modifiers[#MultiRes].vertexCount = 482
$.modifiers[#MultiRes].vertexPercent = 100
$.modifiers[#MultiRes].vertexPercent = 99
$.modifiers[#MultiRes].vertexCount = 476
$.modifiers[#MultiRes].vertexPercent = 98.7552
$.modifiers[#MultiRes].mergeVertex = On
$.modifiers[#MultiRes].mergeThreshold = 0.01
$.modifiers[#MultiRes].mergeWithinMesh = On
$.modifiers[#MultiRes].boundaryMetric = On
$.modifiers[#MultiRes].baseVertices = On
$.modifiers[#MultiRes].creaseAngle = 75.75
```

#### 10.4.39 NCurve\_Sel: Modifier (NURBS 曲线选择修改器)

##### 构造函数

NCurve\_Sel ...

##### 属性

除通用属性之外， NCurve\_Sel 没有额外的属性。

#### 10.4.40 NoiseModifier: Modifier (噪波修改器)

##### 构造函数

NoiseModifier ...

##### 属性

属性名称	数据类型	默认值	说明
<NoiseModifier>.seed	Integer	0	指定噪波修改编辑器的随机根值。可动画
<NoiseModifier>.scale	Float	100.0	设定噪波效果的尺寸（不是长度），高的数值产生平滑的噪波，低的数值产生锯齿状的噪波。可动画
<NoiseModifier>.fractal	Boolean	False	设置是否对噪波进行分型处理。可动画
<NoiseModifier>.roughness <NoiseModifier>.rough	Float	0.0	指定分型面的大小，高的数值比低的数值产生的分形效果更为粗糙。可动画
<NoiseModifier>.iterations	Float	6.0	指定重复进行分型处理的次数，较低的数值可以产生更为平滑的效果，设置为1时等同于关闭分型处理。可动画
<NoiseModifier>.strength	Point3	[0,0,0]	指定三个轴向上的噪波作用强度。可动画
<NoiseModifier>.animate	Boolean	False	开/关动画噪波的设定
<NoiseModifier>.frequency	Float	0.25	指定噪波变化的频率，较高的数值产生一个快速变化的噪波；较低的数值产生一个缓慢变化的噪波
<NoiseModifier>.phase	Time	0f	指定噪波动画的初始相位。可动画
<NoiseModifier>.center	Point3	[0,0,0]	在次级结构层级，可以移动、旋转、放缩噪波编辑器gizmo的中心对象，还可以为噪波编辑器gizmo的中心对象指定变换动画。可动画
<NoiseModifier>.gizmo	SubAnim		在次级结构层级，可以移动、旋转、放缩噪波编辑器gizmo，还可以为gizmo指定变换动画。
<NoiseModifier>.gizmo>.position	Point3	[0,0,0]	设置噪波编辑器gizmo的位置。可动画
<NoiseModifier>.gizmo>.rotation	Quat	默认值：(quat 0 0 0 1)	设置噪波编辑器gizmo的旋转角度。可动画
<NoiseModifier>.gizmo>.scale	Point3		
		[1,1,1]	设置噪波编辑器gizmo的放缩因子。可动画

#### 10.4.41 Normalize\_Spl: Modifier（规格化样条线修改器）

##### 构造函数

Normalize\_Spline ...

Normalize\_Spl ...

##### 属性

属性名称	数据类型	默认值	说明
<Normalize_Spl>.Length	Float	20.0	设置大量的控制点的添加方式，较大的值产生较少的控制点，较小的值产生较多的控制点

### 10.4.42 NormalModifier: Modifier (法线修改器)

#### 构造函数

NormalModifier ...

#### 属性

属性名称	数据类型	默认值	说明
<NormalModifier>.unif	Boolean	False	当设置为 On 时，会反转部分法线，以使它们朝向同样的方向，通常朝外
<NormalModifier>.flip	Boolean	False	当设置为 On 时，会将选择的对象或对象集的表面法线全部反转

### 10.4.43 NSurf\_Sel: Modifier (NURBS 曲面选择修改器)

#### 构造函数

NSurf\_Sel ...

#### 属性

除通用属性外，NSurf\_Sel 没有额外的属性。

### 10.4.44 Optimize: Modifier (优化修改器)

#### 构造函数

Optimize ...

#### 属性

属性名称	数据类型	默认值	说明
<optimize>.renderLOD	Integer	0	设置渲染细节级别：0: L1; 1: L2
<optimize>.viewLOD	Integer	0	设置当前视图显示的细节级别：0: L1; 1: L2
<optimize>.facethreshold1 <optimize>.Face_Threshold_L1	Float	4.0	设置面法线之间的角度阈值。面阈值越小优化效果越小，面阈值越大优化效果越粗糙。可动画
<optimize>.edgethreshold1 <optimize>.Edge_Threshold_L1	Float	1.0	设置开放边的角度阈值。较低的边保护开放边。可动画
<optimize>.bias1 <optimize>.Bias_L1	Float	0.1	用于在优化过程中排除多余的小三角面，较高的数值保持三角面，取值范围在 0~1.0 之间。可动画
<optimize>.Max_Edge_Length_1	Float	0.0	指定在优化过程中的最大边长，比最大边长的边在优化之后不再被延伸。可动画
<optimize>.preservemat1	Boolean	False	当设置为 On 时，保护所有的材质边界不被优化

(续表)

属性名称	数据类型	默认值	说明
<optimize>.preservesmooth1	Boolean	False	当设置为 On 时，保护所有的光滑组的边界不被优化，优化只在光滑组之间进行
<optimize>.facethreshold2 <optimize>.Face_Threshold_L2	Float	4.0	设置面法线之间的角度阈值。面阈值越小优化效果越小，面阈值越大优化效果越粗糙。可动画
<optimize>.edgethreshold2 <optimize>.Edge_Threshold_L2	Float	1.0	设置开放边的角度阈值。较低的边保护开放边。可动画
<optimize>.bias2 <optimize>.Bias_L2	Float	0.1	用于在优化过程中排除多余的小三角面，较高的数值保持三角面，取值范围在 0~1.0 之间。可动画
<optimize>.Max_Edge_Length_2	Float	0.0	指定在优化过程中的最大边长，比最大边长的边在优化之后不再被延伸。可动画
<optimize>.preservemat2	Boolean	False	当设置为 On 时，保护所有的材质边界不被优化
<optimize>.preservesmooth2	Boolean	False	当设置为 On 时，保护所有的光滑组的边界不被优化，优化只在光滑组之间进行
<optimize>.manualUpdate	Boolean	False	当设置为 On 时，指定单击 Update 按钮后进行手动更新
<optimize>.autoedge	Boolean	False	当设置为 On 时，指定优化过程中关闭所有法线在面阈值范围之内的边

#### 10.4.45 PatchDeform: Modifier（面片变形修改器）

##### 构造函数

PatchDeform...

##### 属性

属性名称	数据类型	默认值	说明
<PatchDeform>.U_Percent	Float	50.0	指定对象沿面片变形 gizmo 的 U 轴方向移动的百分比。移动的距离取决于变形 gizmo 在 U 轴方向的长度。如果设置为 50，将对象放置在面片 U 轴的中心位置；如果设置为 0，将面片对象放置在面片 U 轴方向的左边缘。可动画，百分比
<PatchDeform>.U_Stretch	Float	1.0	沿面片变形 gizmo 的 U 轴方向放缩对象。可动画
<PatchDeform>.V_Percent	Float	50.0	指定对象沿面片变形 gizmo 的 V 轴方向移动的百分比。移动的距离取决于变形 gizmo 在 V 轴方向的长度。如果设置为 50，将对象放置在面片 V 轴的中心位置；如果设置为 0，将面片对象放置在面片 V 轴方向的底部。可动画，百分比
<PatchDeform>.V_Stretch	Float	1.0	沿面片变形 gizmo 的 V 轴方向放缩对象。可动画
<PatchDeform>.rotation	Float	0.0	旋转变形对象，使其与基准面片呈一个角度。可动画，Angle
<PatchDeform>.Plane_to_Patch_Deform	Integer	0	指定对象进行面片变形所依据变形 gizmo 的一个平面：0: XY; 1: YZ; 2: ZX

(续表)

属性名称	数据类型	默认值	说明
<PatchDeform>.Flip_deformation_axis	Integer	0	设置是否反转面片变形 gizmo 的方向。 0: 不反转; 1: 反转
<PatchDeform>.gizmo	SubAnim		在次级结构层级, 可以移动、旋转、放缩面片变形修改编辑器 gizmo, 还可以为 gizmo 指定变换动画
<PatchDeform.gizmo>.position	Point3	[0,0,0]	设置面片变形修改编辑器编辑器 gizmo 的位置。可动画
<PatchDeform.gizmo>.rotation	Quat	默认值: (quat 0 0 0 1)	
			设置面片变形修改编辑器 gizmo 的旋转角度。可动画
<PatchDeform.gizmo>.scale	Point3	[1,1,1]	设置面片变形修改编辑器 gizmo 的放缩因子。可动画

**注意** 在 3ds max 的 MAXScript 里不能指定要变形的 patch 面片。

#### 10.4.46 Patch\_Select: Modifier (面片选择修改器)

Patch\_Select(面片选择)修改器可以在堆栈中为后续修改器向上传递一个子对象选择。它提供在编辑面片修改器中可用的选择功能的超集。可以选择顶点、边、面片和元素。也可以将选择从子对象层级更改到对象层级。

#### 10.4.47 PathDeform: Modifier (路径变形修改器)

##### 构造函数

PathDeform ...

##### 属性

属性名称	数据类型	默认值	说明
<PathDeform>.path	Node	undefined	选择变形路径对象
<PathDeform>.Percent_along_path	Float	0.0	依据指定的路径长度百分比, 确定对象在路径上的位置。可动画, 百分比
<PathDeform>.Stretch	Float	1.0	依据路径曲线并依据路径曲线的长度拉伸放缩对象, 对象的轴心点是放缩的依据。可动画
<PathDeform>.rotation	Float	0.0	围绕路径旋转对象。可动画, Angle
<PathDeform>.Twist	Float	0.0	围绕路径扭曲对象。扭曲角度基于旋转过程和路径的长度。可动画, Angle
<PathDeform>.axis	Integer	2	选择一个轴向以将路径变形框的轴向对齐于指定对象的局部坐标系轴向: 0: X; 1: Y; 2: Z
<PathDeform>.Flip_deformation_axis	Integer	0	设置是否沿指定的轴反转路径变形 gizmo: 0: 不反转; 1: 反转
<PathDeform>.gizmo	SubAnim		在次级结构层级, 可以移动、旋转、放缩路径变形修改编辑器 gizmo, 还可以为 gizmo 指定变换动画

(续表)

属性名称	数据类型	默认值	说明
<PathDeform.Gizmo>.position	Point3	[0,0,0]	设置路径变形修改编辑器 gizmo 的位置。可动画
<PathDeform.Gizmo>.rotation	Quat	(quat 0 0 0 1)	设置路径变形修改编辑器 gizmo 的旋转角度。可动画
<PathDeform.Gizmo>.scale	Point3	[1,1,1]	设置路径变形修改编辑器 gizmo 的放缩因子。可动画

#### 10.4.48 Point\_Cache: Modifier（点缓存修改器）

可以使用 Point\_Cache（点缓存）修改器将修改器动画存储到只记录顶点位置改变的磁盘文件中，然后使用磁盘文件中的信息而不是修改器关键帧来播放动画。当修改器动画所需的计算过多，并因此导致了动画播放的缓慢运行或丢帧时，该修改器十分有用。

该修改器的另一个应用是将同样的动画应用到多个对象上，并为每个动画改变开始时间和强度设置使得其运动不相同。

点缓存修改器也有可用的世界空间版本，其用法相同。

##### 构造函数

Point\_Cache ...

PointCache

##### 属性

属性名称	数据类型	默认值	说明
<Point_Cache>.time	Float	0.0	
<Point_Cache>.start_time	Float	0.0	设置记录顶点动画的第一帧
<Point_Cache>.end_time	Float	100.0	设置记录顶点动画的最后一帧
<Point_Cache>.samples	Integer	1	
<Point_Cache>.cache_file	undefined	undefined	从磁盘上的缓存文件中加载顶点动画到点缓存修改器中。如果缓存文件中的顶点编号不匹配对象中的顶点编号，那么会出现警告，但是不会发生错误
<Point_Cache>.relativeOffset <Point_Cache>.Relative_Offset	Boolean	False	当设置为 On 时，相对于其记录的位置启用对动画顶点位置偏移的设置。注意：启用使用相对偏移并在启用修改器的情况下播放缓存动画，缓存的顶点位置会相对于其由修改器计算所得位置进行计算。例如，如果将一个弯曲动画记录到缓存文件中，然后在同时启用使用相对偏移和弯曲修改器并将强度设置为 1.0 的情况下播放该动画，那么所有顶点位置都翻倍，这会导致夸大的运动
<Point_Cache>.strength	Float	1.0	当.relativeOffset=On 时，设置强度值。范围为 -10.0~10.0。值为 1.0 时，动画按照记录播放。强度值在 0~1 之间时，动画会相对受到抑制；强度值大于 1 时，动画被夸大；强度值设置为负值时，运动被反转。可动画

## 方法

### 1. record()

存取顶点动画到磁盘缓存器文件中。

### 2. setCache()

从磁盘缓存器文件中装载顶点动画到点缓存修改器。如果该文件中的顶点数量与对象数量不相匹配，会出现警告提示，但不会出错。

### 3. enableMods()

打开点缓存修改器下面的所有修改器堆栈。

### 4. disableMods()

当回放动画时，为了只显示点缓存动画，关闭点缓存修改器下面的所有对象的修改器堆栈。

## 10.4.49 Poly\_Select: Modifier (多边形选择修改器)

### 构造函数

Poly\_Select ...

PolyMesh Select ...

### 属性

属性名称	数据类型	默认值	说明
<Poly_Select>.useSoftSelection <Poly_Select>.Use_Soft_Selection	Boolean	False	当设置为 On 时，将样条线曲线变形应用到进行变化的选择周围的未选定子对象上
<Poly_Select>.softselUseEdgeDistance <Poly_Select>.Use_Edge_Distance	Boolean	False	
<Poly_Select>.softselEdgeDist <Poly_Select>.Edge_Distance	Integer	1	设置边距离。可动画
<Poly_Select>.softselAffectBackfacing <Poly_Select>.Affect_Backfacing	Boolean	False	当设置为 On 时，那些线方向与选定子对象平均法线方向相反的、取消选择的面就会受到软选择的影响。在顶点和边的情况下，这将应用到它们所依附的面的法线上
<Poly_Select>.softselFalloff	Float	20.0	用以定义影响区域的距离，它是用当前单位表示的从中心到球体的边的距离。使用较高的衰减设置以获得更平缓的倾斜，这取决于几何体的比例。注意：用衰减设置指定的区域在视口中用图形的方式进行了描述，所采用的图形方式与顶点和或边（或者用可编辑的多边形和面片，也可以是面）的颜色渐变相类似。渐变的范围为从选择颜色（通常是红色）到未选择的子对象颜色（通常是蓝色）。另外，在更改衰减设置时，渐变就会实时地进行更新。可动画

(续表)

属性名称	数据类型	默认值	说明
<Poly_Select>.softselPinch <Poly_Select>.Pinch	Float	0.0	沿着垂直轴升高或降低曲线的最高点设置区域的相对指向。当该值为负，就会产生一个坑而不是一个点；当设置成 0 时，收缩就会在该轴上产生平滑的变换。可动画
<Poly_Select>.softselBubble <Poly_Select>.Bubble	Float	0.0	沿着垂直轴展开和收缩曲线设置区域的相对饱满。它受收缩的限制，用以设置膨胀的固定开始点。收缩设为 0 并且膨胀设为 1.0 将会产生最为平滑的凸起；负的膨胀值会将曲线的底部移到曲面以下，在区域的地基周围创建了谷地。可动画
<Poly_Select>.byVertex <Poly_Select>.Select_By_Vertex	Boolean	False	设置是否在当前层级选中使用顶点的任何子对象，应用到所有子对象层级除了顶点
<Poly_Select>.ignoreBackfacing <Poly_Select>.Ignore_Backfacing_in_Selections	Boolean	False	当设置为 On 时，选择子对象时只选择其法线在视口中可见的这些子对象；否则，选择包含所有的子对象，而无论其法线方向
<Poly_Select>.materialID <Poly_Select>.Material_ID	Integer	1	设置材质的 ID 号码
<Poly_Select>.lockSoftSel <Poly_Select>.Lock_Soft_Selection	Boolean	False	设置是否锁定软选择
<Poly_Select>.paintSelSize <Poly_Select>.Brush_Size	Float	20.0	设置用以绘制选择的圆形笔刷的半径
<Poly_Select>.paintSelStrength <Poly_Select>.Brush_Strength	Float	1.0	设置绘制子对象的速率。高的强度值可以快速的达到完全值，而低的强度值需要重复的应用才可以达到完全值
<Poly_Select>.paintSelValue <Poly_Select>.Selection_Value	Float	1.0	设置软选择的最大相对选择。笔刷半径内周围顶点的值会朝着 0 衰减

#### 10.4.50 Preserve: Modifier (保留修改器)

##### 构造函数

Preserve ...

##### 属性

属性名称	数据类型	默认值	说明
<Preserve>.iterations	Integer	25	指定维护计算的次数，该参数设置得越高，维护操作的结果越接近于原始对象的形态；如果指定为 0，则不施加维护操作。可动画
<Preserve>.Edge_Length_Weight	Float	1.0	指定边长度。可动画
<Preserve>.Face_Angle_Weight	Float	0.3	指定面角度。可动画
<Preserve>.Volume_Weight	Float	0.3	指定体积值。可动画

(续表)

属性名称	数据类型	默认值	说明
<Preserve>.Apply_To_Whole_Mesh	Integer	0	当设置为 On 时，忽略在修改编辑堆栈中的下层修改编辑器向上传递的次级结构对象选择集：0: Off; 1: On
<Preserve>.Selected_Verts_Only	Integer	0	当设置为 On 时，对修改编辑堆栈中向上传递的次级结构节点选择集进行维护操作：0: Off; 1: On
<Preserve>.Invert_Selection	Integer	0	当设置为 On 时，反转修改编辑堆栈中向上传递的次级结构节点选择集，对原先所有未被选择的节点进行维护操作：0: Off; 1: On

#### 10.4.51 Push: Modifier (推动修改器)

##### 构造函数

Push ...

##### 属性

属性名称	数据类型	默认值	说明
<Push>.Push_Value	Float	0.0	设置顶点从对象中心偏移的距离，单位为 World 单位，正值表示向外移动顶点，负值表示向内移动顶点。可动画

#### 10.4.52 Relax: Modifier (松弛修改器)

##### 构造函数

Relax ...

##### 属性

属性名称	数据类型	默认值	说明
<Relax>.iterations	Integer	1	设置松弛操作的重复次数，每一次重复计算使用的顶点平均位置都是上一次重复计算的结果。可动画
<Relax>.Relax_Value	Float	0.5	设置顶点靠近或远离相邻顶点的距离，百分比，正值表现收缩效果，负值表现膨胀效果。取值范围[-1.0 1.0]，可动画
<Relax>.Keep_Boundary_Pts_Fixed	Integer	1	当设置为 On 时，边界上的顶点不受松弛操作影响。0: Off; 1: On

#### 10.4.53 Ripple: Modifier (涟漪修改器)

##### 构造函数

Ripple ...

### 属性

属性名称	数据类型	默认值	说明
<Ripple>.amplitude1 <Ripple>.Amplitude_1	Float	5.0	指定沿涟漪修改器平面 X 轴向的振幅。可动画
<Ripple>.amplitude2 <Ripple>.Amplitude_2	Float	5.0	指定沿涟漪修改器平面 Y 轴向的振幅。可动画
<Ripple>.wavelength <Ripple>.Wave_Length	Float	25.0	指定每个涟漪的长度。可动画
<Ripple>.phase	Float	0.0	指定涟漪距离涟漪中心的偏移相位。可动画
<Ripple>.decay	Float	0.0	设置涟漪的衰减值。如果设置为 0.0，则涟漪扭曲在整个作用空间中的振幅都是一致的；如果设定了衰减数值，远离修改编辑器 gizmo 中心位置的涟漪强度逐渐减弱。可动画
<Ripple>.center	Point3	[0,0,0]	在次级结构层级，可以移动、旋转、放缩涟漪编辑器 gizmo 的中心对象，还可以为涟漪编辑器 gizmo 的中心对象指定变换动画。可动画
<Ripple>.gizmo	SubAnim		在次级结构层级，可以移动、旋转、放缩涟漪编辑器 gizmo，还可以为 gizmo 指定变换动画
<Ripple.gizmo>.rotation	Quat	(quat 0 0 0 1)	设置涟漪编辑器 gizmo 的旋转角度。可动画
<Ripple.gizmo>.scale	Point3	[1,1,1]	设置涟漪编辑器 gizmo 的放缩因子。可动画

### 10.4.54 Skew: Modifier（倾斜修改器）

#### 构造函数

Skew ...

#### 属性

属性名称	数据类型	默认值	说明
<Skew>.amount	Float	25.0	设置倾斜的角度。可动画
<Skew>.direction	Float	0.0	指定倾斜相对于水平面的方向。可动画
<Skew>.axis	Integer	2	指定倾斜操作依据的轴向：0: X; 1: Y; 2: Z
<Skew>.limit	Boolean	False	当设置为 On 时，使用限定效果
<Skew>.upperlimit <Skew>.Upper_Limit	Float	0.0	指定倾斜操作的上限范围。可动画
<Skew>.lowerlimit <Skew>.Lower_Limit	Float	0.0	指定倾斜操作的下限范围。可动画
<Skew>.center	Point3	[0,0,0]	在次级结构层级，可以移动、旋转、放缩倾斜编辑器 gizmo 的中心对象，还可以为倾斜编辑器 gizmo 的中心对象指定变换动画。可动画
<Skew>.gizmo	SubAnim		在次级结构层级，可以移动、旋转、放缩倾斜编辑器 gizmo，还可以为 gizmo 指定变换动画

(续表)

属性名称	数据类型	默认值	说明
<Skew.Gizmo>.position	Point3	[0,0,0]	设置倾斜编辑器 gizmo 的位置。可动画
<Skew.Gizmo>.rotation	Quat	(quat 0 0 0 1)	设置倾斜编辑器 gizmo 的旋转角度。可动画
<Skew.Gizmo>.scale	Point3	[1,1,1]	设置倾斜编辑器 gizmo 的缩放因子。可动画

#### 10.4.55 Skin: Modifier (蒙皮修改器)

##### 构造函数

Skin ...

##### 属性

属性名称	数据类型	默认值	说明
<Skin>.Effect	Float	0.0	设置骨骼对选定的节点施加绝对权重的效果
<Skin>.filter_vertices	Boolean	False	开/关节点选择
<Skin>.filter_cross_sections	Boolean	True	开/关交叉截面选择
<Skin>.filter_envelopes	Boolean	True	开/关骨骼封套选择
<Skin>.draw_all_envelopes	Boolean	False	当设置为 On 时，显示所有的封套；当设置为 Off 时，只显示所选择的封套
<Skin>.draw_vertices	Boolean	True	开/关显示节点的不同权重
<Skin>.ref_frame	Integer	0	指定骨骼与网格蒙皮在参考位置时的帧数
<Skin>.paint_radius	Float	24.0	指定喷笔的半径大小
<Skin>.paint_feather	Float	0.7	指定喷绘区域的衰减量
<Skin>.cross_radius	Float	10.0	设置所选择封套的交叉截面放缩量
<Skin>.always_deform	Boolean	True	当设置为 On 时，用于编辑骨骼与所有控制节点之间的相对关系
<Skin>.paint_str	Float	0.1	设置喷绘制作的强度
<Skin>.localSquash	Array	#()	Float 数组，指定骨骼被拉伸时的挤压倍增量
<Skin>.initialSquash	Array	#()	Float 数组，包含骨骼的初始挤压倍增量
<Skin>.initialStaticEnvelope	Boolean	False	如果设置为 On 时，当添加骨骼时，使用初始静态封套大小；如果设置为 Off 时，当添加骨骼时，封套大小由对象绑定框（bounding box）大小决定
<Skin>.initialInnerEnvelopePercent	Float	0.75	当属性.initial StaticEnvelope 为 False 时，设置用来查找内横断面大小的倍增器值
<Skin>.initialOuterEnvelopePercent	Float	3.5	当属性.initial StaticEnvelope 为 False 时，设置用来查找外横断面大小的倍增器值
<Skin>.initialEnvelopeInner	Float	10.0	当属性.initial StaticEnvelope 为 True 时，设置用内横断面大小

(续表)

属性名称	数据类型	默认值	说明
<Skin>.initialEnvelopeOuter	Float	50.0	当属性.initial StaticEnvelope 为 True 时，设置用外横断面大小
<Skin>.draw_all_gizmos	Boolean	True	开/关显示所有的装置
<Skin>.draw_all_vertices	Boolean	False	开/关显示所有节点
<Skin>.shadeweights	Boolean	True	开/关以色彩方式显示节点的权重
<Skin>.envelopesAlwaysOnTop	Boolean	True	开/关将封套显示在所有对象的最上方
<Skin>.rossSectionsAlwaysOnTop	Boolean	True	开/关将交叉截面显示在所有对象的最上方
<Skin>.rigid_vertices	Boolean	False	当设置为 On 时，指定节点的权重只受一个骨骼的影响
<Skin>.rigid_handles	Boolean	False	当设置为 On 时，对于面片模型，迫使面片控制手柄的权重等于节点的权重
<Skin>.fast_update	Boolean	False	当设置为 On 时，在视图中不显示权重变形的效果
<Skin>.gizmos	Array	#()	Max 对象数组，包含所有依附蒙皮的装置
<Skin>.backTransform	Boolean	True	用于将网格链接到骨骼结构。通常，在执行此操作时，任何骨骼移动都会根据需要将网格移动两次，一次随骨骼移动，一次随链接移动
<Skin>.bone_Limit	Integer	20	限制可影响一个顶点的骨骼数
<Skin>.debugMode	Boolean	True	设置修改器调试模式。当设置为 On 时，数据将被输出到 Listener 窗口，这一点对于跟踪调试非常有用
<Skin>.shortenBoneNames <Skin>.Shorten_Bone_Names	Boolean	True	设置被缩短的骨骼名称方式。当设置为 On 时，骨骼名称超出显示的部分将被缩短仅显示首尾的字符，中间部分用“...”表示
<Skin>.showNoEnvelopes <Skin>.Shade_Weights	Boolean	False	即使已选择封套，也不显示封套
<Skin>.updateOnMouseUp	Boolean	False	按下鼠标按钮时，不进行更新，松开鼠标按钮时，进行更新
<Skin>.wt_activeVertexSet <Skin>.Active_Vertex_Set	Integer	0	设置在顶点选择下拉列表中顶点的选择方式： 0: All Vertices（所有顶点） 1: Selected Vertices（选中的顶点） 2: Selected Bone（仅选中被骨骼影响的顶点）
<Skin>.wt_affectSelected <Skin>.Affect_Selected	Boolean	False	存取顶点权重表中影响选择的模式
<Skin>.wt_attribLabelHeight <Skin>.Attrib_Label_Height	Integer	1	设置顶点权重表中属性标签的高度
<Skin>.wt_dragLeftRightMode <Skin>.Drag_Left_Right_Mode	Boolean	True	将顶点权重拖到左边或右边用以更改顶点权重

(续表)

属性名称	数据类型	默认值	说明
<Skin>.wt_flipUI <Skin>.Flip_UI	Boolean	False	存取顶点权重表 UI 反转模式。当设置为 On 时，在左侧从上到下显示骨骼名称
<Skin>.wt_fontSize <Skin>.FontSize	Integer	14	设置顶点权重表文字大小
<Skin>.wt_markerColor <Skin>.Marker_Color	Color	默认值: (color 255 0 255)	
		设置顶点权重表标记颜色	
<Skin>.wt_markerType <Skin>.Marker_Type	Integer	6	设置顶点权重表标记类型
<Skin>.wt_precision <Skin>.Precision	Float	0.01	设置顶点权重表标记类型拖曳时的精确度
<Skin>.wt_shortenLabels <Skin>.Shortern_Labels	Boolean	True	当设置为 On 时，顶点权重表标记将会被缩短
<Skin>.wt_showAffectedBones <Skin>.wt_Show_Affected_Bones	Boolean	False	当设置为 On 时，仅显示影响当前所显示顶点的骨骼
<Skin>.wt_showAttributes <Skin>.Show_Attributes	Boolean	True	当设置为 On 时，切换 S/M/N/R/H 属性的显示
<Skin>.wt_showCopyPasteUI <Skin>.Show_Copy_and_Paste_UI	Boolean	True	设置是否显示复制、粘贴等操作信息
<Skin>.wt_showExclusions <Skin>.Show_Exclusions	Boolean	False	在每个顶点权重字段中切换小复选框区域的显示。当设置为 On 时，将从骨骼中排除顶点
<Skin>.wt_showGlobal <Skin>.Show_Global	Boolean	False	存取球形模式的显示状态
<Skin>.wt_showLocks <Skin>.Show_Locks	Boolean	False	设置是否显示锁定的 toggle 盒子。对应权重表中 Options 菜单项
<Skin>.wt_showMarker <Skin>.Show_Weight_Table_Marker	Boolean	True	设置是否显示顶点权重表标记
<Skin>.wt_showMenu <Skin>.Show_Menu	Boolean	True	设置是否显示顶点权重表的菜单项。只有重新打开顶点权重表后该设置才生效
<Skin>.wt_showOptionsUI <Skin>.Show_Options_UI	Boolean	False	控制是否显示顶点集的信息
<Skin>.wt_showSetUI <Skin>.Show_Set_Sets_UI	Boolean	False	控制顶点选择面板是否在权重表对话框中显示
<Skin>.wt_tableY <Skin>.Top_Border	Integer	10	控制权重表的 Y 轴方向位置
<Skin>.wt_updateOnMouseUp <Skin>.Update_On_Mouse_Up	Boolean	False	按下鼠标时，不发生更新；释放鼠标时，发生更新。该选项可以避免不必要的更新，从而使工作流程快速移动
<Skin>.wt_winHeight <Skin>.Height	Integer	500	设置权重表对话框的高度。只有重新打开顶点权重表后该设置才生效

(续表)

属性名称	数据类型	默认值	说明
<Skin>.wt_winWidth <Skin>.Width	Integer	800	设置权重表对话框的宽度。只有重新打开顶点权重表后该设置才生效
<Skin>.wt_winXPos <Skin>.Xpos	Integer	0	设置权重表在 windows 桌面左上角的水平方向的位置，只有重新打开顶点权重表后该设置才生效
<Skin>.wt_winYPos <Skin>.Ypos	Integer	0	设置权重表在 windows 桌面左上角的垂直方向的位置，只有重新打开顶点权重表后该设置才生效
<Skin>.mirrorPlane <Skin>.Mirror_Plane	Integer	0	确定镜像平面的轴向： 0: X; 1: Y; 2: Z
<Skin>.mirrorOffset <Skin>.Mirror_Offset	Float	0.0	设置镜像平面的偏移值
<Skin>.mirrorUseInitialTM <Skin>.Use_Initial_TM	Boolean	True	当设置为 On 时，将回复到最初的转换矩阵
<Skin>.mirrorEnabled <Skin>.Mirror_Enabled	Boolean	False	当设置为 On 时，启用镜像模式。对应蒙皮修改器的 Mirror Mode 控件
<Skin>.mirrorThreshold <Skin>.Mirror_Threshold	Float	0.5	设置镜像阈值
<Skin>.mirrorProjection <Skin>.Mirror_Projection	Integer	0	设置镜像投影模式： 0: Default Display (默认显示) 1: Positive Direction (正值) 2: Negative Direction (负值) 3: None
<Skin>.manualUpdate <Skin>.Manual_Update	Boolean	False	当设置为 On 时，可手动更新显示，而不是在每次松开鼠标按钮时更新
<Skin>.mirrorFast <Skin>.Fast_Engine	Boolean	True	当设置为 On 时，使用快速镜像模式
<Skin>.ignoreBoneScale <Skin>.Ignore_Bone_Scale	Boolean	False	设置是否忽略骨骼比例
<Skin>.rightJustifyBoneText <Skin>.Right_Justify_Bone_Names	Boolean	True	当设置为 On 时，权重表对话框里的骨骼文本右对齐
<Skin>.paintBlendMode	Boolean	False	设置是否在混合模式绘制顶点权重

## 方法

使用下面方法之前，指定的 Skin Modifier 必须为修改器面板下当前正显示的修改器，修改器面板为活动面板。

1. skinops.addbone <Skin> <Bone\_Node> <Update\_Integer>

向当前系统添加一个骨骼。如果参数< Update\_Integer >为-1，会强制系统进行一次完全刷新；如果为 0，表示不刷新系统。使用 0 设置，可以让用户添加一系列骨骼之后，在最后一次添加时再更新系统。

2. skinops.addBoneFromViewEnd <Skin>

结束 addBoneFromViewStart 命令模式。

3. skinops.addBoneFromViewStart <Skin>

调用本函数后，系统会停下来等待用户用鼠标在视窗里选择骨骼，如果要让系统结束等待，可以右击鼠标或调用 addBoneFromViewEnd 方法。

4. skinops.addCrossSection <Skin> <BoneID\_Integer> \

<U\_Float> <InnerRadius\_Float> <OuterRadius\_Float>

向指定序号骨骼<BoneID\_Integer>添加一个横断面，参数<U\_Float>的取值范围为[0.0 1.0]，定义了横断面的添加位置。

5. skinops.addCrossSection <Skin> <U\_Float>

向当前被选择的骨骼添加一个横断面，横断面的内径和外径的计算基于相邻横断面。

参数<U\_Float>的取值范围为[0.0 1.0]，定义了横断面的添加位置。

6. skinops.addCrossSection <Skin> <U\_Float> <InnerRadius\_Float> <OuterRadius\_Float>

向当前被选择的骨骼添加一个横断面，横断面的内径和外径使用指定值。参数<U\_Float>的取值范围为[0.0 1.0]，定义了横断面的添加位置。

7. skinops.buttonAdd <Skin>

相当于按下界面的 add bone 按钮。

8. skinops.buttonAddCrossSection <Skin>

相当于按下界面的 add cross section 按钮。

9. skinops.buttonAddGizmo <Skin>

相当于按下界面的 add gizmo 按钮。

10. skinops.buttonCopyGizmo <Skin>

相当于按下界面的 copy gizmo 按钮

11. skinops.buttonExclude <Skin>

相当于按下界面的 exclude vertices 按钮。

12. skinops.buttonInclude <Skin>

相当于按下界面的 include vertice 按钮。

13. skinops.buttonPaint <Skin>

相当于按下界面的 paint 按钮。

14. skinops.buttonPasteGizmo <Skin>

相当于按下 gizmo rollout 界面的 paste gizmo 按钮。

15. skinops.buttonRemove <Skin>

相当于按下界面的 remove bone 按钮。

16. skinops.buttonRemoveCrossSection <Skin>

相当于按下界面的 remove cross section 按钮。

17. skinops.buttonRemoveGizmo <Skin>

相当于按下 gizmo rollout 界面的 remove gizmo 按钮。

18. skinops.buttonSelectExcluded <Skin>

相当于按下界面的 select excluded vertices 按钮。

19. skinops.copySelectedBone <Skin>

将当前选择骨骼的属性复制到缓存里。

20. skinops.enableGizmo <Skin> <GizmoID\_Integer> <Enable\_bool>

启用/禁止 gizmo 类型功能。参数<GizmoID\_Integer>为 gizmo 类型序号。

21. skinOps.GetBoneName <Skin> <bone\_Integer> <nameflag\_index>

返回一个字符串，表示指定骨骼的.name 属性。

参数<nameflag\_index>为 0 或 1。如果为 0，函数返回指定包含指定骨骼转换的对象；

如果为 1，函数返回 List Box 里列出的骨骼名称。

22. skinops.getBonePropEnvelopeVisible <Skin> <BoneID\_Integer>

返回指定骨骼的可见性，0 表示不可见，1 表示可见。

23. skinops.getBonePropFalloff <Skin> <BoneID\_Integer>

返回指定骨骼的 Falloff 参数：

◆ 1 线性 (linear)

◆ 2 正弦 (sinual)

◆ 3 快速淡出 (fast out)

◆ 4 缓慢淡出 (slow out)

24. skinops.getBonePropRelative <Skin> <BoneID\_Integer>

返回指定骨骼的 relative 属性，1 表示相对值，0 表示绝对值。

25. skinops.GetCrossSectionU <Skin> <BoneID\_Integer> <CrossSectionID\_Integer>

为指定骨骼的指定横断面设置 U 值。

26. skinops.getCurrentSelectGizmoType <Skin>

返回当前选择的 gizmo 类型。

27. skinops.GetEndPoint <Skin> <BoneID\_Integer>

返回一个 Point3 值，表示指定骨骼的终点。

28. skinOps.GetInnerRadius <Skin> <bone\_Integer> <CrossSectionID\_Integer>

返回指定骨骼的横断面内径。

29. skinOps.GetNumberBones <Skin>

返回指定系统里的骨骼数。

30. skinOps.getNumberCrossSections <Skin> <bone\_Integer>

返回指定骨骼里的横断面数。

31. skinops.getNumberOfGizmos <Skin>

返回指定系统里的 gizmo 数。

32. skinops.getNumberOfGizmoTypes <Skin>

返回指定系统里的 gizmo 类型数。

33. skinOps.GetNumberVertices <Skin>

返回 Skin Modifier 应用对象里的顶点数。

34. skinOps.GetOuterRadius <Skin> <bone\_Integer> <CrossSectionID\_Integer>

返回指定骨骼横断面的外径。

35. `skinOps.GetSelectedBone <Skin>`

返回当前选择骨骼在骨骼列表里的序号。

36. `skinops.getSelectedBonePropEnvelopeVisible <Skin>`

返回当前选择骨骼的可见性参数。0 表示不可见，1 表示可见。

37. `skinops.getSelectedBonePropFalloff <Skin>`

返回当前选择骨骼的 falloff 参数：

- ◆ 1 线性 (linear)

- ◆ 2 正弦 (sinual)

- ◆ 3 快速淡出 (fast out)

- ◆ 4 缓慢淡出 (slow out)

38. `skinops.getSelectedBonePropRelative <Skin>`

返回当前选择骨骼的 relative 属性，1 表示相对值，0 表示绝对值。

39. `skinops.getSelectedGizmo <Skin>`

返回当前选择的 gizmo。

40. `skinops.GetStartPoint <Skin> <BoneID_Integer>`

返回一个 Point3 值，表示指定骨骼的起点。

41. `skinOps.GetVertexWeight <Skin> <vertex_Integer> <vertex_bone_Integer>`

返回指定骨骼对指定顶点的影响权重。

42. `skinOps.GetVertexWeightBoneID <Skin> <vertex_Integer> <vertex_bone_Integer>`

返回对指定顶点产生影响的第 N 个骨骼的序号。

43. `skinOps.GetVertexWeightCount <Skin> <vertex_Integer>`

返回所有对指定顶点产生影响的骨骼数目。

44. `skinops.isBoneSelected <Skin> <BoneID_Integer>`

如果指定序号的骨骼当前被选择，返回 1，否则返回 0。

45. `skinOps.IsVertexModified <Skin> <vertex_Integer>`

如果指定顶点被修改过，返回 1，否则返回 0。“被修改过的顶点”指那些曾经由手工或 Paint 指定了权重的顶点，未被修改的顶点权重仅取决于其封套。

46. `skinOps.IsVertexSelected <Skin> <vertex_Integer>`

如果指定顶点当前被选择，返回 1，否则返回 0。

47. `skinops.loadEnvelope <Skin>`

打开 Load Envelopes 对话框。

48. `skinops.loadEnvelope <Skin> <filename_string>`

将封套加载到指定文件。

49. `skinops.multiRemove <Skin>`

打开一个列表框，用户可以一次选择删除多个骨骼，而不是一次删除一个。

50. `skinops.pasteToAllBones <Skin>`

将缓存的内容粘贴到所有骨骼。

51. skinops.pasteToBone <Skin><BoneID\_int>

将缓存的内容粘贴到指定序号骨骼。

52. skinops.pasteToSelectedBone <Skin>

将缓存的内容粘贴到当前选择的骨骼。

53. skinops.removebone <Skin>

将当前选择的骨骼从骨骼系统删除。

54. skinops.removebone <Skin> <BoneID\_Integer>

将指定序号的骨骼从骨骼系统删除。

55. skinops.RemoveCrossSection \$<Skin>

将当前选择的横断面从当前选择的骨骼里删除。

56. skinops.RemoveCrossSection <Skin> <BoneID\_Integer> <CrossSectionID\_Integer>

将指定横断面从指定序号的骨骼里删除。

57. skinOps.ReplaceVertexWeights <Skin> <vertex\_Integer> \

(<vertex\_bone\_Integer> | <vertex\_bone\_array>) \

(<weight\_Float> | <weight\_array>)

为指定骨骼（或骨骼选择集）设置对指定顶点的影响权重。原来设置的权重信息会被删除。如果指定<vertex\_bone\_array>和<weight\_array>两个数组，其长度必须相等。

58. skinops.resetAllBones <Skin>

重置骨骼系统。

59. skinops.resetSelectedBone <Skin>

重置当前选择的骨骼。

60. skinops.resetSelectedVerts <Skin>

重置当前选择的顶点。

61. skinops.saveEnvelope <Skin>

打开 Save Envelopes 对话框。

62. skinops.saveEnvelope <Skin> <filename\_string>

将封套存储到指定文件里。

63. skinOps.SelectBone <Skin> <bone\_Integer>

从骨骼列表里选择指定序号的骨骼。

64. skinops.selectCrossSection <Skin> \

<CrossSectionID\_Integer> <Inner\_Outer\_Integer>

从当前选择的骨骼里选择指定序号的横断面。

如果参数 Inner\_Outer\_Integer 为 0，表示内部封套，1 表示外部封套。

65. skinops.selectEndPoint <Skin>

选择当前选择骨骼封套的起点。

66. skinops.selectGizmo <Skin> <gizmoID\_int>

选择指定序号的 gizmo。

67. skinops.selectGizmoType <Skin> <gizmoTypeID\_int>

选择指定序号的 gizmo 类型。

68. skinops.selectNextBone <Skin>

选择骨骼列表里的下一个骨骼。

69. skinops.selectPreviousBone <Skin>

选择骨骼列表里的前一个骨骼。

70. skinops.selectStartPoint <Skin>

选择当前选择骨骼封套的终点。

71. skinOps.SelectVertices <Skin> \

(<vertex\_Integer> | <vertex\_array> | <<vertex\_bitarray>>)

选择指定顶点。

72. skinops.setBonePropEnvelopeVisible <Skin> \

<BoneID\_Integer> <Visible\_Integer>

为指定序号的骨骼指定可见性参数。参数<Visible\_Integer>如果为 0，表示可见，如果为 1，表示不可见。

73. skinops.setBonePropFalloff <Skin> <BoneID\_Integer> <Falloff\_Integer>

为指定骨骼设置 Falloff 参数：

◆ 1：线性 (linear)

◆ 2：正弦 (sinual)

◆ 3：快速淡出 (fast out)

◆ 4：缓慢淡出 (slow out)

74. skinops.setBonePropRelative <Skin> <BoneID\_Integer> <Relative\_Integer>

为指定骨骼设置 relative 参数。如果参数<Relative\_Integer>为 0 表示绝对，1 表示相对。

75. skinops.SetCrossSectionU <Skin> <BoneID\_Integer> \

<CrossSectionID\_Integer> <U\_Float>

为指定骨骼的指定横断面设置 U 值。

76. skinops.SetEndPoint <Skin> <BoneID\_Integer> <EndPointPos\_point3>

为指定骨骼设置终点。

77. skinOps.SetInnerRadius <Skin> <bone\_Integer> \

<CrossSectionID\_Integer> <Radius\_Float>

为指定骨骼的指定横断面设置内径。

78. skinOps.SetOuterRadius <Skin> <bone\_Integer> \

<CrossSectionID\_Integer> <Radius\_Float>

为指定骨骼的指定横断面设置外径。

79. skinops.setSelectedBonePropEnvelopeVisible <Skin> <Visible\_Integer>

为当前选择的骨骼指定可见性参数。参数<Visible\_Integer>如果为 0，表示可见；如果为 1，表示不可见。

80. skinops.setSelectedBonePropFalloff <Skin> <Falloff\_Integer>

为当前选择的骨骼设置 Falloff 参数：

- ◆ 1: 线性 (linear)
- ◆ 2: 正弦 (sinual)
- ◆ 3: 快速淡出 (fast out)
- ◆ 4: 缓慢淡出 (slow out)

81. skinops.setSelectedBonePropRelative <Skin> <Relative\_Integer>

为当前选择的骨骼设置 relative 参数。如果参数<Relative\_Integer>为 0 表示相对，1 表示绝对。

82. skinops.SetStartPoint <Skin> <BoneID\_Integer> <EndPointPos\_point3>

为指定骨骼设置起点。

83. skinOps.SetVertexWeights <Skin> <vertex\_Integer> \

(<vertex\_bone\_Integer> | <vertex\_bone\_array>) \

(<weight\_Float> | <weight\_array>)

为指定骨骼设置对指定顶点的影响权重。先前设置的该骨骼对其他顶点的影响权重被保留。如果参数骨骼和顶点都指定为数组，两者的长度必须相等。

84. skinops.ZoomToBone <Skin> <All\_boolean>

将当前选择的骨骼进行视窗缩放。如果参数<All\_boolean>为 True，表示缩放所有视窗，如果参数<All\_boolean>为 False，表示仅缩放活动视窗。

85. skinops.ZoomToGizmo <Skin> <All\_boolean>

将当前选择的 gizmo 进行视窗缩放。如果参数<All\_boolean>为 True，表示缩放所有视窗，如果参数<All\_boolean>为 False，表示仅缩放活动视窗。

#### 10.4.56 Skin\_Morph: Modifier（蒙皮变形修改器）

##### 构造函数

Skin\_Morph ...

MorphByBone ...

##### 属性

属性名称	数据类型	默认值	说明
<Skin_Morph>.bones <Skin_Morph>.Bone_Nodes	Array-Parameter	#()	存取骨骼的阵列
<Skin_Morph>.useSoftSelection <Skin_Morph>.Use_Soft_Selection	Boolean	False	是否启用软选择
<Skin_Morph>.selectionRadius <Skin_Morph>.Selection_Radius	Float	10.0	以系统单位数确定软选择区域的范围
<Skin_Morph>.useEdgeLimit <Skin_Morph>.Use_Edge_Limit	Boolean	False	当设置为 On 时，蒙皮变形使用边限制数值设置，根据一个或多个选定顶点的边数确定软选择区域的范围
<Skin_Morph>.edgeLimit <Skin_Morph>.Edge_Limit	Integer	10	设置边限制数值

(续表)

属性名称	数据类型	默认值	说明
<Skin_Morph>.falloffGraphs <Skin_Morph>.Falloff_Graphs;	Array-Parameter	#()	
<Skin_Morph>.softSelection <Skin_Morph>.Soft_Selection_Graph	SubAnim	默认值: ReferenceTarget	
		设置软选择曲线图形, 可动画	
<Skin_Morph.softSelectionGraph>.curve_1	SubAnim	默认值: SubAnim:Curve_1	
		设置软选择曲线	
<Skin_Morph>.targetNodes <Skin_Morph>.Node_List	Array-Parameter	#()	用于使用不同网格作为变形目标
<Skin_Morph>.enabled	Boolean	True	设置变形是否处于活动状态
<Skin_Morph>.MorphName	String	默认值: "Morph 0"	
		设置当前变形的名称	
<Skin_Morph>.influenceAngle <Skin_Morph>.Influence	Float	90.0	骨骼当前方向周围的角度, 在该范围内变形生效
<Skin_Morph>.falloff	Integer	0	确定骨骼在其影响角度内移动时, 变形的更改速率。有以下几种衰减类型: 0: Linear (线性) 1: Sinual (波形) 2: Fast (快速) 3: Slow (慢速) 4: Custom Falloff (自定义衰减)
<Skin_Morph>.jointType <Skin_Morph>.Joint_Type	Integer	0	设置关节的类型: 0: Ball Joint (球关节, 跟踪骨骼的所有旋转) 1: Planar Joint (平面关节, 仅在骨骼的父骨骼平面上跟踪骨骼的旋转)
<Skin_Morph>.reloadSelected <Skin_Morph>.SOnly_Reload_Selected_Vertices	Boolean	False	使用外部网格中的已编辑顶点位置更新蒙皮变形对象
<Skin_Morph>.showMirrorPlane <Skin_Morph>.Show_Mirror_Plane	Boolean	False	当设置为 On 时, 在视口中将镜像平面显示为红色矩形 gizmo
<Skin_Morph>.previewBone <Skin_Morph>.Preview_Bones	Boolean	False	当设置为 On 时, 在视口中使用红色高亮显示目标骨骼
<Skin_Morph>.previewVerts <Skin_Morph>.Preview_Verts	Boolean	False	当设置为 On 时, 在视口中使用红色显示可变形的合格顶点, 以及源顶点中存在的任何动画
<Skin_Morph>.mirrorPlane <Skin_Morph>.Mirror_Plane	Integer	0	设置镜像平面的轴: 0: X; 1: Y; 2: Z
<Skin_Morph>.mirrorOffset <Skin_Morph>.Mirror_Offset	Float	0.0	设置镜像偏移值

(续表)

属性名称	数据类型	默认值	说明
<Skin_Morph>.mirrorThreshold <Skin_Morph>.Mirror_Threshold	Float	1.0	设置镜像阈值
<Skin_Morph>.safemode <Skin_Morph>.Safe_Mode	Boolean	True	当设置为 On 时，必须使用创建变形按钮才能创建变形，使用编辑按钮才能编辑变形
<Skin_Morph>.showDriverBone <Skin_Morph>.Show_Driver_Bone	Boolean	True	设置是否显示驱动器骨骼矩阵
<Skin_Morph>.showMorphBone <Skin_Morph>.Show_Morph_Bone	Boolean	True	设置是否显示变形骨骼矩阵
<Skin_Morph>.showCurrentAngle <Skin_Morph>.Show_Current_Angle	Boolean	True	设置是否显示描述驱动器骨骼矩阵与变形骨骼矩阵之间的当前角度
<Skin_Morph>.showEdges <Skin_Morph>.Show_Edges	Boolean	True	设置是否以橙色高亮显示与可变形顶点连接的边
<Skin_Morph>.matrixSize <Skin_Morph>.Matrix_Size	Float	10.0	设置矩阵大小
<Skin_Morph>.boneSize <Skin_Morph>.Bone_Size	Float	1.0	设置骨骼大小
<Skin_Morph>.showLimitAngle <Skin_Morph>.Show_Limit_Angle	Boolean	False	当设置为 On 时，在视窗显示限制角度
<Skin_Morph>.showDeltas <Skin_Morph>.Show_Deltas	Boolean	True	
<Skin_Morph>.showX <Skin_Morph>.Show_X	Boolean	True	
<Skin_Morph>.showY <Skin_Morph>.Show_Y	Boolean	True	
<Skin_Morph>.showZ <Skin_Morph>.Show_Z	Boolean	True	

## 方法

### 1. addBone <node>node

添加指定的骨骼到参数卷展栏下的骨骼列表中。

### 2. removeBone <node>node

从参数卷展栏下的骨骼列表中删除指定的骨骼。

### 3. selectBone <node>node <string>morphName

从指定的变形中选择指定的骨骼。

### 4. selectVertices <node>node <bitArray>sel

从指定的 Bitarray 值中选择顶点。

### 5. isSelectedVertex <node>node <Integer>vertexIndex <Boolean>

当指定的节点中的索引顶点被选择时，返回 True。

### 6. resetSelectionGraph()

重新设置软选择图表。相当于在选择卷展栏里按下 Reset Graph 按钮。

7. ring()

相当于在选择卷展栏里按下 Ring 按钮。

8. loop()

相当于在选择卷展栏里按下 Loop 按钮。

9. shrink()

相当于在选择卷展栏里按下 Shrink 按钮。

10. grow()

相当于在选择卷展栏里按下 Grow 按钮。

11. createMorph <node>node

相当于在局部属性卷展栏里按下 Create Morph 按钮。

12. removeMorph <node>node <string>morphName

相当于在局部属性卷展栏里按下 Delete Morph 按钮。

13. editMorph <boolean>edit

相当于在局部属性卷展栏里按下 Edit 按钮。

14. clearSelectedVertices()

相当于在局部属性卷展栏里按下 Clear Verts 按钮。

15. resetOrientation <node>node <string>morphName

相当于在局部属性卷展栏里按下 Reset Orient.按钮。

16. deleteSelectedVertices()

相当于在局部属性卷展栏里按下 Remove Verts 按钮。

17. editFalloffGraph <node>node <string>morphName

相当于在局部属性卷展栏里按下 G 按钮。

18. setExternalNode <node>morphnode <string>morphName <node>externalnode

相当于在局部属性卷展栏里按下 External Mesh 按钮。

19. reloadTarget <node>node <string>morphName

相当于在局部属性卷展栏里按下 Reload Target 按钮。

20. mirrorPaste <node>node

相当于按下 Paste Mirror 按钮。

21. moveVerts <point3>vec

沿指定的向量移动选择的顶点。

22. transformVerts <matrix3>tmMotion <matrix3>tmToLocalSpace

转换给定矩阵里的顶点。

23. boneGetInitialNodeTM <node>node <matrix3>

返回指定骨骼节点的初始节点转换矩阵。

24. boneSetInitialNodeTM <node>node <matrix3>tm

设置指定骨骼节点转换矩阵值。

25. boneGetObjectTM <node>node <matrix3>

返回指定骨骼节点的初始对象转换矩阵。

26. boneSetInitialObjectTM <node>node <matrix3>tm

设置指定骨骼节点的对象转换矩阵值。

27. boneGetInitialParentTM <node>node <matrix3>

返回指定骨骼节点的初始父对象转换矩阵。

28. boneSetInitialParentTM <node>node <matrix3>tm

设置指定骨骼节点的父对象转换矩阵值。

29. numberofBones()<Integer>

返回修改器里的骨骼数量。

30. boneGetNumberOfMorphs <node>node <Integer>

返回用于指定骨骼节点的变形数量。

31. boneGetMorphName <node>node <Integer>index <string>

返回用于指定骨骼节点的索引变形名称。

32. boneSetMorphName <node>node <Integer>index <string>name

指定用于指定骨骼节点的索引变形名称。

33. boneGetMorphAngle <node>node <Integer>index <Float>

返回用于指定骨骼节点的索引变形角度。

34. boneSetMorphAngle <node>node <Integer>index <Float>angle

指定用于指定骨骼节点的索引变形角度。

35. boneGetMorphTM <node>node <Integer>index <matrix3>

返回用于指定骨骼节点的索引变形转换矩阵。

36. boneSetMorphTM <node>node <Integer>index <matrix3>tm

指定用于指定骨骼节点的索引变形转换矩阵值。

37. boneGetMorphParentTM <node>node <Integer>index <matrix3>

返回用于指定骨骼节点的索引父对象变形转换矩阵。

38. boneSetMorphParentTM <node>node <Integer>index <matrix3>tm

指定用于指定骨骼节点的索引父对象变形转换矩阵。

39. boneGetMorphIsDead <node>node <Integer>index <boolean>

当用于指定骨骼节点索引变形已经消失了，返回 True。

40. boneSetMorphSetDead <node>node <Integer>index <boolean>dead

设置指定骨骼节点索引变形的消失状态。

41. boneGetMorphNumPoints <node>node <Integer>index <boolean>

返回指定骨骼节点索引变形点数量。

42. boneSetMorphNumPoints <node>node <Integer>index <Integer>numPoints

设置指定骨骼节点索引变形点数量。

43. boneGetMorphVertID <node>node \

<Integer>morphIndex <Integer>ithVertIndex <Integer>

返回指定骨骼节点索引变形顶点 ID 号码。

44. boneSetMorphVertID <node>node \  
     <Integer>morphIndex <Integer>ithVertIndex <Integer>index  
 设置指定骨骼节点索引变形顶点 ID 号码。

45. boneGetMorphVec <node>node \  
     <Integer>morphIndex <Integer>ithVertIndex <point3>  
 返回指定骨骼节点索引变形的向量值。

46. boneSetMorphVec <node>node\  
     <Integer>morphIndex <Integer>ithVertIndex <point3>vec  
 设置指定骨骼节点索引变形的向量值。

47. boneGetMorphVecInParentSpace <node>node \  
     <Integer>morphIndex <Integer>ithVertIndex <point3>  
 返回指定骨骼节点索引父对象位置的变形向量值。

48. boneSetMorphVecInParentSpace <node>node \  
     <Integer>morphIndex <Integer>ithVertIndex <point3>vec  
 设置指定骨骼节点索引父对象位置的变形向量值。

49. boneGetMorphBasePoint <node>node \  
     <Integer>morphIndex <Integer>ithVertIndex <point3>  
 返回指定骨骼节点索引变形的基点。

50. boneSetMorphBasePoint <node>node \  
     <Integer>morphIndex <Integer>ithVertIndex <point3>point  
 设置指定骨骼节点索引变形的基点。

51. boneGetMorphOwner <node>node \  
     <Integer>morphIndex <Integer>ithVertIndex <node>  
 返回指定骨骼节点索引变形的索引点。

52. boneSetMorphOwner <node>node \  
     <Integer>morphIndex <Integer>ithVertIndex <node>node  
 设置指定骨骼节点索引变形的索引点。

53. boneGetMorphFalloff <node>node <Integer>morphIndex <Integer>  
 返回指定骨骼节点索引变形的衰减值。

54. boneSetMorphFalloff <node>node \  
     <Integer>morphIndex <Integer>falloff  
 设置指定骨骼节点索引变形的衰减值。

55. getWeight <node>node <string>morphName <Float>  
 设置于指定骨骼节点索引变形的权重值。

56. boneGetMorphEnabled <node>node <Integer>morphIndex <boolean>  
 返回指定骨骼节点索引变形的可用状态。

57. boneSetMorphEnabled <node>node \  
     <Integer>morphIndex <boolean>enabled

设置指定骨骼节点索引变形的可用状态。

### 58. update()

对 Skin\_Morph Modifier 施加一次更新。

## 10.4.57 Skin\_Wrap: Modifier（蒙皮包裹修改器）

### 构造函数

`Skin_Wrap ...`

`MeshDeformPW ..`

### 属性

属性名称	数据类型	默认值	说明
<code>&lt;Skin_Wrap&gt;.meshList</code>	<code>ArrayParameter</code>	<code>#()</code>	存取、设置面片列表的对象阵列，相当于参数卷展栏里的对象列表
<code>&lt;Skin_Wrap&gt;.mesh</code>	<code>undefined</code>	<code>undefined</code>	
<code>&lt;Skin_Wrap&gt;.engine</code>	<code>Integer</code>	<code>1</code>	确定执行变形的引擎： 0: Face Deformation（面变形引擎，每个控制顶点被绑定到基础对象中的最接近面。面变形可以使用衰减，也可以通过将衰减设置为可能的最低值 0.001 来作为刚体变形） 1: Vertex Deformation（顶点变形引擎是一个权重引擎，其使用顶点接近度执行变形）
<code>&lt;Skin_Wrap&gt;.falloff</code>	<code>Float</code>	<code>1.0</code>	设置衰减值
<code>&lt;Skin_Wrap&gt;.Distance</code>	<code>Float</code>	<code>2</code>	以系统单位数确定控制对象中的控制顶点的影响距离
<code>&lt;Skin_Wrap&gt;.faceLimit</code> <code>&lt;Skin_Wrap&gt;.Face_Limit</code>	<code>Integer</code>	<code>3</code>	确定控制对象面或控制顶点的影响范围
<code>&lt;Skin_Wrap&gt;.blend</code> <code>&lt;Skin_Wrap&gt;.Blend_To_Base_Mesh</code>	<code>Boolean</code>	<code>False</code>	设置是否与基础网格混合
<code>&lt;Skin_Wrap&gt;.blendDistance</code> <code>&lt;Skin_Wrap&gt;.Blend_Distance</code>	<code>Float</code>	<code>5.0</code>	确定控制对象中的面和基础对象中的顶点之间的距离
<code>&lt;Skin_Wrap&gt;.threshold</code>	<code>Float</code>	<code>5.0</code>	设置阈值
<code>&lt;Skin_Wrap&gt;.weightAllVerts</code> <code>&lt;Skin_Wrap&gt;.Weight_All_Verts</code>	<code>Boolean</code>	<code>False</code>	设置是否强制所有基础对象点具有权重
<code>&lt;Skin_Wrap&gt;.showMirrorData</code> <code>&lt;Skin_Wrap&gt;.Show_Mirror_Data</code>	<code>Boolean</code>	<code>False</code>	当设置为 On 时，启用镜像平面 gizmo 的显示
<code>&lt;Skin_Wrap&gt;.mirrorPlane</code> <code>&lt;Skin_Wrap&gt;.Mirror_Plane</code>	<code>Integer</code>	<code>0</code>	设置镜像轴：0: X; 1: Y; 2: Z
<code>&lt;Skin_Wrap&gt;.mirrorOffset</code> <code>&lt;Skin_Wrap&gt;.Mirror_Offset</code>	<code>Float</code>	<code>0.0</code>	设置镜像偏移值

(续表)

属性名称	数据类型	默认值	说明
<Skin_Wrap>.mirrorThreshold <Skin_Wrap>.Mirror_Threshold	Float	0.5	设置镜像阈值
<Skin_Wrap>.showLoops <Skin_Wrap>.Show_Loops	Boolean	True	当设置为 On 时, 将选定控制顶点的影响体积显示为红色循环
<Skin_Wrap>.showAxis <Skin_Wrap>.Show_Axis	Boolean	True	设置是否显示选定控制顶点的轴三轴架
<Skin_Wrap>.showFaceLimit <Skin_Wrap>.Show_Face_Limit	Boolean	True	设置是否显示面限制
<Skin_Wrap>.showUnassignedVerts <Skin_Wrap>.Show_Unassigned_Verts	Boolean	False	设置 Display Unassigned Points 的状态
<Skin_Wrap>.showVerts <Skin_Wrap>.Show_Verts	Boolean	True	设置 Display Control Verts 控件的状态

### 方法

#### 1. reset()

重设控制点的强度值和放缩值, 相当于按下 Reset 按钮。

#### 2. setLocalStr <Float str>

设置控制点的局部强度值。

#### 3. setLocalScale <Integer axis> <Float scale>

设置控制点的沿指定轴的局部放缩值, 允许的轴向值为:

- ◆ 1: X
- ◆ 2: Y
- ◆ 3: Z

#### 4. ConvertToSkin <Boolean silent>

相当于按下 Convert To Skin 按钮。

#### 5. mirrorSelectedControlPoints()

相当于按下 Mirror Selected 按钮。

#### 6. bakeControlPoint()

相当于按下 Bake Control Verts 按钮。

#### 7. retrieveControlPoint()

相当于按下 Retrieve Control Verts 按钮。

#### 8. getSelectedVertices <Integer wrapNodeIndex>

返回指定索引包裹选择点的 Bitarray 数组。

#### 9. setSelectVertices <Integer wrapNodeIndex> <bitArray sel> <Boolean update>

指定索引包裹选择点的 Bitarray 数组。

#### 10. getNumberOfControlPoints <Integer wrapNodeIndex>

返回指定索引包裹的控制点数量。

11. `getControlPointScale <Integer wrapNodeIndex> <Integer index>`  
返回一个 point3 值，表示指定索引包裹里指定控制点的缩放值。
12. `setControlPointScale <Integer wrapNodeIndex> <Integer index> <point3 scale>`  
设置指定索引包裹里指定控制点的缩放值。
13. `getControlPointStr <Integer wrapNodeIndex> < Integer index>`  
返回一个 Float 值，表示指定索引包裹里指定控制点的强度值。
14. `setControlPointStr <Integer wrapNodeIndex> < Integer index> <Float str>`  
设置指定索引包裹里指定控制点的强度值。
15. `getControlPointInitialTM <Integer wrapNodeIndex> < Integer index> <matrix3>`  
返回指定索引包裹里指定控制点初始转换矩阵。
16. `getControlPointCurrentTM <Integer wrapNodeIndex> < Integer index> <matrix3>`  
返回指定索引包裹里指定控制点的当前转换矩阵。
17. `getControlPointDist <Integer wrapNodeIndex> < Integer index>`  
返回指定索引包裹里指定控制点的距离。
18. `getControlPointXVert <Integer wrapNodeIndex> < Integer index>`  
返回指定索引包裹里指定控制点的转换顶点。

## 10.4.58 Skin\_Wrap\_Patch: Modifier（蒙皮包裹面片修改器）

### 构造函数

`Skin_Wrap_Patch ...`

### 属性

属性名称	数据类型	默认值	说明
<code>&lt;Skin_Wrap_Patch&gt;.patch</code>	<code>undefined</code>	<code>undefined</code>	指定使网格对象变形的面片对象
<code>&lt;Skin_Wrap_Patch&gt;.sampleRate</code>	<code>Integer</code>	<code>100</code>	设置采样率
<code>&lt;Skin_Wrap_Patch&gt;.Sample_Rate</code>			

### 方法

`resample()`

相当于按下 Resample 按钮。

## 10.4.59 SliceModifier: Modifier（切片修改器）

### 构造函数

`SliceModifier ...`

### 属性

属性名称	数据类型	默认值	说明
<sliceModifier>.Slice_Type	Integer	0	设置切片类型: 0: Refine Mesh (在剪切平面与对象的相交处加入新的节点和边, 被剪切的面被分割为两个新面) 1: Split Mesh (将剪切平面与对象的相交处的节点复制一份, 这两分节点分属于两个分离的对象) 2: Remove Top (删除剪切平面上部的对象) 3: Remove Bottom (删除剪切平面下部的对象)
<sliceModifier>.Faces__Polygons_Toggle	Integer	1	设置切片操作作用的方式: 0: 用于三角面; 1: 作用于多边形面
<sliceModifier>.slice_plane	SubAnim		在次级结构层级, 可以变换切片平面, 还可以将变换过程记录为动画。但是放缩变换操作不会对切片平面产生影响, 切片平面在场景空间中是无限延伸的, 可以通过对对象或次级结构对象的选择集, 约束切片平面的作用范围
<sliceModifier>.Slice_Plane>.position	Point3	[0,0,0]	设置切片平面位置。可动画
<sliceModifier>.Slice_Plane>.rotation	Quat	默认值: (quat 0 0 0 1)	设置切片平面转角。可动画
<sliceModifier>.Slice_Plane>.scale	Point3	[1,1,1]	设置切片平面缩放因子。可动画

#### 10.4.60 smooth: Modifier (平滑修改器)

##### 构造函数

smooth ...

##### 属性

属性名称	数据类型	默认值	说明
<smooth>.autosmooth	Boolean	False	当设置为 On 时, 为对象表面施加自动光滑操作。自动光滑操作依据相邻面的面法线之间的角度, 如果该角度小于阈值, 则该相邻面被指定为同一个光滑组
<smooth>.preventIndirect	Boolean	False	当设置为 On 时, 可以避免由于间接光滑的指定, 使自动光滑过程产生漏洞
<smooth>.threshold	Float	30.0	指定相邻面法线之间角度的阈值, 如果面法线夹角小于阈值, 则这两个面被指定为同一个平滑组。可动画, Angle
<SmoothModifier>.smoothingBits	Integer	0	为选择的面指定平滑组

### 10.4.61 Spherify: Modifier（球形化修改器）

#### 构造函数

Spherify ...

#### 属性

属性名称	数据类型	默认值	说明
<Spherify>.percent	Float	100.0	设置球形扭曲的百分比。可动画

### 10.4.62 Spline\_IK\_Control: Modifier（样条线 IK 控制修改器）

#### 构造函数

Spline\_IK\_Control ...

#### 属性

属性名称	数据类型	默认值	说明
<Spline_IK_Control>.box	Boolean	True	设置是否显示小长方体辅助对象。可动画
<Spline_IK_Control>.helper_size	Float	20.0	设置辅助对象大小。可动画
<Spline_IK_Control>.constantscreensize	Boolean	False	设置是否保持辅助对象大小恒定，而不管将视口放大或缩小多少
<Spline_IK_Control>.drawontop <Spline_IK_Control>.Draw_On_Top	Boolean	True	设置是否在场景中其他所有对象上显示辅助对象
<Spline_IK_Control>.helper_axistripod <Spline_IK_Control>.Axis_Tripod	Boolean	False	设置是否在每一个结上显示三轴架
<Spline_IK_Control>.helper_centermarker <Spline_IK_Control>.Center_Marker	Boolean	False	设置是否在每一个结上显示小 X 标记
<Spline_IK_Control>.helper_cross <Spline_IK_Control>.Cross	Boolean	False	设置是否显示交叉标记
<Spline_IK_Control>.helper_list	Array-Parameter	#()	设置用于控制样条线的辅助对象列表
<Spline_IK_Control>.linkTypes <Spline_IK_Control>.Link_Types	Integer	0	设置链接类型： 0: Link All In Hierarchy (层次链接全部) 1: Link All to Root (链接全部到根) 2: No Linking (无链接)

#### 方法

1. <Integer>getHelperCount()

返回 Spline\_IK\_Control Modifier 里辅助对象数量。

2. <Integer>getKnotCount()

返回控制样条线的 knots 数量。

### 3. <boolean>linkToRoot()

设置修改器里 Link All To Root 模式。当设置成功后，返回 True。

### 4. <boolean>linkInHierarchy()

设置修改器里 Link All In Hierarchy 模式。当设置成功后，返回 True。

### 5. <boolean>noLinking()

设置修改器里 No Linking 模式。当设置成功后，返回 True。

### 6. <boolean>createHelper <Integer>knotCount

为指定数量的 knots 创建辅助对象。当设置成功后，返回 True。

## 10.4.63 SplineSelect: Modifier (样条线选择修改器)

### 构造函数

SplineSelect ...

### 属性

除通用属性外，SplineSelect 类没有额外的属性。

## 10.4.64 Squeeze: Modifier (挤压修改器)

### 构造函数

Squeeze ...

### 属性

属性名称	数据类型	默认值	说明
<Squeeze>.Bulge_Amount	Float	0.0	控制膨胀效果的大小。可动画
<Squeeze>.Bulge_Curvature	Float	2.0	指定膨胀端点的膨胀曲线曲率。可动画
<Squeeze>.Squeeze_Amount	Float	0.0	控制挤扁效果的大小。如果设置的数值大于 0，可以创建收腰挤扁的效果；如果设置的数值小于 0，可以创建膨胀的效果。可动画
<Squeeze>.Squeeze_Curvature	Float	2.0	指定挤扁端点的挤扁曲率，较低的数值创建尖锐的挤扁效果；较高的数值创建平滑渐变的挤扁效果。可动画
<Squeeze>.Limit_Squeeze_Effects	Integer	0	当设置为 On 时，使挤扁操作的限定效果更为有效：0: Off; 1: On
<Squeeze>.Squeeze_Lower_Limit	Float	-50.0	指定在局部坐标 Z 轴正方向的限定位置。可动画
<Squeeze>.Squeeze_Upper_Limit	Float	50.0	指定在局部坐标 Z 轴负方向的限定位置。可动画
<Squeeze>.Bias	Float	0.0	当保持对象的恒定体积时，指定膨胀或挤扁效果的相对偏移量。可动画，百分比

(续表)

属性名称	数据类型	默认值	说明
<Squeeze>.Volume	Float	0.0	指定对象在膨胀或挤扁编辑过程中体积的变化量。该参数可以增加或减少挤扁/膨胀的效果。可动画，百分比
<Squeeze>.center	Point3	[0,0,0]	在次级结构层级，可以变换和动画编辑器 gizmo 的中心对象。可动画
<Squeeze>.gizmo	SubAnim		在次级结构层级，可以变换和动画编辑器 gizmo
<Squeeze.Gizmo>.position	Point3	[0,0,0]	设置挤扁/膨胀编辑器 gizmo 的位置。可动画
<Squeeze.Gizmo>.rotation	Quat	默认值: (quat 0 0 0 1)	
		设置挤扁/膨胀编辑器 gizmo 的旋转角度。可动画	
<Squeeze.Gizmo>.scale	Point3	[1,1,1]	设置挤扁/膨胀编辑器 gizmo 的缩放因子。可动画

#### 10.4.65 STL\_Check: Modifier (STL 检查修改器)

##### 构造函数

STL\_Check ...

##### 属性

属性名称	数据类型	默认值	说明
<STL_Check>.Selection_Type	Integer	4	设置进行 STL 检查的错误类型: 0: Open Edge (检查对象表面是否存在开发边错误) 1: Double Face (检查对象表面同一位置是否存在重叠的面) 2: Spike (检查对象表面是否存在增生的面，增生面只有一条边与对象表面连接) 3: Multiple Edge (检查对象表面是否存在两个面共享多条边的错误); 4: Everything (检查对象表面是否存在所有以上的错误)
<STL_Check>.Select_Faces	Integer	1	设置选择类型: 0: Don't Select (不选择被检查出错误的次级结构对象) 1: Select Faces (选择被检查出错误的次级结构面) 2: Select Edges (选择被检查出错误的次级结构边)
<STL_Check>.Check_Now	Integer	0	设置是否执行 STL 检查。0: Off; 1: On
<STL_Check>.Change_MatID	Integer	1	当设置为 On 时，STL 检查通过指定一个唯一的材质 ID 码来标记错误的面。0: Off; 1: On
<STL_Check>.Material_ID	Integer	2	设置 STL 检查标记错误面的材质 ID 码

### 10.4.66 Stretch: Modifier (拉伸修改器)

#### 构造函数

Stretch ...

#### 属性

属性名称	数据类型	默认值	说明
<Stretch>.Stretch	Float	0.0	设置延展放缩的强度。可动画
<Stretch>.Amplify	Float	0.0	设置在次轴向上延展强度的倍增量，正值倍增延展放缩强度；负值减小延展放缩强度。可动画
<Stretch>.axis	Integer	2	指定延展变形依据的轴向：0: X; 1: Y; 2: Z
<Stretch>.limit	Boolean	False	设置是否启用限定效果
<Stretch>.from	Float	0.0	指定延展操作的上限范围。可动画
<Stretch>.to	Float	0.0	指定延展操作的下限范围。可动画
<Stretch>.center	Point3	[0,0,0]	在次级结构层级，可以变换和动画延展编辑器 gizmo 的中心对象。可动画
<Stretch>.gizmo	SubAnim		在次级结构层级，可以变换和动画延展编辑器 gizmo
<Stretch.Gizmo>.position	Point3	[0,0,0]	设置延展编辑器 gizmo 的位置。可动画
<Stretch.Gizmo>.rotation	Quat	默认值: (quat 0 0 0 1)	
		设置延展编辑器 gizmo 的旋转角度。可动画	
<Stretch.Gizmo>.scale	Point3	[1,1,1]	设置延展编辑器 gizmo 的放缩因子。可动画

### 10.4.67 Subdivide: Modifier (细分修改器)

细分修改器提供了用于光能传递处理创建网格的一种算法。

#### 构造函数

Subdivide ...

#### 属性

属性名称	数据类型	默认值	说明
<subdivide>.manualUpdate <subdivide>.Manual_Update	Integer	1	设置更新类型： 0: Manual (手动) 1: Automatic (自动) 2: Render (渲染)
<subdivide>.showAllTriangles <subdivide>.Show_All_Triangles	Boolean	True	控制是否显示所有的三角形，或只是显示更改了面属性的边。如果三角形显示得比较杂乱，可以用此选项来减少场景中三角形的显示
<subdivide>.size	Float	39.3701	控制细分网格中三角形的大小。可动画

#### 10.4.68 Substitute: Modifier（替代修改器）

##### 构造函数

Substitute...

SubstituteMod...

##### 属性

属性名称	数据类型	默认值
<Substitute>.DisplayInViewport	Boolean	True
<Substitute>.DisplayInRender	Boolean	True
<Substitute>.ObjectName	String	“”
<Substitute>.SceneNodeName	Node	undefined
<Substitute>.ObjectReference	Integer	2
<Substitute>.SelectFrom	String	“”
<Substitute>.MoveCopyInstance	Boolean	False
<Substitute>.RetainPosition	Boolean	True
<Substitute>.RetainLocalRotation	Boolean	True
<Substitute>.RetainLocalScale	Boolean	True

#### 10.4.69 Surface: Modifier（曲面修改器）

##### 构造函数

Surface ...

##### 属性

属性名称	数据类型	默认值	说明
<surface>.threshold	Float	1.0	设置焊接样条对象的顶点时的距离阈值，可动画
<surface>.steps	Integer	5	设置每两顶点的计算步数，步数设置得越高，得到的曲线越光滑。可动画

**注意** 反转法线、移去内部面片和仅使用选择的线段等属性都不被MAXScript支持。

#### 10.4.70 SurfDeform: Modifier（曲面变形修改器）

##### 构造函数

SurfDeform ...

### 属性

属性名称	数据类型	默认值	说明
<SurfDeform>.surface	Node	undefined	指定变形曲面
<SurfDeform>.U_Percent	Float	50.0	指定对象沿曲面变形 gizmo 的 U 轴方向移动的百分比。移动的距离取决于曲面 gizmo 在 U 轴方向的长度。如果设置为 50，将对象放置在曲面 U 轴的中心位置；如果设置为 0，将曲面对象放置在曲面 U 轴方向的左边缘。可动画，百分比
<SurfDeform>.U_Stretch	Float	1.0	沿面片曲面 gizmo 的 U 轴方向放缩对象。可动画
<SurfDeform>.V_Percent	Float	50.0	指定对象沿曲面变形 gizmo 的 V 轴方向移动的百分比。移动的距离取决于曲面 gizmo 在 V 轴方向的长度。如果设置为 50，将对象放置在曲面 V 轴的中心位置；如果设置为 0，将面片对象放置在曲面 V 轴方向的底部。可动画，百分比
<SurfDeform>.V_Stretch	Float	1.0	沿面片曲面 gizmo 的 V 轴方向放缩对象。可动画
<SurfDeform>.rotation	Float	0.0	旋转变形对象，使其与基准面呈一个平面。可动画，Angle
<SurfDeform>.Plane_to_Patch_Deform	Integer	0	指定对象进行曲面变形所依据变形 gizmo 的平面：0: XY; 1: YZ; 2: ZX
<SurfDeform>.Flip_deformation_axis	Integer	0	设置是否反转曲面变形 gizmo 的方向：0: 不反转；1: 反转
<SurfDeform>.gizmo	SubAnim		在次级结构层级，可以变换和动画曲面变形编辑器 gizmo 的对象
<SurfDeform.Gizmo>.position	Point3	[0,0,0]	设置曲面变形编辑器 gizmo 的位置。可动画
<SurfDeform.Gizmo>.rotation	Quat	默认值: (quat 0 0 0 1)	
		设置曲面变形编辑器 gizmo 的旋转角度。可动画	
<SurfDeform.Gizmo>.scale	Point3	[1,1,1]	设置曲面变形编辑器 gizmo 的放缩因子。可动画

### 10.4.71 Symmetry:Modifier (对称修改器)

可以对任意几何体应用对称修改器，并且可以通过设置修改器 gizmo 的动画来对镜像或切片设置动画。

#### 构造函数

Symmetry ...

#### 属性

属性名称	数据类型	默认值	说明
<symmetry>.axis	Integer	0	指定执行对称所围绕的轴：0: X; 1: Y; 2: Z;
<symmetry>.flip	Boolean	False	设置是否使用反转效果
<symmetry>.slice	Integer	1	当设置为 On 时，启用沿镜像轴切片，使镜像 gizmo 作为一个切片平面

(续表)

属性名称	数据类型	默认值	说明
<symmetry>.weld	Integer	1	当设置为 On 时，启用焊接缝来确保沿镜像轴的顶点在阈值以内时会自动焊接
<symmetry>.threshold	Float	0.1	设置阈值。可动画
<symmetry>.mirror	SubAnim	SubAnim -m:Mir -ror	控制镜像 gizmo 的转换
<symmetry.mirror>.position	Point3	[0,0,0]	设置镜像 gizmo 的位置
<symmetry.mirror>.rotation	Quat	(quat 0 0 0 1)	设置镜像 gizmo 的旋转角度。可动画
<symmetry.mirror>.scale	Point3	[1,1,1]	设置镜像 gizmo 的缩放倍数。可动画
<symmetry.mirror>.value	Matrix3	默认值: (matrix3 [1,0,0] [0,1,0] [0,0,1] [0,0,0])	
		设置镜像 gizmo 的完整变换矩阵值。可动画	

#### 10.4.72 Taper: Modifier (锥化修改器)

##### 构造函数

Taper ...

##### 属性

属性名称	数据类型	默认值	说明
<Taper>.amount	Float	1.0	指定端面的放缩程度，这是一个相对值，最大可设定为 10。可动画
<Taper>.curve <Taper>.curvature	Float	0.0	为锥化 gizmo 的侧面指定一个曲率，该数值影响锥化变形的最终形态，正值产生向外凸出的侧面；负值产生一个向内凹进的侧面；如果设定为 0，侧面是平直的。可动画
<Taper>.primaryaxis	Integer	2	指定锥化操作所依据的坐标轴：0: X; 1: Y; 2: Z
<Taper>.effectaxis	Integer	2	指定影响效果的轴向：0: Z; 1: Y; 2: ZY
<Taper>.symmetry	Boolean	False	设置锥化的效果是否对称。可动画
<Taper>.limit	Boolean	False	设置是否启用限定效果
<Taper>.upperlimit <Taper>.Upper_Limit	Float	0.0	设置锥化操作的上限范围。可动画
<Taper>.lowerlimit <Taper>.Lower_Limit	Float	0.0	设置锥化操作的下限范围。可动画
<Taper>.center	Point3	[0,0,0]	在次级结构层级，可以变换和动画锥化编辑器 gizmo 的中心对象。可动画
<Taper>.gizmo	SubAnim		在次级结构层级，可以变换和动画锥化编辑器 gizmo
<Taper.Gizmo>.position	Point3	[0,0,0]	设置锥化编辑器 gizmo 的位置。可动画

(续表)

属性名称	数据类型	默认值	说明
<Taper.Gizmo>.rotation	Quat	(quat 0 0 0 1)	设置锥化编辑器 gizmo 的旋转角度。可动画
<Taper.Gizmo>.scale	Point3	[1,1,1]	设置锥化编辑器 gizmo 的放缩因子。可动画

#### 10.4.73 Tessellate: Modifier (细化修改器)

##### 构造函数

Tessellate ...

##### 属性

属性名称	数据类型	默认值	说明
<Tessellate>.tension	Float	2500.0	指定细化后的面是否平坦，正值使细化后的表面向外凸起；负值使细化后的表面向内凹陷；取值为 0 时，细化后的表面保持平坦。可动画，百分比
<tessellate>.faceType	Integer	0	指定细化操作的方式： 0: Operate on Faces (作为三角面集来处理) 1: Operate on Polygons (细分多边形面)
<tessellate>.type	Integer	0	指定细化类型： 0: Edge (从多边形或曲面的中心到每条边的中点进行细分。 应用于三角面时，也会将与选中曲面共享边的非选中曲面进行细分) 1: Face-Center (对从中心到顶点角的曲面进行细分)
<tessellate>.iterations	Integer	0	指定应用细化的次数
<tessellate>.updateWhen	Integer	0	设置更新的选项： 0: Always (始终，无论何时改变了基本几何体都对细化进行更新) 1: When Rendering (渲染时，仅在对象渲染后进行细化的更新) 2: Manually (手动，仅在用户单击更新时对细化进行更新)

**注意** 属性.tension 的值是 3ds max 的界面里显示的 tension 值乘以 100。

#### 10.4.74 Trim\_Extend: Modifier (修剪/延伸修改器)

##### 构造函数

Trim\_Extend ...

##### 属性

除通用属性外，Trim\_Extend 没有额外的属性。

### 10.4.75 TurboSmooth: Modifier(涡轮平滑修改器)

#### 构造函数

TurboSmooth ...

#### 属性

属性名称	数据类型	默认值	说明
<TurboSmooth>.explicitNormals <TurboSmooth>.Explicit_Normals	Boolean	False	当设置为 On 时, 允许涡轮平滑修改器为输出计算法线, 此方法要比 3ds max 中网格对象平滑组中用于计算法线的标准方法迅速
<TurboSmooth>.isolineDisplay <TurboSmooth>.Isoline_Display	Boolean	False	当设置为 On 时, 仅显示等值线: 对象在平滑之前的原始边。使用此项的好处是减少混乱的显示; 否则, 会显示所有通过涡轮平滑添加的曲面, 因此更高的迭代次数会产生更多数量的线条
<TurboSmooth>.iterations	Integer	1	设置网格细分的次数。增加该值时, 每次新的迭代会通过在迭代之前对顶点、边和曲面创建平滑差补顶点来细分网格, 修改器会细分曲面来使用这些新的顶点。可动画
<TurboSmooth>.renderIterations	Integer	0	设置渲染迭代次数。可动画
<TurboSmooth>.sepByMats <TurboSmooth>.Separate_By_Materials	Boolean	False	设置是否按材质分隔。当设置为 On 时, 防止在不共享材质 ID 的曲面之间的边创建新曲面
<TurboSmooth>.sepBySmGroups <TurboSmooth>.Separate_By_Smoothing_Group	Boolean	False	设置是否按平滑组分隔。当设置为 On 时, 防止在不共享至少一个平滑组的曲面之间的边上创建新曲面
<TurboSmooth>.smoothResult <TurboSmooth>.Smooth_Output	Boolean	True	当设置为 On 时, 对所有曲面应用相同的平滑组
<TurboSmooth>.update <TurboSmooth>.Update_Options	Integer	0	设置更新的方式: 0: Always (无论何时改变任何涡轮平滑设置都自动更新对象) 1: When Rendering (仅在渲染时更新视口中对象的显示) 2: Manually (手动更新)
<TurboSmooth>.useRenderIterations <TurboSmooth>.Use_Render_Iterations	Boolean	False	当设置为 On 时, 允许在渲染时选择一个不同数量的平滑迭代次数应用于对象。启用渲染迭代次数

#### 方法

ForceUpdate()

对修改器进行了一次更新，相当于在修改器用户界面上按下更新的按钮。

#### 10.4.76 Turn\_to\_Mesh: Modifier (转化为网格修改器)

##### 构造函数

Turn\_to\_Mesh ...

##### 属性

属性名称	数据类型	默认值	说明
<Turn_to_Mesh>.useInvisibleEdges <Turn_to_Mesh>.Use_Invisible_Edges	Boolean	True	当设置为 On 时，使用不可见边来表示多边形。否则，会产生一个完整的三角网格，其所有边都可见。可动画
<Turn_to_Mesh>.selectionConversion <Turn_to_Mesh>.Selection_Conversion	Integer	0	
<Turn_to_Mesh>.useSoftSelection <Turn_to_Mesh>.Use_Soft_Selection	Boolean	True	当设置为 On 时，对围绕变形选中子对象的未选中顶点应用一个变形样条曲线。这会通过一个球体围绕变形的影响来提供一个类似磁体的效果。例如，如果在选中可编辑多边形上的顶点时启用使用软选择，并且应用旋转至网格同时启用包括软选择，那么同样的软选择会应用于网格顶点
<Turn_to_Mesh>.selectionLevel <Turn_to_Mesh>.Selection_Level	Integer	0	设置子对象选择层级来传递堆栈的其余部分： 0: From Pipeline (使用输入对象所使用的任何对象来传递堆栈。例如，如果创建一个长方体，将其在面片模式转换为可编辑面片，并对它应用旋转至网格修改器，3ds max 会将面片模式中的子对象选择向上传输进堆栈，旋转至网格修改器会考虑子对象面片的选择并选中来源于面片选择的网格面) 1: Object (使用对象作为选择层级来传递堆栈的其余部分) 2: Edge (使用边作为子对象选择层级来传递堆栈其余部分) 3: Vertex (使用顶点作为子对象选择层级来传递堆栈的其余部分) 4: Face (使用面作为子对象选择层级来传递堆栈的其余部分)

#### 10.4.77 Turn\_to\_Patch: Modifier (转化为面片修改器)

旋转至面片修改器允许在修改器堆栈中应用对象转化。使用旋转至面片修改器可以微调转化过程比如将四边形转化为四边形面片。

**注意** 从一个对象类型转换为另一种类型会在修改器堆栈中启用一个完全的缓存。在场景中存在大型对象时，此操作会占用很多空间。例如，由网格开始的对象转换为面片然后转换回网格所占用的空间是仅使用普通修改器比如应用弯曲或 UVW 贴图的网格所占用空间的三倍。

**构造函数**

Turn\_to\_Patch ...

ConvertToPatch ...

**属性**

属性名称	数据类型	默认值	说明
<Turn_to_Patch>.quadsToQuads <Turn_to_Patch>.Quads_to_Quad_Patches	Boolean	True	当设置为 On 时，将四边形曲面转换为网格或将多边形转换为四边形面片。可动画
<Turn_to_Patch>.selectionConversion <Turn_to_Patch>.Selection_Conversion	Integer	0	
<Turn_to_Patch>.useSoftSelection <Turn_to_Patch>.Use_Soft_Selection	Integer	1	当设置为 On 时，对围绕变形选中子对象的未选中顶点应用一个变形样条曲线
<Turn_to_Patch>.selectionLevel <Turn_to_Patch>.Selection_Level	Integer	0	设置子对象选择层级来传递堆栈的其余部分： 0: From Pipeline (使用相当于输入对象所使用的任何对象) 1: Object (使用对象作为选择层级来传递堆栈的其余部分) 2: Edge (使用边作为子对象选择层级来传递堆栈的其余部分) 3: Vertex (使用顶点作为子对象选择层级来传递堆栈的其余部分) 4: Patch (使用曲面作为子对象选择层级来传递堆栈的其余部分)

**10.4.78 Turn\_to\_Poly: Modifier (转化为多边形修改器)****构造函数**

Turn\_to\_Poly ...

**属性**

属性名称	数据类型	默认值	说明
<Turn_to_Poly>.keepConvex <Turn_to_Poly>.Keep_Convex	Boolean	False	设置是否保持多边形为凸面。如果多边形不为凸面则对边不会相结合。凸面意味着可以用一条线连接多边形的任意两点而不会超出多边形以外。如果可以以一条线连接多边形两点而这条线位于多边形外部，则此多边形不是凸面。对非凸面多边形的操作会出现问题，这包括改变输入对象几何体会导致与转换至多边形结果不同的拓扑。可动画

(续表)

属性名称	数据类型	默认值	说明
<Turn_to_Poly>.limitPolySize <Turn_to_Poly>.Limit_Polygon_Size	Boolean	False	设置是否限制多边形大小。对多边形限制侧面的数量使曲面更易于定义。可动画
<Turn_to_Poly>.maxPolySize <Turn_to_Poly>.Max_Polygon_Size	Integer	4	设置多边形侧面的最大数量。可动画
<Turn_to_Poly>.requirePlanar <Turn_to_Poly>.Require_Planar_Polygons	Boolean	False	创建包含平面的多边形。如果边上有一超过阈值的锐角，此操作不会将面横跨该边合成。可动画
<Turn_to_Poly>.planarThresh <Turn_to_Poly>.Planar_Threshold	Float	15.0	控制多边形平面之间的角度阈值。可动画
<Turn_to_Poly>.removeMidEdgeVertices <Turn_to_Poly>.Remove_Mid_Edge_Vertices	Boolean	True	
<Turn_to_Poly>.selectionConversion <Turn_to_Poly>.Selection_Conversion	Integer	0	
<Turn_to_Poly>.useSoftSelection <Turn_to_Poly>.Use_Soft_Selection	Boolean	True	影响子对象的移动、旋转和缩放的操作功能。当设置为 On 时，对围绕变形选中子对象的未选中顶点应用一个变形样条曲线
<Turn_to_Poly>.selectionLevel <Turn_to_Poly>.Selection_Level	Integer	0	设置子对象选择层级来传递堆栈的其余部分： 0: From Pipeline (使用相当于输入对象所使用的任何对象，面片层级变为面层级等) 1: Object (使用对象作为选择层级来传递堆栈的其余部分) 2: Edge (使用边作为子对象选择层级来传递堆栈的其余部分) 3: Vertex (使用顶点作为子对象选择层级来传递堆栈的其余部分) 4: Face (使用面作为子对象选择层级来传递堆栈的其余部分)

#### 10.4.79 Twist: Modifier (扭曲修改器)

##### 构造函数

Twist ...

##### 属性

属性名称	数据类型	默认值	说明
<Twist>.angle	Float	0.0	指定扭曲变形的角度。可动画
<Twist>.bias	Integer	0	指定扭曲变形效果在对象表面上向或向下的偏向数值。负值，扭曲效果靠近变换 gizmo 的中心；正值，扭曲效果沿扭曲轴向远离中心；如果该值为 0，扭曲效果在扭曲轴向上是一致的。可动画
<Twist>.axis	Integer	2	指定扭曲变形依据的轴向：0: X; 1: Y; 2: Z
<Twist>.limit	Boolean	False	设置是否启用限定效果
<Twist>.upperlimit <Twist>.Upper_Limit	Float	0.0	设置扭曲变形操作的上限范围。可动画
<Twist>.lowerlimit <Twist>.Lower_Limit	Float	0.0	设置扭曲变形操作的下限范围。可动画
<Twist>.center	Point3	[0,0,0]	在次级结构层级，可以变换和动画扭曲变形编辑器 gizmo 的中心对象。可动画
<Twist>.gizmo	SubAnim		在次级结构层级，可以变换和动画扭曲变形编辑器 gizmo
<Twist.Gizmo>.position	Point3	[0,0,0]	设置扭曲变形编辑器 gizmo 的位置。可动画
<Twist.Gizmo>.rotation	Quat	默认值：(quat 0 0 0 1)	设置扭曲变形编辑器 gizmo 的旋转角度。可动画
<Twist.Gizmo>.scale	Point3	[1,1,1]	设置扭曲变形编辑器 gizmo 的放缩因子。可动画

#### 10.4.80 Unwrap\_UVW: Modifier (展开 UVW 修改器)

##### 构造函数

Unwrap\_UVW ...

UVWunwrap ...

##### 属性

除通用属性外，Unwrap\_UVW 没有额外的属性。

##### 方法

下面的方法需要在修改器面板活动时，指定 Unwrap UVW Modifier 在 Modifier 堆栈里被选择的情况下才能使用。

###### 1. PlanarMap()

相当于按下界面里 Planar Map 按钮。

###### 2. Save()

相当于按下界面里 Save 按钮。

###### 3. Load()

相当于按下界面里 Load 按钮。

### 4. Reset()

相当于按下界面里 Reset 按钮。

### 5. Edit()

相当于按下界面里 Edit 按钮。

### 6. SetMapChannel <channel>

为界面里 Map Channel 控件设置通道值。

### 7. GetMapChannel()

返回界面里 Map Channel 控件的值。

### 8. SetVC <VertexColor>

开关界面里 Vertex Color Channel 按钮。

### 9. GetVC()

返回界面里 Vertex Color Channel 按钮的值。

### 10. ProjectionType <Proj>

为 Sub Objects Params 按钮设置贴图的投影类型：

- ◆ 1 X 轴对齐
- ◆ 2 Y 轴对齐
- ◆ 3 Z 轴对齐
- ◆ 4 法线对齐

### 11. GetProjectionType()

返回 Sub Objects Params 按钮的值。

- ◆ 1 X 轴对齐
- ◆ 2 Y 轴对齐
- ◆ 3 Z 轴对齐
- ◆ 4 法线对齐

### 12. Move()

相当于按下 Edit UVWs 浮动窗口的 Move 按钮。

### 13. MoveH()

相当于按下 Edit UVWs 浮动窗口的 Move Horizontal 按钮。

### 14. MoveV()

相当于按下 Edit UVWs 浮动窗口的 Move Vertical 按钮。

### 15. Rotate()

相当于按下 Edit UVWs 浮动窗口的 Rotate 按钮。

### 16. Scale()

相当于按下 Edit UVWs 浮动窗口的 Scale 按钮。

### 17. ScaleH()

相当于按下 Edit UVWs 浮动窗口的 Scale Horizontal 按钮。

### 18. ScaleV()

相当于按下Edit UVWs浮动窗口的Scale Vertical按钮。

#### 19. MirrorH()

相当于按下Edit UVWs浮动窗口的Mirror Horizontal按钮。

#### 20. MirrorV()

相当于按下Edit UVWs浮动窗口的Mirror Vertical按钮。

#### 21. ExpandSelection()

相当于按下Edit UVWs浮动窗口的Expand Selection按钮。

#### 22. ContractSelection()

相当于按下Edit UVWs浮动窗口的Contract Selection按钮。

#### 23. SetFalloffType <falloff>

为Edit UVWs浮动窗口设置FallOff类型:

- ◆ 1 线性衰减 (Linear falloff)
- ◆ 2 正弦衰减 (Sinual falloff)
- ◆ 3 快速衰减 (Fast falloff)
- ◆ 4 缓慢衰减 (Slow falloff)

#### 24. GetFalloffType()

返回Edit UVWs浮动窗口设置FallOff类型值:

- ◆ 1 线性衰减 (Linear falloff)
- ◆ 2 正弦衰减 (Sinual falloff)
- ◆ 3 快速衰减 (Fast falloff)
- ◆ 4 缓慢衰减 (Slow falloff)

#### 25. SetFalloffSpace <space>

为Edit UVWs浮动窗口设置FallOff计算空间:

- ◆ 1 XY (对象的Local坐标系)
- ◆ 2 UV (对象的UVW坐标系)

#### 26. GetFalloffSpace()

返回Edit UVWs浮动窗口的FallOff计算空间设置。

#### 27. SetFalloffDist <dist>

为Edit UVWs浮动窗口设置FallOff距离。

#### 28. GetFalloffDist()

返回Edit UVWs浮动窗口的FallOff距离。

#### 29. BreakSelected()

相当于按下Edit UVWs浮动窗口的Break Selected按钮。

#### 30. Weld()

相当于按下Edit UVWs浮动窗口的Target Weld按钮。

#### 31. WeldSelected()

相当于按下 Edit UVWs 浮动窗口的 Weld Selected 按钮。

### 32. Updatemap()

相当于按下 Edit UVWs 浮动窗口的 Update Map 按钮。

### 33. Displaymap <DisplayMap>

开关 Edit UVWs 浮动窗口的 Display Map 按钮。

### 34. IsMapDisplayed()

返回 Edit UVWs 浮动窗口的 Display Map 按钮的当前状态。

### 35. SetUVSpace <UVspace>

为 Edit UVWs 浮动窗口设置 UV Space fly out。

参数 UVSpace 指定了用户要在哪一空间观察纹理顶点：

- ◆ 1 UV
- ◆ 2 VW
- ◆ 3 UW

### 36. GetUVSpace()

返回 Edit UVWs 浮动窗口的 UV Space fly out 设置：

- ◆ 1 UV
- ◆ 2 VW
- ◆ 3 UW

### 37. Options()

相当于按下 Edit UVWs 浮动窗口的 Options 按钮。

### 38. Lock()

开关 Edit UVWs 浮动窗口的 Lock Selected Vertices 按钮。

### 39. Hide()

相当于按下 Edit UVWs 浮动窗口的 Hide 按钮。

### 40. Unhide()

相当于按下 Edit UVWs 浮动窗口的 Unhide 按钮。

### 41. Freeze()

相当于按下 Edit UVWs 浮动窗口的 Freeze 按钮。

### 42. Thaw()

相当于按下 Edit UVWs 浮动窗口的 Unfreeze 按钮。

### 43. FilterSelected()

开关 Edit UVWs 浮动窗口的 Filter Selected Faces 按钮。

### 44. Pan()

相当于按下 Edit UVWs 浮动窗口的 Pan 按钮。

### 45. Zoom()

相当于按下 Edit UVWs 浮动窗口的 Zoom 按钮。

### 46. ZoomRegion()

相当于按下 Edit UVWs 浮动窗口的 Zoom Region 按钮。

47. Fit()

相当于按下 Edit UVWs 浮动窗口的 Fit 按钮。

48. FitSelected()

相当于按下 Edit UVWs 浮动窗口的 Fit Selected 按钮。

49. Snap()

相当于按下 Edit UVWs 浮动窗口的 Snap 按钮。

50. GetCurrentMap()

返回 Edit UVWs 浮动窗口里当前显示的贴图在 MAP 下拉列表里的序号。

51. SetCurrentMap <map>

改变 Edit UVWs 浮动窗口里显示的贴图。参数<map>表示指定贴图在 MAP 下拉列表里的序号。

52. NumberMaps()

返回 Edit UVWs 浮动窗口里 MAP 下拉列表里的贴图数量。

53. GetLineColor()

返回一个 Point3 值，表示用来连接纹理顶点的边线的颜色。

54. SetLineColor <color>

设置用来连接纹理顶点的边线的颜色。参数<color>为一个 Point3 值。

55. GetSelColor()

返回一个 Point3 值，表示当纹理顶点被选择时，用什么颜色显示。

56. SetSelColor <color>

设置当纹理顶点被选择时，用什么颜色显示。参数<color>为一个 Point3 值。

57. GetRenderWidth <dist>

返回 Edit UVWs 浮动窗口 BitMap Options 栏里 Width 控件的值，表示渲染 2d/3d 纹理时，生成的位图文件 (\*.bmp) 宽度，以像素为单位。

58. SetRenderWidth <width>

设置渲染 2d/3d 纹理时，生成的位图文件 (\*.bmp) 宽度，以像素为单位。

59. GetRenderHeight()

返回 Edit UVWs 浮动窗口 BitMap Options 栏里 Width 控件的值，表示渲染 2d/3d 纹理时，生成的位图文件 (\*.bmp) 高度，以像素为单位。

60. SetRenderHeight <height>

设置渲染 2d/3d 纹理时，生成的位图文件 (\*.bmp) 高度，以像素为单位。

61. GetUseBitmapRes()

返回一个 Boolean 值，表示 Edit UVWs 浮动窗口 BitMap Options 栏里 Use Bitmap Resolution 控件的状态，如果返回 False，位图采用系统变量 RenderWidth 和 RenderHeight 的值。

62. SetUseBitmapRes <useBitmapRes>

设置 Use Bitmap Resolution 的状态。

## 63. GetWeldThresold()

返回 Edit UVWs 浮动窗口里的焊接阈值。

## 64. SetWeldThreshold &lt;threshold&gt;

设置焊接阈值。

## 65. GetConstantUpdate()

返回一个 Boolean 值, 表示 Edit UVWs 浮动窗口 ViewPort Options 栏里 Constant Update 控件的状态, 如果为 True, 视窗在每一次鼠标移动后都会被刷新, 否则仅在放开鼠标后才会刷新视窗。

## 66. SetConstantUpdate &lt;constantUpdates&gt;

为 Edit UVWs 浮动窗口 ViewPort Options 栏里 Constant Update 控件设置状态。

## 67. GetShowSelectedVertices()

返回一个 Boolean 值, 表示被选择的纹理顶点是否也显示在视窗里。

## 68. SetShowSelectedVertices &lt;show&gt;

设置被选择的纹理顶点是否也显示在视窗里。参数<show>表示 Show Selected Vertices 的状态。

## 69. GetMidPixelSnape()

返回一个 Boolean 值, 表示是否使用像素中点捕捉, 如果返回 False, 表示捕捉方式为像素右下角, 如果返回 True, 表示捕捉方式为像素中心。

## 70. SetMidPixelSnape &lt;snap&gt;

设置是否使用像素中点捕捉方式。

## 71. GetMatID()

返回 Material 下拉列表的当前材质 ID 号码。

## 72. SetMatID &lt;matid&gt;

设置 Material 下拉列表的当前被选择序号。

## 73. NumberMatIDs()

返回 Material 下拉列表里的材质 ID 数目。

## 74. GetSelectedVerts()

返回一个 Bitarray 值, 表示 Edit UVWs 浮动窗口里当前被选择的纹理顶点。

## 75. SelectVerts &lt;sel&gt;

选择 Edit UVWs 浮动窗口里指定的纹理顶点, 参数<sel>为一个 Bitarray 值。

## 76. IsVertexSelected &lt;index&gt;

返回一个 Boolean 值, 表示指定序号的纹理顶点是否被选择。

## 77. MoveSelectedVertices &lt;offset&gt;

将选择纹理顶点偏移指定距离, 参数<offset>为一个 Point3 值, 表示顶点偏移的距离。

## 78. RotateSelectedVertices &lt;angle&gt;

将选择纹理顶点偏移绕其中心旋转, 参数<angle>为一个 Float 值, 表示旋转的弧度。

## 79. RotateSelectedVertices &lt;angle&gt; &lt;axis&gt;

将选择纹理顶点偏移绕指定轴旋转。

- ◆ <angle> 为一个 Float 值，表示旋转的弧度。
- ◆ <axis> 为一个 Point3 值，表示旋转轴，该值在 Window 空间下。

## 80. ScaleSelectedVertices &lt;scale&gt; &lt;dir&gt;

将选择纹理顶点偏移绕其中心缩放。

- ◆ 参数<Scale> 为一个 Float 值，表示缩放大小。
- ◆ 参数<dir> 为一个整数，表示缩放的方向：

- 0 均匀缩放 (Scales uniform)
- 1 仅在 X 方向缩放 (Scales in the x)
- 2 仅在 Y 方向缩放 (Scales in the y)

## 81. ScaleSelectedVertices &lt;scale&gt; &lt;dir&gt; &lt;axis&gt;

将选择纹理顶点偏移绕指定轴进行缩放。

参数<Scale>为一个 Float 值，表示缩放大小。

参数<dir>为一个整数，表示缩放的方向：

- 0 均匀缩放 (Scales uniform)
- 1 仅在 X 方向缩放 (Scales in the x)
- 2 仅在 Y 方向缩放 (Scales in the y)

参数<axis>为一个 Point3 值，表示旋转轴，该值在 Window 空间下。

## 82. GetVertexPosition &lt;time&gt; &lt;index&gt;

返回一个 Point3 值，表示指定顶点在指定时间的位置。

- ◆ <time> timevalue，指定时间
- ◆ <index> Integer，指定顶点序号

## 83. NumberVertices()

返回纹理顶点的数目。

## 84. MoveX &lt;p&gt;

将选择顶点的 X 坐标（绝对坐标）移到指定处。

## 85. MoveY &lt;p&gt;

将选择顶点的 Y 坐标（绝对坐标）移到指定处。

## 86. MoveZ &lt;p&gt;

将选择顶点的 Z 坐标（绝对坐标）移到指定处。

## 87. GetSelectedPolygons()

返回一个 Bitarray 值，表示胆怯当前被选择的多边形 (polygon)。

## 88. SelectPolygons &lt;sel&gt;

选择视窗里指定的多边形，参数<sel>为 BitArray 值。

## 89. IsPolygonSelected &lt;index&gt;

返回一个 Boolean 值，表示指定序号顶点是否被选择。

## 90. NumberPolygons()

返回当前对象里的多边形 (polygon) 数目。

#### 91. DetachEdgeVerts()

将没有完全被选择顶点包围的所有顶点分开。

#### 92. FlipH()

相当于按下 Edit UVWs 浮动窗口里的 Flip Horizontal 按钮。

#### 93. FlipV()

相当于按下 Edit UVWs 浮动窗口里的 Flip Vertical 按钮。

#### 94. GetLockAspect()

返回 Edit UVWs 浮动窗口里的 Lock Aspect Ratio 控件的状态。如果返回 False，图像伸展以充满整个窗口。

#### 95. SetLockAspect <aspect>

设置 Edit UVWs 浮动窗口里的 Lock Aspect Ratio 控件的状态。参数<aspect>为一个 Boolean 值。

#### 96. GetMapScale()

返回一个 Float 值，表示当用户使用 planar 贴图时的缩放因子。缩放因子越小，平面贴图越小。

#### 97. SetMapScale <scale>

设置用户使用 planar 贴图时的缩放因子。缩放因子越小，平面贴图越小。

#### 98. GetSelectionFromFace()

本方法用来选择纹理顶点：所有与当前多边形选择集相连的纹理顶点都会被选择。

#### 99. ForceUpdate <update>

设置当 Unwrap 的拓扑结构发生改变时，其反应方式。如果参数<update>为 True，贴图将被重置；否则，Unwrap 会跳过这一步。

#### 100. ZoomToGizmo <all>

将活动视窗或所有视窗对当前平面贴图 gizmo 进行缩放。

参数<all>指定是对活动视窗还是所有视窗进行缩放。

#### 101. UpdateView()

强制视窗和对话框刷新。

#### 102. SetVertexPosition <t> <index> <pos>

为指定 UVW 顶点在指定时间设置位置。

◆ <t> timeValue，指定时间

◆ <index> Integer，指定顶点

◆ <pos> point3，指定顶点在 UVW 空间的位置

### 操作 UVW 拓扑结构的方法

下面的函数用来操作 UVW 拓扑结构，可以重排或添加拓扑结构。用户在使用这些函数时应小心，因为很容易创建一些非法的拓扑结构，导致不可预期的后果。Unwrap 的拓扑结构和 Mesh 类对象相似。Unwrap 有一个 Point3 值列表，其内容为纹理顶点(texture vertice)

的位置。纹理面（texture face）包含一个整数数组，这些整数表示纹理顶点在列表中的序号。这个整数数组的长度最小为 3，取决于 UnWrap 作用的对象类型。另外如果面片类型为 Patch，还有两个整数数组用于 Handle 和 Interior Handle，数组元素同样表示纹理顶点在列表中的序号。

#### 1. MarkAsDead <index>

将指定序号顶点标记为“死”的，系统不能再使用它了。该顶点并没有被删除，只是被标记为不可使用，当需要时，可以重新利用。其意思为：当用户要添加一个顶点时，系统会先检测是否有被标记的顶点，如果有这个顶点会被重新使用，否则系统会新加一个顶点，插入到顶点列表的末尾。用户要谨慎使用本函数，因为如果将一个正在使用的顶点标记为“死”的，会导致不可预见的后果。

#### 2. NumberPointsInFace <index>

返回指定面包含的顶点数目。一个面包含的顶点数目取决于 UnWrap 作用的对象类型。如 TriMesh 类对象，顶点数为 3；Patch 类对象，顶点数为 3 或 4；Polygon 类对象，顶点数为 3 或更大。

#### 3. GetVertexIndexFromFace <faceindex> <iVertexIndex>

返回指定面上指定顶点在 Unwrap 顶点列表中的序号。如：

```
GetVertexIndexFromFace(1, 3)
```

- ◆ <FaceIndex> Integer, 指定面序号；
- ◆ <IthVertexIndex> Integer, 指定面上第几个顶点，取值范围为 1~N（面包含顶点数目）。

#### 4. GetHandleIndexFromFace <faceindex> <iVertexIndex>

返回指定面上指定 Handle 在 Unwrap 的 Handle 列表中的序号。如：

```
GetHandleIndexFromFace(1, 3)
```

- ◆ <FaceIndex> Integer, 指定面序号；
- ◆ <IthVertexIndex> Integer, 指定面的第几个 Handle，取值范围为 1~2\*N（面包含顶点数目）。

#### 5. GetInteriorIndexFromFace <faceindex> <iVertexIndex>

返回指定面上指定 Interior Handle 在 Unwrap 的 Interior Handle 列表中的序号。如：

```
GetInteriorIndexFromFace(1, 3)
```

- ◆ <FaceIndex> Integer, 指定面序号。
- ◆ <IthVertexIndex> Integer, 指定面的第几个 Interior Handle，取值范围为 1~N（面包含顶点数目）。

#### 6. GetVertexGIndexFromFace <faceindex> <iVertexIndex>

返回指定面上指定几何顶点的序号。这个几何顶点是组成 Mesh 对象的顶点，而不是纹理面的顶点。如：

```
GetVertexGeomIndexFromFace(1,3)
```

- ◆ <FaceIndex> Integer, 指定面序号;
- ◆ <IthVertexIndex> Integer, 指定面的第几个几何顶点, 取值范围为 1~N (面包含顶点数目)。

#### 7. GetHandleGIndexFromFace <faceindex><ithVertexIndex>

返回指定 Patch 面上指定几何 Handle 的序号。这个几何 Handle 是组成 Patch 对象的 Handle, 而不是纹理面的 Handle。如:

```
GetHandleGeomIndexFromFace(1,3)
```

- ◆ <FaceIndex> Integer, 指定面序号;
- ◆ <IthVertexIndex> Integer, 指定面的第几个几何 Handle, 取值范围为 1~2\*N (面包含顶点数目)。

#### 8. GetInteriorGIndexFromFace <faceindex><ithVertexIndex>

返回指定 Patch 面上指定几何 Interior Handle 的序号。这个几何 Interior Handle 是组成 Patch 对象的 Interior Handle, 而不是纹理面的 Handle。如:

```
GetHandleGeomIndexFromFace(1,3)
```

- ◆ <FaceIndex> Integer, 指定面序号;
- ◆ <IthVertexIndex> Integer, 指定面的第几个几何 Interior Handle, 取值范围为 1~N (面包含顶点数目)。

#### 9. SetFaceVertex <pos><fIndex><vIndex><sel>

为指定面上指定顶点设置位置。一般来说, 如果有几个面共用这个点, 系统先解除该点与这些面的附属关系, 然后将该点移到指定位置。比如要把面 1 上第 3 点移到(5.,5.,0), 可以使用函数:

```
setFaceVertex [.5 .5 .0] 1 3
```

如果用户不想解除指定点与相连面的附属关系, 可以使用函数 SetVertexSPosition()。

- ◆ <pos> point3, 指定顶点顶点的新位置;
- ◆ <fIndex> Integer, 指定面序号;
- ◆ <vIndex> Integer, 指定面上的顶点;
- ◆ <sel> boolean, 移动顶点后是否选择它。

#### 10. SetFaceHandle <pos><fIndex><vIndex><sel>

功能同 SetFaceVertex, 只是作用对象为 Patch Handle。

#### 11. SetFaceInterior <pos><fIndex><vIndex><sel>

功能同 SetFaceVertex, 只是作用对象为 Patch Interior Handle。

#### 12. SetFaceVertexIndex <fIndex><ithV><vIndex>

为指定面上指定顶点设置序号。比如, 要把面 1 的第 3 个顶点的序号改为顶点 100, 可以用:

```
setFaceVertexIndex 1 3 100
```

- ◆ <fIndex> Integer, 指定面序号;
- ◆ <ithV> Integer, 指定面上第几个顶点;
- ◆ <vIndex> Integer, 指定新顶点在顶点列表中的序号。

### 13. SetFaceHandleIndex <fIndex><ithV><vIndex>

功能同 setFaceVertexIndex, 只是作用对象为 Patch Handle。

### 14. SetFaceInteriorIndex <fIndex><ithV><vIndex>

功能同 setFaceVertexIndex, 只是作用对象为 Patch Interior Handle。

## 10.4.81 UVW\_Xform: Modifier (UVW 变换修改器)

### 构造函数

UVW\_Xform ...

### 属性

属性名称	数据类型	默认值	说明
<UVW_Xform>.U_Tile	Float	1.0	设置沿 U 轴的贴图重复次数。可动画
<UVW_Xform>.U_Flip	Integer	0	设置是否沿 U 轴反转贴图: 0: Off; 1: On
<UVW_Xform>.V_Tile	Float	1.0	设置沿 V 轴的贴图重复次数。可动画
<UVW_Xform>.V_Flip	Integer	0	设置是否沿 V 轴反转贴图: 0: Off; 1: On
<UVW_Xform>.W_Tile	Float	1.0	设置沿 W 轴的贴图重复次数。可动画
<UVW_Xform>.W_Flip	Integer	0	设置是否沿 W 轴反转贴图: 0: Off; 1: On
<UVW_Xform>.U_Offset	Float	0.0	设置贴图沿 U 轴的偏移量。可动画
<UVW_Xform>.Map_or_Vertex_Color	Integer	0	设置用于贴图的通道是使用 Map 通道还是使用 Vertex Color 通道: 0: Map 通道; 1: Vertex Color 通道
<UVW_Xform>.Map_Channel	Integer	1	指定一个 UVW 通道用作贴图, 并且使用它右侧的微调器来设置通道数目
<UVW_Xform>.Rotation_Angle	Float	0.0	设置旋转角度。可动画, Angle
<UVW_Xform>.Rotation_Center	Integer	0	设置贴图是否将绕着对象中心旋转

## 10.4.82 UVWmap: Modifier (UVW 贴图修改器)

### 构造函数

UVWmap ...

### 属性

属性名称	数据类型	默认值	说明
<UVWmap>.maptype	Integer	0	设置贴图坐标的类型： 0: Planar (平面贴图方式) 1: Cylindrical (柱面贴图方式) 2: Spherical (球面贴图的方式) 3: Shrink Wrap (收缩包裹的贴图方式) 4: Box (从长方体的六个面向对象表面投射贴图, 每个面采用平面贴图) 5: Face (为对象表面的每个面复制一个贴图) 6: XYZ To UVW
<UVWmap>.cap	Boolean	False	设置是否为圆柱体的端面指定平面贴图
<UVWmap>.length	Float	1.0	设置贴图 gizmo 的长度。可动画
<UVWmap>.width	Float	1.0	设置贴图 gizmo 的宽度。可动画
<UVWmap>.height	Float	1.0	设置贴图 gizmo 的高度。可动画
<UVWmap>.utile <UVWmap>.U_Tile	Float	1.0	设置沿 U 轴的贴图重复次数。可动画
<UVWmap>.uflip	Boolean	False	设置是否沿 U 轴反转贴图: 0: Off; 1: On
<UVWmap>.vtile <UVWmap>.V_Tile	Float	1.0	设置沿 V 轴的贴图重复次数。可动画
<UVWmap>.vflip	Boolean	False	设置是否沿 V 轴反转贴图: 0: Off; 1: On
<UVWmap>.wtile <UVWmap>.W_Tile	Float	1.0	设置沿 W 轴的贴图重复次数。可动画
<UVWmap>.wflip	Boolean	False	设置是否沿 W 轴反转贴图: 0: Off; 1: On
<UVWmap>.channel	Integer	0	设置贴图通道。UVW 贴图修改器默认为通道 1, 所以贴图具有默认行为方式 (即早期版本软件中的方式), 除非用户显式地更改到另一个通道。 0: Map 通道; 1: 顶点颜色通道
<UVWmap>.mapChannel	Integer	1	设置使用贴图转换的贴图通道
<UVWmap>.axis	Integer	2	指定 gizmo 的哪一轴向与对象的 Local 坐标系的 Z 轴对齐: 0: X; 1: Y; 2: Z
<UVWmap>.gizmo	SubAnim		可以移动、旋转、放缩 gizmo, 还可以为 gizmo 指定变换动画
<UVWmap.Gizmo>.position	Point3	[0,0,0]	设置 gizmo 的移动位置。可动画
<UVWmap.Gizmo>.rotation	Quat	默认值: (quat 0 0 -1 0)	设置 gizmo 的旋转角度。可动画
<UVWmap.Gizmo>.scale	Point3	[1,1,1]	
			设置 gizmo 的放缩因子。可动画

**注意** 属性 .gizmo 只有在 UVWmap Modifier 被应用到一个对象后才可用。

### 10.4.83 Vertex\_Colors: Modifier（顶点颜色修改器）

#### 构造函数

Vertex\_Colors Modifier 不能由 MAXScript 创建，而仅能用 Assign Vertex Colors utility 工具创建。

#### 属性

除通用属性外，Vertex\_Colors 没有额外的属性。

### 10.4.84 VertexPaint: Modifier（顶点绘制修改器）

#### 构造函数

VertexPaint ...

PaintLayerMod...

#### 属性

属性名称	数据类型	默认值	说明
<VertexPaint>.ignoreBackfacing <VertexPaint>.Ignore_Backfacing	Boolean	False	当设置为 On 时，可防止错误地选择背面朝向用户的子对象
<VertexPaint>.mapChannel <VertexPaint>.Map_Channel	Integer	0	设置贴图通道
<VertexPaint>.layerMode <VertexPaint>.Layer_Mode	String	“Normal”	设置层的模式
<VertexPaint>.layerOpacity <VertexPaint>.Layer_Opacity	Float	100.0	设置层的透明度，可动画
<VertexPaint>.layerIsolated <VertexPaint>.Layer_Isolated	Boolean	False	当设置为 On 时，顶点绘制将忽略从其堆栈下方接收到的任何顶点颜色。
<VertexPaint>.surviveCondense <VertexPaint>.Survive_Condense	Boolean	False	当设置为 On 时，任何压缩到单个层的操作都不会删除修改器。由于压缩到单个层执行两个独立的操作（创建一个新的颜色烘焙修改器，然后删除原有的修改器），该选项允许必要时只访问压缩到单个层的第一个功能。也就是说，可以将颜色烘焙至一个新的绘制修改器，而不强制删除旧修改器
<VertexPaint>.lightingModel <VertexPaint>.Lighting_Model	Integer	1	设置照明方式： 0: Lighting; 1: Shaded; 2: Diffuse
<VertexPaint>.colorBy <VertexPaint>.Color_By	Integer	0	设置决定颜色的方式： 1: Color By Face; 2: Color By Vertex
<VertexPaint>.castShadows <VertexPaint>.Cast_Shadows	Boolean	False	设置是否投射阴影
<VertexPaint>.useMaps <VertexPaint>.Use_Maps	Boolean	False	设置是否使用贴图

(续表)

属性名称	数据类型	默认值	说明
<VertexPaint>.useRadiosity <VertexPaint>.Use_Radiosity	Boolean	False	设置是否使用光能传递
<VertexPaint>.radiosityOnly <VertexPaint>.Radiosity_Only	Boolean	False	设置是否仅使用光能传递间接的照明
<VertexPaint>.radiosityOption <VertexPaint>.Lighting_Model	Integer	0	设置光照模式

#### 10.4.85 Vertex\_Weld: Modifier (顶点焊接修改器)

##### 构造函数

Vertex\_Weld ...

VertexWeld ...

##### 属性

<Vertex\_Weld>.threshold      Float, 默认值: 0.1, 可动画

设置顶点缝合的阈值。

#### 10.4.86 VolumeSelect: Modifier (体积选择修改器)

##### 构造函数

volumeselect ...

vol\_Select ...

##### 属性

属性名称	数据类型	默认值	说明
<Volumeselect>.level	Integer	0	设置堆栈选择级别: 0: Object (体积选择修改编辑器作用于对象级别) 1: Vertex (体积选择修改编辑器作用于顶点级别) 2: Face (体积选择修改编辑器作用于面级别)
<Volumeselect>.type	Integer	0	设置选择类型: 0: Window (如果三角面的三个顶点都处于选择容器中, 该面被选择) 1: Crossing (如果三角面的一个顶点都处于选择容器中, 该面被选择)
<Volumeselect>.invert	Boolean	False	当设置为 On 时, 取消当前的选择集, 而选择原先未选择的对象或次级结构对象
<Volumeselect>.method	Integer	0	设置体积选择类型: 0: Replace (取消上一次的选择, 使用新的体积选择结果) 1: Add (将新的体积选择结果加入到以前的选择集中) 2: Subtract (从以前的体积选择集中减去新的体积选择结果)

(续表)

属性名称	数据类型	默认值	说明
<Volumeselect>.volume	Integer	0	设置选择依据： 0: Box 1: Sphere 2: Cylinder 3: Mesh Object 4: Texture Map 5: Material Id 6: Smoothing Group
<Volumeselect>.matID <Volumeselect>.Material_ID	Integer	1	当设置为 On 时，具有指定材质 ID 码的对象或次级结构对象会被选择
<Volumeselect>.Node	Node	undefined	当属性.volume=3 (Mesh 对象) 时，用来定义选择空间的对象
<Volumeselect>.smGroup <Volumeselect>.Smoothing_Group	Integer	1	当属性.volume=6 (smoothing group) 时，如果设置为 On，具有指定光滑组的对象或次级结构对象被选择
<Volumeselect>.texture	TextureMap	undefined	当.volume=4 (texture map) 属性时，如果设置为 On，具有指定纹理贴图的对象或次级结构对象被选择
<Volumeselect>.map <Volumeselect>.Map_Channel_Type	Integer	0	设置用于贴图的通道是使用 Map 通道还是 Vertex Color 通道：0: Map 通道；1: Vertex Color 通道
<Volumeselect>.mapChannel <Volumeselect>.Map_Channel	Integer	1	当属性.volume=0 时，设置使用哪一贴图通道
<Volumeselect>.autofit <Volumeselect>.Auto_fit	Boolean	True	当设置为 On 时，重新缩放 gizmo 对象，使其体积正好与修改后的对象合适
<Volumeselect>.UseAffectRegion <Volumeselect>.Use_affect_region	Boolean	False	设置是否使用软选择功能
<Volumeselect>.falloff	Float	20.0	指定作用范围半径的大小。可动画
<Volumeselect>.pinch	Float	0.0	指定曲线在箭头尖顶点汇集的形状，负指值表现平坦的顶部，正值表现为尖锐的顶部。可动画
<Volumeselect>.bubble	Float	0.0	指定隆起曲线的曲率，指定为 1.0 时产生一个半球体隆起，降低该数值隆起曲线会变得陡峭；负值可以产生下陷的效果。可动画
<Volumeselect>.center	Point3	[0,0,0]	在次级结构层级，可以变换和动画体积选择编辑器 gizmo 的中心对象。可动画

(续表)

属性名称	数据类型	默认值	说明
off<Volumeselect>.gizmo	SubAnim		在次级结构层级，可以变换和动画体积选择编辑器 gizmo
<Volumeselect.Gizmo>.position	Point3	[0,0,0]	设置体积选择编辑器 gizmo 的位置。可动画
<Volumeselect.Gizmo>.rotation	Quat	(quat 0 0 0 1)	设置体积选择编辑器 gizmo 的旋转角度。可动画
<Volumeselect.Gizmo>.scale	Point3	[1,1,1]	设置体积选择编辑器 gizmo 的放缩因子。可动画

**注意** 属性.gizmo 和.center 只有在 Volumeselect Modifier 被应用到一个对象后才可用。

#### 10.4.87 Wave: Modifier (波浪修改器)

##### 构造函数

Wave ...

##### 属性

属性名称	数据类型	默认值	说明
<Wave>.amplitude1 <Wave>.Amplitude_1	Float	5.0	指定沿波浪修改器平面 X 轴向的振幅。可动画
<Wave>.amplitude2 <Wave>.Amplitude_2	Float	5.0	指定沿波浪修改器平面 Y 轴向的振幅。可动画
<Wave>.wavelength <Wave>.Wave_Length	Float	25.0	指定每个波浪的长度。可动画
<Wave>.phase	Float	0.0	指定波浪距离波浪中心的偏移相位。可动画
<Wave>.decay	Float	0.0	设置波浪的衰减值。如果设置为 0.0，则波纹扭曲在整个作用空间中的振幅都是一致的；如果设定了衰减数值，远离修改编辑器 gizmo 中心位置的波浪强度逐渐减弱。可动画
<Wave>.center	Point3	[0,0,0]	在次级结构层级，可以变换和动画波浪编辑器 gizmo 的中心对象。可动画
<Wave>.gizmo	SubAnim		在次级结构层级，可以变换和动画波浪编辑器 gizmo
<Wave.Gizmo>.position	Point3	[0,0,0]	设置波浪编辑器 gizmo 的位置。可动画
<Wave.Gizmo>.rotation	Quat	(quat 0 0 0 1)	设置波浪编辑器 gizmo 的旋转角度。可动画
<Wave.Gizmo>.scale	Point3	[1,1,1]	设置波浪编辑器 gizmo 的放缩因子。可动画

#### 10.4.88 XForm: Modifier(变换修改器)

##### 构造函数

XForm ...

##### 属性

属性名称	数据类型	默认值	说明
<XForm>.center	Point3	[0,0,0]	在次级结构层级，可以变换和动画变换编辑器 gizmo 的中心对象。可动画
<XForm>.gizmo	SubAnim		在次级结构层级，可以变换和动画变换编辑器 gizmo
<XForm.Gizmo>.position	Point3	[0,0,0]	设置变换编辑器 gizmo 的位置。可动画
<XForm.Gizmo>.rotation	Quat	(quat 0 0 0 1)	设置变换编辑器 gizmo 的旋转角度。可动画
<XForm.Gizmo>.scale	Point3	[1,1,1]	设置变换编辑器 gizmo 的放缩因子。可动画

## 10.5 世界空间修改器构造函数和属性

SpaceWarp Binding SpacewarpModifier 类不能直接由 MAXScript 创建，而是由方法 bindSpaceWarp() 创建并应用到对象上。大多数 SpacewarpModifier 除通用属性外，都没有额外的属性。

##### 构造函数和属性

SimpleOSMToWSMMod:

```
bindSpaceWarp <Node> <spaceBend_Node>
bindSpaceWarp <Node> <spaceNoise_Node>
bindSpaceWarp <Node> <spaceSkew_Node>
bindSpaceWarp <Node> <spaceStretch_Node>
bindSpaceWarp <Node> <spaceTaper_Node>
bindSpaceWarp <Node> <spaceTwist_Node>
```

BombBinding:

```
bindSpaceWarp <Node> <bomb_Node>
```

DeflectorBinding:

```
bindSpaceWarp <particlesys_Node> <deflector_Node>
```

DisplaceBinding:

```
bindSpaceWarp <Node> <spaceDisplace_Node>
```

FFD\_Binding:

```
bindSpaceWarp <Node> <spaceFFDBox_Node>
bindSpaceWarp <Node> <spaceFFDCyl_Node>
```

```
GravityBinding:  
bindSpaceWarp <Node> <gravity_Node>  
MotorMod:  
bindSpaceWarp <Node> <motor_Node>  
PathFollowMod:  
bindSpaceWarp <particlesys_Node> <path_Follow_Node>  
PBombMod:  
bindSpaceWarp <Node> <pbomb_Node>  
PDynaFlectMod:  
bindSpaceWarp <Node> <PDynaFlect_Node>  
POmniFlectMod:  
bindSpaceWarp <particlesys_Node> <POmniFlect_Node>  
PushMod:  
bindSpaceWarp <Node> <pushSpaceWarp_Node>  
RippleBinding:  
bindSpaceWarp <Node> <spaceRipple_Node>  
<RippleBinding>.Flexibility Float, 默认值: 1.0, 可动画  
SDeflectMod:  
bindSpaceWarp <particlesys_Node> <sdeflector_Node>  
SDynaFlectMod:  
bindSpaceWarp <Node> <SDynaFlect_Node>  
SOmniFlectMod:  
bindSpaceWarp <particlesys_Node> <SOmniFlect_Node>  
SpaceConform:  
bindSpaceWarp <Node> <conformSpaceWarp_Node>  
UDeflectorMod:  
bindSpaceWarp <particlesys_Node> <uDeflector_Node>  
UDynaFlectMod:  
bindSpaceWarp <Node> <UDynaFlect_Node>  
UOmniFlectMod:  
bindSpaceWarp <particlesys_Node> <UOmniFlect_Node>  
WaveBinding:  
bindSpaceWarp <Node> <spaceWave_Node>  
<WaveBinding>.Flexibility Float, 默认值: 1.0, 可动画  
WindBinding:  
bindSpaceWarp <Node> <wind_Node>
```

### 10.5.1 世界空间修改器分类

- ◆ Displace\_Mesh（位移网格修改器）
- ◆ Displace\_NURBS（位移 NURBS 修改器）
- ◆ LS\_Colors（LS 颜色修改器）
- ◆ MapScaler（贴图缩放修改器）
- ◆ SpaceCameraMap（摄影机贴图修改器）
- ◆ SpacePatchDeform（面片变形修改器）
- ◆ SpacePathDeform（路径变形修改器）
- ◆ SpaceSurfDeform（曲面变形修改器）
- ◆ SubdivideSpacewarpModifier（细分修改器）
- ◆ Surface\_Mapper（曲面贴图修改器）

### 10.5.2 Displace\_Mesh: SpacewarpModifier（位移网格修改器）

#### 构造函数

displace\_Mesh ...

#### 属性

Displace\_Mesh 除通用属性外，都没有额外的属性。

### 10.5.3 Displace\_NURBS: SpacewarpModifier（位移 NURBS 修改器）

#### 构造函数

displace\_NURBS ...

#### 属性

Displace\_NURBS 除通用属性外，都没有额外的属性。

### 10.5.4 LS\_Colors: SpacewarpModifier（LS 颜色修改器）

#### 构造函数

LS\_Colors ...

LSColors ...

#### 属性

属性名称	数据类型	默认值	说明
<LS_Colors>.addToColors <LS_Colors>.Add_to_colors	Boolean	False	当设置为 On 时，如果存在颜色转换结果，则将它添加到现有的顶点颜色中
<LS_Colors>.brightness	Float	50.0	控制监视器上显示图像的亮度。此控件的设置不会影响到模型中的实际照明级别。可动画

(续表)

属性名称	数据类型	默认值	说明
<LS_Colors>.contrast	Float	50.0	控制模型中亮区域和暗区域之间的对比度。可动画
<LS_Colors>.convertLight <LS_Colors>.Convert_Light	Integer	1	
<LS_Colors>.daylight	Boolean	True	确定计算中是否使用自然日光
<LS_Colors>.exterior	Boolean	False	设置是否使用模拟室外日光
<LS_Colors>.useExposureControl <LS_Colors>.Use_Exposure_Control	Boolean	False	当设置为 On 时, 将丢弃亮度、对比度、日光与室外设置, 并使用活动曝光控制的设置来代替它们
<LS_Colors>.useSelfIllumination <LS_Colors>.Use_self_illumination			当设置为 On 时, 材质自发光包含在最终顶点颜色中

**注意** LS 颜色修改器将 Lightscape 光能值转换为 3ds max 的顶点颜色。当将一个 Lightscape 模型导入 3ds max 时, 光能值保持为发光度, 它是描述射到网格上光密度的物理单位。此修改器将物理单位转换为 RGB 颜色。联合使用 Lightscape 网格修改器, 可以用来生成适合游戏引擎的网格。

### 10.5.5 MapScaler: SpacewarpModifier (贴图缩放修改器)

#### 构造函数

```
MapScaler ...
MapScalerOSM ...
```

#### 属性

属性名称	数据类型	默认值	说明
<MapScaler>.scale	Float	100.0	指定贴图重复一次的尺寸。可动画
<MapScaler>.channel	Integer	1	指定贴图通道
<MapScaler>.Wrap_Texture	Integer	1	当设置为 On 时, 贴图放缩修改编辑器将贴图均匀地包裹到对象表面: 0: Off; 1: On
<MapScaler>.uOffset	Float	0.0	设置贴图沿 u 轴方向的偏移值。可动画
<MapScaler>.vOffset	Float	0.0	设置贴图沿 v 轴方向的偏移值。可动画

### 10.5.6 SpaceCameraMap: SpacewarpModifier (摄影机贴图修改器)

#### 构造函数

```
SpaceCameraMap ...
```

#### 属性

属性名称	数据类型	默认值	说明
<SpaceCameraMap>.camera	Node	undefined	

### 10.5.7 SpacePatchDeform: SpacewarpModifier (面片变形修改器)

#### 构造函数

SpacePatchDeform ...

#### 属性

属性名称	数据类型	默认值	说明
<SpacePatchDeform>.Flip_deformation_axis	Integer	0	设置是否反转 gizmo 的方向: 0: Off; 1: On
<SpacePatchDeform>.U_Percent	Float	50.0	根据 U 向距离的百分比, 沿着 gizmo 面片的 U(水平)轴移动对象。0% 的设置将对象放置在 gizmo 面片的底部。可动画, 百分比
<SpacePatchDeform>.U_Stretch	Float	1.0	沿着 gizmo 面片的 U(水平)轴缩放对象, 可动画
<SpacePatchDeform>.V_Percent	Float	50.0	根据 V 向距离的百分比, 沿着 gizmo 面片的 V(垂直)轴移动对象。0% 的设置将对象放置在 gizmo 面片的底部。可动画, 百分比
<SpacePatchDeform>.V_Stretch	Float	1.0	沿着 gizmo 面片的 V(垂直)轴缩放对象, 可动画
<SpacePatchDeform>.rotation	Float	0.0	关于 gizmo 面片旋转被修改对象。可动画, Angle
<SpacePatchDeform>.Plane_to_Patch_Deform	Integer	0	选择对象的一个两轴平面, 使其与 gizmo 面片的 XY 平面相平行。0: XY; 1: YZ; 2: ZX

### 10.5.8 SpacePathDeform: SpacewarpModifier (路径变形修改器)

#### 构造函数

SpacePathDeform ...

#### 属性

属性名称	数据类型	默认值	说明
<SpacePathDeform>.path	Node	undefined	设置变形的路径
<SpacePathDeform>.Percent_along_path	Float	0.0	根据路径长度的百分比, 沿着 gizmo 路径移动对象。可动画, Angle
<SpacePathDeform>.Stretch	Float	1.0	使用对象的轴点作为缩放的中心, 沿着 gizmo 路径缩放对象。可动画
<SpacePathDeform>.rotation	Float	0.0	关于 gizmo 路径旋转对象。可动画, Angle

(续表)

属性名称	数据类型	默认值	说明
<SpacePathDeform>.Twist	Float	0.0	根据路径总体长度一端的旋转决定扭曲的角度。通常，变形对象只占据路径的一部分，所以产生的效果很微小。可动画，Angle
<SpacePathDeform>.axis	Integer	2	设置旋转 gizmo 路径的轴向，使其与对象的指定局部轴相对齐：0: X; 1: Y; 2: Z
<SpacePathDeform>.Flip_deformation_axis	Integer	0	设置是否反转面片变形线框的方向：0: Off; 1: On

### 10.5.9 SpaceSurfDeform: SpacewarpModifier (曲面变形修改器)

#### 构造函数

SpaceSurfDeform ...

#### 属性

属性名称	数据类型	默认值	说明
<SpaceSurfDeform>.surface	Node	undefined	设置用来变形的曲面
<SpaceSurfDeform>.U_Percent	Float	50.0	指定对象沿面片变形线框的 U 轴（水平）方向移动的百分比。移动的距离取决于变形线框在 U 轴方向的长度。默认值为 50，将对象放置在面片 U 轴的中心位置；如果设置为 0，将面片对象放置在变形线框的左边缘。可动画，百分比
<SpaceSurfDeform>.U_Stretch	Float	1.0	沿面片变形线框的 U 轴（水平）方向缩放对象。可动画
<SpaceSurfDeform>.V_Percent	Float	50.0	指定对象沿面片变形线框的 V 轴（垂直）方向移动的百分比。移动的距离取决于变形线框在 V 轴方向的长度。如果设置为 50，将对象放置在面片 U 轴的中心位置；如果设置为 0，将面片对象放置变形线框的底部。可动画，百分比
<SpaceSurfDeform>.V_Stretch	Float	1.0	沿面片变形线框的 V 轴（垂直）方向缩放对象。可动画
<SpaceSurfDeform>.rotation	Float	0.0	设置被修改对象绕变形线框的旋转角度。可动画，Angle
<SpaceSurfDeform>.Plane_to_Patch_Deform	Integer	0	指定对象进行曲面变形所依据变形线框的平面：0: XY; 1: YZ; 2: ZX
<SpaceSurfDeform>.Flip_deformation_axis	Integer	0	设置是否反转曲面变形线框的方向：0: Off; 1: On

### 10.5.10 SubdivideSpacewarpModifier: SpacewarpModifier (细分修改器)

细分 (WSM) 修改器类似于对象空间细分修改器，并且具有相同的参数。在细分的世界空间版本中，变换成世界空间坐标系之后，大小限制是在网格上。

**构造函数**

```
subdivideSpacewarpModifier...
```

**属性**

属性名称	数据类型	默认值	说明
<subdivideSpacewarpModifier>.manualUpdate <subdivideSpacewarpModifier>.Manual_Update	Integer	1	设置更新类型： 0: Manual (手动) 1: Automatic (自动) 2: Render (渲染)
<subdivideSpacewarpModifier>.showAllTriangles <subdivideSpacewarpModifier>.Show_All_Triangles	Boolean	True	控制是否显示所有的三角形，或只是显示更改了面属性的边。如果三角形显示得比较杂乱，可以用此选项来减少场景中三角形的显示
<subdivideSpacewarpModifier>.size	Float	1.0	控制细分网格中三角形的大小

**10.5.11 Surface\_Mapper: SpacewarpModifier (曲面贴图修改器)****构造函数**

```
surface_mapper ...
```

**属性**

Surface\_Mapper 类除通用属性外，没有额外的属性。

**方法**

```
updateSurfaceMapper <surface_mapper>
```

本方法对指定 NURBS 曲面执行一次手工贴图刷新。

# 第 11 章 Material (材质) 和 TextureMap (贴图)

Material 类代表在 3ds max 里赋给对象的材质。用户可以在 MAXScript 里创建各种材质（如 Standard、Blend 和 Raytrace 等），存取这些材质的各种属性，并将这些材质赋给 3ds max 对象。

Material 类由 MAXWrapper 类派生而来，并继承了 MAXWrapper 类的所有属性和方法，这些属性和方法参见第 7 章“对象超级类 MAXWrapper”。

## 11.1 Material 类通用属性和方法

### Material 类的通用属性

#### 1. <material>.name

所有材质的子类都可以存取.name 属性。

#### 2. <material>.effectsChannel

所有材质的子类都可以存取.effectsChannel 属性。

#### 3. <material>.showInViewport

获取材质级别在视窗的显示状态。

### 方法

#### 1. assignNewName <Material>

系统自动修改的材质名称，使它与别的材质不重名。其格式为 Material #1，最后的数字自动递增以确保名称的惟一性。

#### 2. okMtlForScene <Material>

检测指定材质的名称是否与场景中别的材质重名。返回 True，表示没有重名；返回 False 表示有重名。用户在将材质赋给场景对象前最好调用本函数进行检测，以避免同名材质。

#### 3. getMTLMEEditFlags(<Material> | <texture>)

返回一个 Bitarray 值，表示指定材质或纹理在 Material editor 窗口 Options 的各项设置，Bitarray 值各位依次表示：

#{MTL\_BEING\_EDITED,BACKGROUND,BACKLIGHT,VIDEO\_COLOR\_CHECK}

如果指定材质或纹理为 Material editor 窗口的当前活动槽，第一位为 1，否则为 0。

#### 4. setMTLMEEditFlags(<Material> | <texture>)<bitarray>

为指定材质或纹理的 Material editor 窗口设置各选项的设置，Bitarray 值的第一位会被忽略。

5. getMTLMeditObjType(<Material> | <texture>)

setMTLMeditObjType(<Material> | <texture>)<Integer>

存取指定材质或纹理在 Material editor 窗口的样本对象类型，有效值为：

- ◆ 1 sphere
- ◆ 2 cylinder
- ◆ 3 cube
- ◆ 4 custom

6. getMTLMeditTiling(<Material> | <texture>)

setMTLMeditTiling(<Material> | <texture>)<Integer>

存取指定材质或纹理在 Material editor 窗口的 tiling 类型，有效值为：

- ◆ 1 1x1
- ◆ 2 2x2
- ◆ 3 3x3
- ◆ 4 4x4

7. updateMTLInMedit(<Material> | <texture>)

强制系统对指定材质或纹理进行刷新，包括 Material editor 窗口和 Material Browser 窗口里的相应内容。

用户可以用函数 getMeditMaterial() 和 setMeditMaterial() 来存取 Material editor 窗口，也可以用一个虚拟数组 meditMaterials 来从 Material editor 窗口获取一个材质。

## 11.2 Material 材质类型

下面列出了 3ds max 里 Material 类的全部子类：

- ◆ Advanced\_Lighting\_Override (高级照明覆盖材质)
- ◆ Architectural (建筑材质)
- ◆ Blend (混合材质)
- ◆ Composite (合成材质)
- ◆ DoubleSided (双面材质)
- ◆ InkNPaint (卡通材质)
- ◆ Lightscape\_Mtl (Lightscape 材质)
- ◆ MatteShadow (不可见/投影材质)
- ◆ MorpherMaterial (变形材质)
- ◆ MultiMaterial (多维材质)
- ◆ NoMaterial (无材质)
- ◆ Radiosity\_Override (光能传递覆盖材质)

- ◆ RaytraceMaterial (光线跟踪材质)
- ◆ Shellac (胶合材质)
- ◆ Standard (标准材质)
- ◆ TopBottom (顶/底材质)

### 11.2.1 Advanced\_Lighting\_Override: Material (高级照明覆盖材质)

该材质可以直接控制材质的光能传递属性。

**3ds max 的构造函数**

```
advanced_lighting_override...
discreetRadiosityMaterial...
```

**Autodesk VIZ 的构造函数**

```
radiosity_override ...
discreetRadiosityMaterial ...
```

**属性**

属性名称	数据类型	默认值	说明
<Advanced_Lighting_Override>.baseMaterial	Standardmaterial	默认值: (null):Standard 设置基础材质	
<Advanced_Lighting_Override>.bumpScale	Float	1.0	在间接照明的区域中, 缩放基础材质的凹凸贴图效果。当该值为零时, 对间接照明不产生凹凸贴图。增加间接灯光凹凸比可以增强间接照明下的凹凸效果。该值不影响直接照明的基础材质区域中的凹凸量。该值不能小于零, 可动画
<Advanced_Lighting_Override>.colorBleed	Float	1.0	增加或减少反射颜色的饱和度。可动画
<Advanced_Lighting_Override>.luminanceScale	Float	0.0	设置亮度比。该参数大于零时, 缩放基础材质的自发光组件。使用该参数以便自发光对象在光能传递或光跟踪解决方案中起作用。该值不能小于零, 可动画
<Advanced_Lighting_Override>.reflectanceScale	Float	1.0	设置反射比比例。可动画
<Advanced_Lighting_Override>.transmittanceScale	Float	1.0	设置变形比例。可动画

### 11.2.2 Architectural: Material (建筑材质)

**构造函数**

```
Architectural ...
```

ArchitecturalMaterial ...

### 属性

属性名称	数据类型	默认值	说明
<Architectural>.bumpMap <Architectural>.Bump_Map	undefined	undefined	为材质指定凹凸贴图
<Architectural>.bumpMapAmount <Architectural>.Bump_Map_Amount	Float	100.0	设置凹凸贴图量
<Architectural>.bumpMapEnable <Architectural>.Bump_Map_Enable	Boolean	True	设置是否使用凹凸贴图
<Architectural>.colorBleed <Architectural>.Color_Bleed_Scale	Float	100.0	设置位移贴图量。值为 1000.0 时，位移贴图的效果最强；数量越低，位移越不明显；值为 0.0 时，根本看不到位移；负值会反转位移效果的方向
<Architectural>.cutoutMap <Architectural>.Cutout_Map	undefined	undefined	为材质指定裁切贴图
<Architectural>.cutoutMapAmount <Architectural>.Cutout_Map_Amount	Float	100.0	设置裁切贴图量。百分比
<Architectural>.cutoutMapEnable <Architectural>.Cutout_Map_Enable	Boolean	True	设置是否使用裁切贴图
<Architectural>.diffuse	Color	默认值: (color 150 150 150)	
		控制漫反射颜色。可动画	
<Architectural>.diffuseColor	Color	默认值: (color 255 255 255)	
		控制漫反射颜色。可动画	
<Architectural>.diffuseMap <Architectural>.Diffuse_Map	undefined	undefined	指定材质的漫反射贴图
<Architectural>.diffuseMapAmount <Architectural>.Diffuse_Map_Amount			设置漫反射贴图的数量。百分比
<Architectural>.diffuseMapEnable <Architectural>.Diffuse_Map_Enable	Boolean	True	设置是否使用漫反射贴图
<Architectural>.displacementMap <Architectural>.Displacement_Map	undefined	undefined	指定材质的置换贴图
<Architectural>.displacementMapAmount <Architectural>.Displacement_Map_Amount	Float	100.0	设置置换贴图的数量。百分比
<Architectural>.displacementMapEnable <Architectural>.Displacement_Map_Enabled	Boolean	True	设置是否使用置换贴图
<Architectural>.emitLuminance <Architectural>.Emit_Luminance	Boolean	False	是否使用基于亮度的发射能量
<Architectural>.filterMap <Architectural>.Filter_Map	undefined	undefined	指定材质的过滤贴图
<Architectural>.indirectBumpAmount <Architectural>.Indirect_Bump_Scale	Float	100.0	设置间接灯光的凹凸比。可动画，百分比

(续表)

属性名称	数据类型	默认值	说明
<Architectural>.intensityMap <Architectural>.Intensity_Map	undefined	undefined	指定材质的强度贴图
<Architectural>.intensityMapAmount <Architectural>.Intensity_Map_Amount	Float	100.0	设置强度贴图的数量
<Architectural>.intensityMapEnable <Architectural>.Intensity_Map_Enable	Boolean	True	设置是否使用强度贴图
<Architectural>.ior	Float	1.5	设置材质的折射率。可动画
<Architectural>.luminance	Float	0.0	设置材质的亮度值
<Architectural>.luminanceMap <Architectural>.Luminance_Map	undefined	undefined	指定材质的亮度贴图
<Architectural>.luminanceMapEnable <Architectural>.Luminance_Map_Enable	Boolean	True	设置是否使用亮度贴图
<Architectural>.mapAmounts <Architectural>.Map_Amounts	数据类型: ArrayParameter		
	默认值: #(100.0, 0.0, 0.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0)		
	设置所有用于材质贴图量的数组		
<Architectural>.maps	数据类型: ArrayParameter		
	默认值: #(undefined,undefined,undefined,undefined, undefined,undefined, undefined, undefined)		
	设置所有用于材质贴图的数组		
<Architectural>.mapsEnables <Architectural>.Map_Enables	Array-Parameter	默认值: #(True, True, True, True, True, True, True, True, True)	
		设置所有是否使用材质贴图开关状况的数组。	
<Architectural>.rawDiffuseTexture <Architectural>.Raw_Diffuse_Texture	Boolean	False	设置是否使用粗糙漫反射纹理
<Architectural>.reflectanceScale <Architectural>.Reflectance_Scale	Float	100.0	设置反射比比例
<Architectural>.sampler	Integer	0	使用采样索引号设置采样方式
<Architectural>.samplerAdaptOn <Architectural>.Sampler_Adapt_On	Boolean	False	设置是否使用 adaptive sampling
<Architectural>.samplerAdaptThreshold <Architectural>.Sampler_Adapt_Threshold	Float	0.0	设置使用 adaptive sampling 的阈值
<Architectural>.samplerAdvancedOptions <Architectural>.Sampler_Advanced_Options	Boolean	False	设置是否使用 advanced options
<Architectural>.samplerByName <Architectural>.Sampler_By_Name	String	“Max 2.5 Star”	使用采样名称设置采样方式
<Architectural>.samplerEnable <Architectural>.Sampler_Eabler	Boolean	False	设置是否使用局部超级采样器
<Architectural>.samplerQuality <Architectural>.Sampler_Quality	Float	0.1	设置采样质量值

(续表)

属性名称	数据类型	默认值	说明
<Architectural>.samplerUseGlobal <Architectural>.Use_Global_Settings	Boolean	True	设置是否使用全局超级采样设置
<Architectural>.shininess	Float	0.0	设置材质的反光度
<Architectural>.shininessMap <Architectural>.Shininess_Map	undefined	undefined	为材质指定反光度贴图
<Architectural>.shininessMapEnable <Architectural>.Shininess_Map_Enable	Boolean	True	设置是否为材质指定反光度贴图
<Architectural>.subSampleTextureOn <Architectural>.Subsample_Texture_On	Boolean	True	设置是否使用超级采样贴图
<Architectural>.supersampling	Max_2_5 _Star	默认值: ReferenceTarget:Max_2_5_Star	
		设置采样对象	
<Architectural>.template	String	“User Defined”	使用模板名称来设置模板的方式
<Architectural>.textureHeight <Architectural>.Texture_Height	Float	40.0	使用世界单位来设置贴图的高度。可动画
<Architectural>.textureUOffset <Architectural>.Texture_U_Offset	Float	0.0	使用世界单位来设置贴图在 U 方向上的偏移量。可动画
<Architectural>.textureVOffset <Architectural>.Texture_V_Offset	Float	0.0	使用世界单位来设置贴图在 V 方向上的偏移量。可动画
<Architectural>.textureWidth <Architectural>.Texture_Width	Float	40.0	使用世界单位来设置贴图的宽度。可动画
<Architectural>.translucency	Float	0.0	设置贴图的半透明值
<Architectural>.translucencyEnable <Architectural>.Translucency_Enable	Boolean	True	设置是否使用半透明贴图
<Architectural>.translucencyMap <Architectural>.Translucency_Map	undefined	undefined	为材质指定半透明贴图
<Architectural>.transmittanceScale <Architectural>.Transmittance_Scale	Float	100.0	设置透射比比例
<Architectural>.transparency	Float	0.0	设置材质的透明度值
<Architectural>.transparencyMapEnable <Architectural>.Filter_Map_Enable	Boolean	True	设置是否使用透明贴图
<Architectural>.twoSided <Architectural>.Two_Sided	Boolean	False	为材质指定透明贴图
<Architectural>.UserParam0 <Architectural>.User_Param0	Float	0.0	设置使用参数
<Architectural>.UserParam1 <Architectural>.User_Param1	Float	0.0	设置使用参数
<Architectural>.useTextureSize <Architectural>.Use_Texture_Size	Boolean	False	设置是否使用参数
<Architectural>.noExposureControl	Boolean		当设置为 On 时, 暴光不对材质起作用。可读写

(续表)

属性名称	数据类型	默认值	说明
<Architectural>.exposureControlInvertSelfIllum	Boolean		当设置为 On 时, 将反转材质的自发光值。可读写
<Architectural>.exposureControlInvertReflection	Boolean		当设置为 On 时, 将反转材质的反射值。可读写
<Architectural>.exposureControlInvertRefraction	Boolean		当设置为 On 时, 将反转材质的折射值。可读写

### 方法

```
averageColor <&color>color [map:<texturemap>] <boolean>
convertMaterial <material>material [replace:<boolean>]<boolean>
```

## 11.2.3 Blend: Material (混合材质)

### 构造函数

```
blend ...
```

### 属性

属性名称	数据类型	默认值	说明
<blend>.map1 <blend>.Map_1	Material	Standard	设置混合材质中第一个子级别材质
<blend>.map1Enabled <blend>.Map_1_Enable	Boolean	True	设置在混合时是否使用第一个子级别材质的作用效果
<blend>.map2 <blend>.Map_1	Material	Standard	设置混合材质中第二个子级别材质
<blend>.map2Enabled <blend>.Map_2_Enable	Boolean	True	设置在混合时是否使用第二个子级别材质的作用效果
<Blend>.mask	TextureMap	undefined	为混合材质指定一个混合遮罩贴图。遮罩贴图的明度数值影响两个子级材质的呈现效果, 在纯黑的图案中显示材质 2; 在纯白的图案中显示材质 1; 在介于黑与白之间的灰度图案中, 将具有一定清晰度的材质 1 与具有一定清晰度的材质 2 同时混合显示
<blend>.maskEnabled <blend>.MaskEnable	Boolean	True	设置混合材质是否使用混合遮罩贴图
<blend>.interactive	Integer	0	指定在场景视图中以实体渲染交互方式显示时, 在对象的表面显示哪一个子级材质: 0: 显示 Material 1 ; 1: 显示 Material 2
<blend>.mixAmount	Float	0.0	指定混合的百分比率, 设定为 0 表示仅有材质 1 呈现; 设定为 100 表示仅有材质 2 呈现; 设定为 60 表示 40% 的材质 1 和 60% 的材质 2 同时呈现。可动画, 百分比
<blend>.useCurve	Boolean	False	设置混合曲线是否影响混合效果

(续表)

属性名称	数据类型	默认值	说明
<blend>.upper	Float	0.75	设置高明度限定参数。可动画
<blend>.lower	Float	0.25	设置低明度限定参数。如果 upper 与 lower 两个参数的数值接近，会得到高对比度的锐化融合边缘效果；如果两个参数的数值相差很大，可以获得低对比度的柔化融合边缘效果。当改变这两数值时，可以从混合曲线反映出数值的变化。可动画

### 11.2.4 CompositeMaterial: Material (合成材质)

#### 构造函数

CompositeMaterial...

#### 属性

属性名称	数据类型	默认值	说明
<compositeMaterial>.Materialist <compositeMaterial>.Material	Array	默认值: #(Standard, undefined, ..., undefined, undefined)	
		包含组成合成材质的所有材质的数组	
<compositeMaterial>.MapEnables <compositeMaterial>.Map_Enable	Array	默认值: #(True, True, ..., True, True)	
		设置在合成材质里是否使用.Materialist 数组里的各材质，数组里的元素分别对应.Materialist 数组里的材质	
<compositeMaterial>.mixType <compositeMaterial>.Composite_Type	Array	默认值: #(0, 0, ..., 0, 0)	
		设置.Materialist 数组里每一种材质的合成方式： 0: additive (采用加色合成方式) 1: Subtractive (采用减色合成方式) 2: Mix (采用混合合成方式，利用 amount 参数控制合成色彩和不透明度的效果)	
<compositeMaterial>.amount	Array	默认值: #(100.0, 100.0, ..., 100.0, 100.0)	
		设置.Materialist 数组里每一种材质的合成百分比。 对于 additive 和 subtractive 合成方式，合成量的取值范围在 0~200 之间；对于 mix 合成方式，合成量的取值范围在 0~100 之间。可动画	
<compositeMaterial>.baseMaterial	Standard Material	(null)	MaterialList[1]的别名，指定合成材质的基础材质，默认的基础材质是标准材质，其他九种子级材质都被合成到基础材质之上

#### 注意

数组.MaterialList 里存储了材质的基础材质和子材质；数组.mapEnables 里存储了是否使用每一种子材质；数组.mixTypes 里存储了每一种子材质的合成类型；数组.amount 里存储了每一种子材质的合成百分比。如果数组.MaterialList 的长度为 N，那么其他数组的长度应该为 (N-1)，分别对应 (N-1) 个子材质。

数组.amount 的每一个元素都有一个属性别名，其格式为 amount\_X，X 表示对应元素的序号，比如 amount\_1、amount\_2。

### 11.2.5 DoubleSided: Material (双面材质)

#### 构造函数

DoubleSided ...

DoubleSidedMat ...

#### 属性

属性名称	数据类型	默认值	说明
<DoubleSided>.translucency	Float	0.0	指定透过一个材质呈现另一个材质的系数。该参数是百分率数值，取值范围为[0 100]，如果指定为 100，在外部材质叠加显示内部材质，在内部材质叠加显示外部材质；默认为 0，外部材质不透明，只能从对象的内部才能看见内部材质。可动画
<DoubleSided>.Material1 <DoubleSided>.Material_1	Material	Standard	设置双面材质中外部子级材质
<DoubleSided>.map1Enabled <DoubleSided>.Map_1_Enable	Boolean	True	设置是否显示双面材质中外部子级材质
<DoubleSided>.Material2 <DoubleSided>.Material_2	Material	Standard	设置双面材质中内部子级材质
<DoubleSided>.map2Enabled <DoubleSided>.Map_2_Enable	Boolean	True	设置是否显示双面材质中内部子级材质

### 11.2.6 InkNPaint: Material (卡通材质)

#### 构造函数

InkNPaint ...

#### 属性

属性名称	数据类型	默认值	说明
<InkNPaint>.backface_error_color	Color	默认值：(color 0 255 255)	设置标记背面错误的颜色。注意：该属性只在 MAXScript 内部使用，并不被材质用户界面所支持
<InkNPaint>.backface_error_on	Boolean	True	设置是否使用标记背面错误。注意：该属性只在 MAXScript 内部使用，并不被材质用户界面所支持
<InkNPaint>.bump_Map	TextureMap	undefined	为材质指定凹凸贴图
<InkNPaint>.bump_map_amt	Float	30.0	控制凹凸贴图数量
<InkNPaint>.bump_map_on	Boolean	False	设置是否为材质指定凹凸贴图

(续表)

属性名称	数据类型	默认值	说明
<InkNPaint>.color1 <InkNPaint>.Ink_Color	Color	默认值: (color 0 0 0)	
		设置墨水颜色	
<InkNPaint>.dsp_Map	TextureMap	undefined	为材质指定置换贴图
<InkNPaint>.dsp_map_amt	Float	20.0	控制置换贴图数量
<InkNPaint>.dsp_map_on	Boolean	False	设置是否为材质指定置换贴图
<InkNPaint>.edge_ang_ink_color	Color	默认值: (color 0 0 0)	
		设置内部边的墨水颜色	
<InkNPaint>.edge_ang_ink_Map <InkNPaint>.Internal_Map	TextureMap	undefined	为材质指定内部边的贴图
<InkNPaint>.edge_ang_ink_map_amt <InkNPaint>.Internal_Map_Amt	Float	100.0	指定材质内部边贴图的数量
<InkNPaint>.edge_ang_ink_map_on <InkNPaint>.Internal_Map_On	Boolean	False	设置是否为材质指定内部边的贴图
<InkNPaint>.edge_ang_ink_on <InkNPaint>.Internal_Ink_On	Boolean	False	设置是否为内部边使用墨水效果
<InkNPaint>.edge_ang_thresh <InkNPaint>.Edge_Ang_Thresh	Float	0.5	设置被指定给内部边的边角度阈值。可动画
<InkNPaint>.face_map_on	Boolean	False	设置是否将材质应用到几何体的面
<InkNPaint>.faceted_on	Boolean	False	设置是否像平面一样渲染曲面的每个面
<InkNPaint>.fog_bg Fog_B_G	Boolean	False	当设置为 On 时, 材质颜色的已绘制区域与背景一致。否则, 绘制区域中的背景将受到摄影机与对象之间雾的影响
<InkNPaint>.init_fail_color	Color	默认值: (color 0 255 0)	
		设置标记初始失败的颜色。注意: 该属性只在 MAXScript 内部使用, 并不被材质用户界面所支持	
<InkNPaint>.init_fail_on	Boolean	True	设置是否使用标记初始失败。注意: 该属性只在 MAXScript 内部使用, 并不被材质用户界面所支持
<InkNPaint>.ink_auto_vary_on	Boolean	False	设置是否启用墨水可变宽度
<InkNPaint>.ink_on	Boolean	True	当设置为 On 时, 会对渲染施墨。否则, 不出现墨水线
<InkNPaint>.ink_quality	Integer	1	设置墨水的质量值
<InkNPaint>.ink_width_clamp_on	Boolean	False	设置是否钳制功能来强制墨水宽度始终保持在最大值和最小值之间, 而不受照明的影响
<InkNPaint>.intersect_bias	Float	0.0	设置相交偏移值

(续表)

属性名称	数据类型	默认值	说明
<InkNPaint>.mat1 <InkNPaint>.Material_1	Material	undefined	设置用于阴影表面的可选子材质。注意：这一特点只能通过 MAXScript 来实现，在 InkNPaint 用户界面不被支持
<InkNPaint>.mat1_on Material_1_Enabled	Boolean	False	设置是否可以指定用于阴影表面的可选子材质
<InkNPaint>.matid_adj_req_on <InkNPaint>.MatID_Adj_Req_On	Boolean	True	当设置为 On 时，对相邻面之间的材质 ID 边界施墨，但不对不同对象之间施墨。否则，对两对象间的材质 ID 边界或其他不相邻面施墨
<InkNPaint>.matid_ink_color	Color	默认值：(color 0 0 0) 指定相邻面之间的材质 ID 边界施墨时墨水的颜色。可动画	当设置为 On 时，对相邻面之间的材质 ID 边界施墨，但不对不同对象之间施墨。否则，对两对象间的材质 ID 边界或其他不相邻面施墨
<InkNPaint>.matid_ink_on	Boolean	True	设置是否在不同材质 ID 之间绘制墨水
<InkNPaint>.matid_intersect_bias	Float	0.0	设置相交偏移值。可动画
<InkNPaint>.matid_Map	TextureMap	undefined	为材质 ID 指定贴图
<InkNPaint>.matid_map_amt	Float	100.0	设置材质 ID 贴图的数量
<InkNPaint>.matid_map_on	Boolean	False	设置是否为材质 ID 指定贴图
<InkNPaint>.max_ink_width <InkNPaint>.Max_Ink_Width	Float	4.0	设置最大墨水宽度。可动画
<InkNPaint>.min_ink_width <InkNPaint>.Min_Ink_Width	Float	2.0	设置最小墨水宽度。可动画
<InkNPaint>.missed_rays_error_color	Color	默认值：(color 255 0 255) 设置标记遗失光线的色彩。注意：该属性只在 MAXScript 内部使用，并不被材质用户界面所支持	当设置为 On 时，对相邻面之间的材质 ID 边界施墨，但不对不同对象之间施墨。否则，对两对象间的材质 ID 边界或其他不相邻面施墨
<InkNPaint>.missed_rays_error_on	Boolean	True	设置是否标记遗失光线。注意：该属性只在 MAXScript 内部使用，并不被材质用户界面所支持
<InkNPaint>.opaque_alpha_on	Boolean	False	设置是否启用不透明 Alpha
<InkNPaint>.out_ink_Map	TextureMap	undefined	为材质指定轮廓贴图
<InkNPaint>.out_ink_map_amt	Float	100.0	设置材质轮廓贴图的数量。可动画
<InkNPaint>.out_ink_map_on	Boolean	False	设置是否使用材质轮廓贴图
<InkNPaint>.out_ink_on	Boolean	True	设置是否使用外围轮廓墨水
<InkNPaint>.paint_color	Color	默认值：(color 79 165 210) 设置亮区颜色	当设置为 On 时，对相邻面之间的材质 ID 边界施墨，但不对不同对象之间施墨。否则，对两对象间的材质 ID 边界或其他不相邻面施墨
<InkNPaint>.paint_levels	Integer	2	设置绘制级别
<InkNPaint>.paint_Map	TextureMap	undefined	为材质指定亮区贴图
<InkNPaint>.paint_map_amt	Float	100.0	设置亮区贴图的数量
<InkNPaint>.paint_map_on	Boolean	False	设置是否为材质指定亮区贴图

(续表)

属性名称	数据类型	默认值	说明
<InkNPaint>.paint_on	Boolean	True	设置是否使用亮区绘制
<InkNPaint>.pixel_grid_on	Boolean	False	设置是否使用栅格像素。注意：这一特点只能通过 MAXScript 来实现，在 InkNPain 用户界面不被支持
<InkNPaint>.pixel_sampler <InkNPaint>.sixel_sampler	Integer	3	使用索引号选择像素采样的方式： 0: Adaptive Halton 1: Adaptive Uniform 2: Hammersley 3: Max 2.5 Star (默认)
<InkNPaint>.rayengine	Ray_Engine	默认值: ReferenceTarget:Ray_Engine	
		设置光线引擎。该属性只能在 MAXScript 里使用	
<InkNPaint>.samplerAdaptOn <InkNPaint>.Adaptive_On	Boolean	True	设置是否使用 Adaptive sampling
<InkNPaint>.samplerAdaptThreshold <InkNPaint>.Adaptive_Threshold	Float	0.1	设置 Adaptive sampling 的阈值
<InkNPaint>.samplerAdvancedOptions <InkNPaint>.Sampler_Advanced_Options	Boolean	True	设置是否开启超级采用的高级选项
<InkNPaint>.samplerByName <InkNPaint>.Sampler_By_Name	String	“Max 2.5 Star”	使用名字来指定像素采样
<InkNPaint>.samplerEnable <InkNPaint>.Sampler_On	Boolean	True	设置是否使用局部超级采样器
<InkNPaint>.samplerQuality <InkNPaint>.Sampler_Quality	Float	0.5	设置局部超级采样质量值
<InkNPaint>.samplerUseGlobal <InkNPaint>.Use_Global_Settings	Boolean	True	当设置为 On 时，使用全局采样设置
<InkNPaint>.self_overlap_bias	Float	10.0	指定重叠偏移值。可动画
<InkNPaint>.self_overlap_ink_color	Color	默认值: (color 0 0 0)	
		设置延伸重叠墨水的颜色值	
<InkNPaint>.self_overlap_ink_Map <InkNPaint>.Self_Overlap_Map	TextureMap	undefined	为材质的延伸重叠指定贴图
<InkNPaint>.self_overlap_ink_map_amt <InkNPaint>.Self_Overlap_Map_Amt	Float	100.0	设置材质延伸重叠的贴图数量

(续表)

属性名称	数据类型	默认值	说明
<InkNPaint>.self_overlap_ink_map_on	Boolean	False	设置是否为材质的延伸重叠指定贴图
<InkNPaint>.self_overlap_ink_on	Boolean	True	设置是否使用材质延伸重叠墨水
<InkNPaint>.self_underlap_bias	Float	10.0	设置相交偏移值
<InkNPaint>.self_underlap_ink_color	Color	默认值: (color 0 0 0)	
		设置延伸重叠墨水的颜色值	
<InkNPaint>.self_underlap_ink_Map <InkNPaint>.Self_Underlap_Map	TextureMap	undefined	为材质的延伸重叠指定贴图
<InkNPaint>.self_underlap_ink_map_amt <InkNPaint>.Self_Underlap_Map_Amt	Float	100.0	设置材质重叠的贴图数量
<InkNPaint>.self_underlap_ink_map_on	Boolean	False	设置是否为材质的重叠指定贴图
<InkNPaint>.self_underlap_ink_on	Boolean	False	设置是否使用材质重叠墨水
<InkNPaint>.shade_amt	Float	70.0	设置暗区百分比
<InkNPaint>.shade_amt_on Shade_Amt_On	Boolean	True	当设置为 Off 时, 将显示色样, 使用色样可以为着色区域指定不同的颜色
<InkNPaint>.shade_color	Color	默认值: (color 40 80 100)	
		指定暗区颜色	
<InkNPaint>.shade_color_Map	TextureMap	undefined	为材质暗区指定贴图
<InkNPaint>.shade_color_map_amt	Float	100.0	设置暗区贴图的数量
<InkNPaint>.shade_color_map_on	Boolean	False	设置是否为材质指定暗区贴图
<InkNPaint>.smgroup_edge_ink_color <InkNPaint>.Sharp_Edge_Ink_Color	Color	默认值: (color 0 0 0)	
		设置对象平滑组墨水颜色	
<InkNPaint>.smgroup_edge_ink_on <InkNPaint>.Sharp_Edge_Ink_On	Boolean	True	设置是否使用对象平滑组
<InkNPaint>.smgroup_edge_Map <InkNPaint>.Sharp_Edge_Map	TextureMap	undefined	指定对象平滑组的贴图

(续表)

属性名称	数据类型	默认值	说明
<InkNPaint>.smgroup_edge_map_amt <InkNPaint>.Sharp_Edge_Map_Amt	Float	100.0	指定对象平滑组的贴图数量。可动画
<InkNPaint>.smgroup_edge_map_on <InkNPaint>.Sharp_Edge_Map_On	Boolean	False	设置是否指定对象平滑组的贴图
<InkNPaint>.spec_color	Color	默认值: (color 255 255 255) 指定材质高光区的颜色	
<InkNPaint>.spec_gloss	Float	50.0	指定材质高光区的光泽度
<InkNPaint>.spec_Map	TextureMap	undefined	为材质指定高光区的贴图
<InkNPaint>.spec_map_amt	Float	100.0	设置高光区贴图的数量
<InkNPaint>.spec_map_on	Boolean	False	设置是否为材质指定高光区贴图
<InkNPaint>.spec_on	Boolean	False	设置是否使用高光区效果
<InkNPaint>.sub_mtl_amt	Float	100.0	设置子材质的数量
<InkNPaint>.subSampleTextureOn <InkNPaint>.Sample_SubTexmaps_On	Boolean	True	设置是否使用贴图超级采样
<InkNPaint>.two_side_on	Boolean	True	设置是否使用双面着色
<InkNPaint>.UserParam0 Param_0	Float	0.0	设置用户参数值
<InkNPaint>.UserParam1 Param_1	Float	0.0	设置用户参数值
<InkNPaint>.width_Map	TextureMap	undefined	为材质指定宽度贴图
<InkNPaint>.width_map_amt	Float	100.0	设置材质宽度贴图的数量
<InkNPaint>.width_map_on	Boolean	False	设置是否为材质指定宽度贴图

### 11.2.7 Lightscape\_Mtl: Material (Lightscape 材质)

#### 构造函数

Lightscape\_Mtl...

LightscapeMtl...

#### 属性

属性名称	数据类型	默认值	说明
<Lightscape_Mtl>.ambientLight <Lightscape_Mtl>.Ambient_Light	Float	0.0	控制在使用光能传递计算中将要混合的 3ds max 环境光的数量
<Lightscape_Mtl>.baseMaterial <Lightscape_Mtl>.Base_Material	数据类型: Standardmaterial		
	默认值: (null):Standard		
	显示应用光能传递照明的基础 3ds max 材质		
<Lightscape_Mtl>.brightness	Float	50.0	设置亮度值
<Lightscape_Mtl>.bumpAmount <Lightscape_Mtl>.Bump_Amount	Float	5.0	控制应用于 Lightscape 照明的凹凸贴图的长度。取值范围在-10~10 之间
<Lightscape_Mtl>.contrast	Float	50.0	设置相对值
<Lightscape_Mtl>.daylight	Boolean	True	确定计算中是否使用自然日光
<Lightscape_Mtl>.disabled	Boolean	False	确定是否计算光能传递
<Lightscape_Mtl>.exterior	Boolean	False	设置是否使用模拟室外日光

### 11.2.8 MatteShadow: Material (不可见/投影材质)

#### 构造函数

MatteShadow ...

Matte ...

#### 属性

属性名称	数据类型	默认值	说明
<MatteShadow>.opaqueAlpha <MatteShadow>.Opaque_Alpha	Boolean	True	指定是否将具有 Matte 材质的对象渲染到 Alpha 通道中。如果设置为 Off 不可见对象不包含在渲染图像的 Alpha 通道中
<MatteShadow>.applyAtmosphere	Boolean	False	设置不可见对象是否受场景中大气效果的影响
<MatteShadow>.AtmosphereDepth	Integer	0	当属性.applyAtmosphere 为 True 时, 设置不可见对象受场景中大气效果影响的方式: 0: At Background Depth (指定一种二维模式, 扫描线渲染器首先渲染场景中的雾, 然后渲染投射的阴影, 于是阴影不能被雾化, 如果想得到亮化的阴影, 就要增加阴影亮度的设置) 1: At Object Depth (指定一种三维模式, 扫描线渲染器首先渲染场景中的阴影, 然后渲染场景中的雾, 雾效将遮盖在不可见对象的表面, 但是创建的 matte/Alpha 通道不能与背景很好地融合地一起)
<MatteShadow>.receiveShadows	Boolean	False	设置在不可见对象的表面是否接受其他对象投射的阴影

(续表)

属性名称	数据类型	默认值	说明
<MatteShadow>.affectAlpha	Boolean	False	设置是否将不可见对象的表面接受的阴影渲染到 Alpha 通道
<MatteShadow>.shadowBrightness <MatteShadow>.Shadow_Brightness	Float	0.0	设置阴影的亮度。如果设置为 0.5，在不可见对象表面的阴影不被削弱；如果设置为 1.0，在不可见对象表面的阴影被亮化；如果设置为 0.0，在不可见对象表面的阴影被暗化处理，并完全掩盖不可见对象的表面。可动画
<MatteShadow>.color <MatteShadow>.Shadow_Color	Color	默认值：(color 0 0 0)	
		设置阴影的色彩。可动画	
<MatteShadow>.amount <MatteShadow>.Reflection_Amount	Float	50.0	用于控制反射效果的总量。百分比数值，取值范围为 0~100。可动画，百分比
<MatteShadow>.map	TextureMap	undefined	为反射指定贴图
<MatteShadow>.useReflMap	Boolean	True	设置是否使用反射贴图

### 11.2.9 MorpherMaterial: Material（变形材质）

#### 构造函数

MorpherMaterial ...

#### 属性

属性名称	数据类型	默认值	备注
<MorpherMaterial>.Channel_0	Float	0.0	可动画，百分比
<MorpherMaterial>.Channel_1	Float	0.0	可动画，百分比
<MorpherMaterial>.Channel_2	Float	0.0	可动画，百分比
<MorpherMaterial>.Channel_98	Float	0.0	可动画，百分比
<MorpherMaterial>.Channel_99	Float	0.0	可动画，百分比
<MorpherMaterial>.Channel_100	Float	0.0	可动画，百分比
<MorpherMaterial>.base <MorpherMaterial>.Mat_1 <MorpherMaterial>.Mat_2 <MorpherMaterial>.Mat_3 ... <MorpherMaterial>.Mat_98 <MorpherMaterial>.Mat_99 <MorpherMaterial>.Mat_100			

**注意**

在 Morph Modifier 的通道百分比和属性.Channel\_N 之间存在着对应关系：属性.Channel\_N 对应 Morph Modifier 的通道 N+1，属性.Channel\_100 对应 Morph Modifier 的基对象百分比。

属性.Channel\_N 的值为对应 Morph Modifier 的通道百分比值乘以 100。

请读者参考下面的例子，假设对象 MyMorph 定义了两个 Morph 通道，且被赋给了一个 MorpherMaterial 类材质，在用户界面里设置好通道子材质后，用下面的脚本可以存取材质及其子材质：

```
mtl=$MyMorph.Material --获取 MorpherMaterial 材质
MM_C1_percent=mtl.Channel_0/100. --获取 morph 目标 1 的百分比值并进行缩放
MM_C1_mtl=mtl[1] --获取 morph 目标 1 的 subMaterial
MM_base_percent=mtl.Channel_100/100. --获取 morph 基对象的百分比值并进行缩放
MM_base_mtl=mtl[101] --获取 morph 基对象的 subMaterial
```

### 11.2.10 MultiMaterial: Material (多维材质)

#### 构造函数

```
MultiMaterial [ numsubs:<Integer> ]
MultiSubMaterial [ numsubs:<Integer> ]
```

#### 属性

属性名称	数据类型	默认值	说明
<multiMaterial>.numsubs <multisubMaterial>.numsubs		10	设定多维次对象材质中包含的子级材质数量
<multisubMaterial>.MaterialList	ArrayParameter	#(Standard, Standard, ...Standard, Standard)	存储每一种子材质的数组
<multisubMaterial>.mapEnabled	ArrayParameter	#(True, True, ...True, True)	Boolean 类数组，为每一种子材质设置是否激活本材质
<multiMaterial>.names	ArrayParameter	#("","","","","","","")	存储每一种子材质在 Medit 里的 Slot 名称(不是子材质的真正名称)的数组
<MultiMaterial>.MaterialIDList	ArrayParameter	#(0, 1, 2, 3, 4, 5, 6, 7, 8, 9)	存储每一种子级材质的 ID 号码的数组，具有最小的 ID 号码的子级材质位于列表的最上方
<MultiMaterial>.Material1	StandardMaterial	Standard	.MaterialList[0] 属性的别名

**注意**

以上每一个数组的长度都为.numsubs 属性值。

MultiMaterial 材质的子材质按表格形式排列，所以我们也可以按数组的方法来存取子材质，如：

```
mm = multiMaterial numsubs:3
mm[1] = $foo.Material
$baz.Material = mm[2]
```

如果属性.numsubs 在创建时未指定，其默认值为 10。

子级材质数量在材质被创建后仍然可以改变，只要改变属性.numsubs 的值就可以了。

子级材质数量还可以通过改变属性.MaterialList、.mapEnabled 或.names 的.count 属性来改变。

### 11.2.11 NoMaterial: Material (无材质)

赋给对象 NoMaterial 类材质相当于在 Material Editor 窗口里将材质设为 None，也相当于给对象的.Material 属性设为 undefined。假如一个 Box 对象的材质类型为 Top\_Bottom，下面的脚本将 Top 子材质设为 None：

```
$Box01.Material.TopMaterial=noMaterial()
```

或

```
$Box01.Material.TopMaterial=undefined
```

#### 构造函数

NoMaterial ...

#### 属性

NoMaterial 没有额外的属性。

### 11.2.12 RayTraceMaterial: Material (光线跟踪材质)

#### 构造函数

RayTraceMaterial ...

#### 属性

属性名称	数据类型	默认值	说明
<RaytraceMaterial>.Ambient_Color_On	Float	0.0	当本属性为 1 时，材质使用属性.Ambient；否则，材质使用属性.Ambient_Amount
<RaytraceMaterial>.Ambient	Color	(color 0 0 0)	设置光线跟踪材质的阴影色，用来控制受环境影响的程度。如果为白色，等同于在 Standard 材质中将阴影色与过渡色锁定。可动画
<RaytraceMaterial>.Ambient_Amount	Float	0.0	如果将 Ambient_Color_On 属性设为 0，材质使用灰度级作为阴影色，本属性用于设置材质阴影色贴图的灰度值。可动画，百分比
<RaytraceMaterial>.Luminosity_Color_On	Float	0.0	当本属性为 1 时，材质使用属性.Luminosity，否则材质使用属性.Self_Illum_Amount
<RaytraceMaterial>.Luminosity	Color	(color 0 0 0)	当 Luminosity_Color_On 属性为 1 时，设置材质的自发光色，类似于标准材质中的自发光成分，但发光色不依赖于过渡色，可以为具有蓝色过渡色的对象指定红色的发光色。可动画

(续表)

属性名称	数据类型	默认值	说明
<RaytraceMaterial>.Self_Illum_Amount	Float	0.0	当.Luminosity_Color_On 属性为 0 时, 材质使用灰度级作为发光色, 本属性用于设置材质发光色贴图的灰度值。可动画
<RaytraceMaterial>.Diffuse	Color	默认值: (color 127.5 127.5 127.5)	设置材质的过渡色, 与标准材质中的过渡色含义不同, 是对象向外反射的色彩, 反射与透明效果位于过渡区色彩的上层, 如果将反射指定为 100%, (纯白色), 过渡区的色彩是不可见的。可动画
<RaytraceMaterial>.Transparency_Color_On	Float	0.0	当本属性为 1 时, 材质使用属性.Transparecy, 可以为材质指定一个透明色彩; 否则材质使用属性.Transparency_Amount, 材质使用一个滑块控制灰度级别
<RaytraceMaterial>.Transparecy	Color	(color 0 0 0)	当.Transparency_Color_On 属性为 1 时, 设置材质的透明度颜色。可动画
<RaytraceMaterial>.Transparency_Amount	Float	0.0	当.Transparency_Color_On 属性为 0 时, 材质使用灰度级别作为透明色, 本属性用于调整灰度贴图的灰度值。可动画, 百分比
<RaytraceMaterial>.Reflect_Color_On	Float	0.0	当.Reflect_Color_On 属性为 1 时, 材质使用属性.Reflect, 可以为材质指定一个反射色彩; 否则材质使用.Reflect_Amount 属性, 材质使用一个滑块控制灰度级别
<RaytraceMaterial>.Reflect	Color	(color 0 0 0)	当.Reflect_Color_On 属性为 1 时, 指定高光反射的色彩, 即经过反射过滤的环境色彩, 颜色值控制反射的量。可动画
<RaytraceMaterial>.Reflect_amount	Float	0.0	当.Reflect_Color_On 属性为 0 时, 材质使用灰度级别作为反射色, 本属性用于设置反射贴图的灰度值。可动画, 百分比
<RaytraceMaterial>.Index_of_Refraction	Float	100.0	指定光线跟踪材质的折射率 (IOR), 用来控制材质折射光线的程度。如果折射率设定为 1.0 (空气折射率), 在透明对象后面的其他对象不发生扭曲变形; 如果折射率设定为 1.5, 透明对象后面的其他对象如同透过玻璃一样产生扭曲变形。可动画
<RaytraceMaterial>.Spec_Color	Color	默认值: (color 255 255 255)	设置反射高光的颜色。可动画
<RaytraceMaterial>.Specular_Level	Float	50.0	指定高光区的强度, 增加该数值使高光更为明亮。可动画, 百分比
<RaytraceMaterial>.Glossiness	Float	40.0	设置光泽度, 影响高光区的尺寸, 增大光泽数值, 高光区变得小而锐。可动画, 百分比
<RaytraceMaterial>.Softten	Float	0.1	用于柔化高光的效果, 如果柔化值设置得高, 反射高光非常类似橡胶材质的效果。可动画
<RaytraceMaterial>.Extra_Lighting	Color	(color 0 0 0)	设置附加灯光颜色, 为光线跟踪材质对象加入一盏局部灯光, 该灯光只作用于当前材质, 不会影响整个场景的光环境。可动画

(续表)

属性名称	数据类型	默认值	说明
<RaytraceMaterial>.translucency	Color	(color 0 0 0)	设置半透明颜色，用于创建半透明的效果，半透明色是一种无方向性的漫反射。可动画
<RaytraceMaterial>.Fluorescence	Color	(color 0 0 0)	设置荧光颜色，产生在黑暗环境中，荧光涂料发光的效果，荧光色不受场景中灯光环境的影响。增大荧光色的饱和度可以加大荧光涂料的效果，可以为动画角色的皮肤和眼睛设置轻微的荧光效果。可动画
<RaytraceMaterial>.Fluorescence_Bias	Float	70.0	设置荧光偏移量。当偏移量设定为 0.5 时，荧光就像对象的过渡色一样，偏移量高于 0.5 时会增加荧光效果，使对象比场景中的其他对象都要亮。可动画，百分比
<RaytraceMaterial>.Wire_Size	Float	100.0	指定线框模式 (wireframe) 下线框的粗细。可动画，百分比
<RaytraceMaterial>.Color_Density_Enable	Boolean	False	设置是否使用颜色密度控制透明材质，如果材质是不透明的，不会产生密度效果。可动画
<RaytraceMaterial>.Color_Density_Start	Float	0.0	依据场景距离单位，指定密度色开始呈现的厚度。对象的厚度至少要达到密度色开始点的厚度。可动画
<RaytraceMaterial>.Color_Density_End	Float	25.0	依据场景距离单位，指定密度色达到 Amount 指定的最大量时的厚度。可动画
<RaytraceMaterial>.Color_Density_Amount	Float	1.0	指定密度色效果的级别，取值范围为 0~1.0，减小该设置会降低密度色的效果。可动画
<RaytraceMaterial>.Color_Density_Color	Color	默认值：(color 255 255 255) 设置基于对象的厚度指定传播的色彩，可以模拟其他对象透过有色玻璃被染上颜色时，依据透明玻璃本身的不同厚度，呈现不均匀的染色效果。可动画	
<RaytraceMaterial>.Fog_Enable	Boolean	False	开/关光线跟踪材质密度雾效果。可动画
<RaytraceMaterial>.Fog_Start	Float	0.0	依据场景距离单位，指定密度雾开始呈现的厚度。可动画
<RaytraceMaterial>.Fog_End	Float	25.0	依据场景距离单位，指定密度雾达到 Amount 指定的最大量时的厚度。可动画
<RaytraceMaterial>.Fog_Amount	Float	1.0	指定密度雾效果的级别，取值范围为 0~1.0，减小该设置会降低密度色的效果，使雾更为透明。可动画
<RaytraceMaterial>.Fog_Color	Color	(color 255 255 255)	设置密度雾的颜色值。可动画
<RaytraceMaterial>.Reflection_Type	Integer	0	设置反射类型。 0: Default (默认，反射对象的过渡色) 1: Additive (附加，反射被附加到过渡色中) 可动画

(续表)

属性名称	数据类型	默认值	说明
<RaytraceMaterial>.Reflection_Gain	Float	0.5	设置反射的亮度，本属性值越低，反射越明亮，如果将本属性指定为 1.0，反射是不可见的。可动画
<RaytraceMaterial>.Enable_Raytraced_Reflections	Boolean	False	开/关光线跟踪中的反射设置
<RaytraceMaterial>.Enable_Raytraced_Refractions	Boolean	True	可以开/关光线跟踪中的折射设置
<RaytraceMaterial>.Reflection_Falloff_Mode	Integer	0	设置反射衰减的方式： 0: Off (关闭衰减控制) 1: Linear (在 start 和 end 参数之间进行线性衰减计算) 2: Inverse Square (使用 start 参数进行反转平方衰减计算) 3: Exponential (在 start 和 end 参数之间进行指数衰减计算) 可动画
<RaytraceMaterial>.Reflection_Falloff_End_Distance	Float	100.0	指定反射效果变黑的距离。可动画
<RaytraceMaterial>.Refraction_Falloff_Mode	Integer	0	设置折射衰减的方式： 0: Off (关闭衰减控制) 1: Linear (线性衰减计算) 2: Inverse Square (使用 start 参数进行反转平方衰减计算) 3: Exponential (在 start 和 end 参数之间进行指数衰减计算) 可动画
<RaytraceMaterial>.Refraction_Falloff_End_Distance	Float	100.0	指定折射效果变黑的距离。可动画
<RaytraceMaterial>.Bump_Map_Effect	Float	1.0	调整光线跟踪反射/折射的凹凸贴图效果。可动画
<RaytraceMaterial>.Override_Global_Antialiasing_Settings	Boolean	False	设置是否启用局部抗锯齿处理。可动画
<RaytraceMaterial>.Adaptive_Antialiasing_On	Boolean	False	当.Adaptive_Antialiasing_On 属性为 True，忽略通用抗锯齿设置，使用复合处理适配抗锯齿处理，这个参数只在当属性.Override_Global_Antialiasing_Settings = True 时才起作用。可动画

(续表)

属性名称	数据类型	默认值	说明
<RaytraceMaterial>. Options__Raytracer_Enable	Boolean	True	开关光线跟踪器的设置。可动画
<RaytraceMaterial>. Options__Antialiasing_Enable	Boolean	True	是否启用通用抗锯齿处理。可动画
<RaytraceMaterial>. Options__Self_Reflect_Refra	Boolean	True	开关光线反射和折射的设置。可动画
<RaytraceMaterial>. Options__Raytrace_Atmospherics	Boolean	True	指定是否对场景大气效果进行光线跟踪，这些大气效果包括 fire(火焰)、fog(雾)、volume light (体积光) 等。可动画
<RaytraceMaterial>. Options__Reflect_Refra_Material_ID_s	Boolean	True	当设为 On 时，为光线跟踪反射指定材质的 ID 号码，并将材质的 ID 号码传递到材质特效通道 (G 缓冲)。可动画
<RaytraceMaterial>. Options__Raytrace_Objects_in_Glass	Boolean	True	设置是否对光线跟踪对象内部的对象执行光线跟踪计算。可动画
<RaytraceMaterial>. Options__Raytrace_Atmospherics_in_Glass	Boolean	True	设置是否对光线跟踪对象内部的大气效果执行光线跟踪计算。可动画
<RaytraceMaterial>. Options__Color_Density_Fog_Enable	Boolean	True	指定是否计算密度色/密度雾效果。可动画
<RaytraceMaterial>. Local_Threshold	Float	0.1	指定适配计算的灵敏度。取值范围为 0~1，设置为 0 时总是计算最大的光线数量；设置为 1 时总是计算最初光线数量。可动画
<RaytraceMaterial>. Local_Min_Rays	Integer	4	设置每个像素进行光线计算的初始数量。可动画
<RaytraceMaterial>. Local_Max_Rays	Integer	32	设置每个像素进行光线计算的最大数量。可动画
<RaytraceMaterial>. Local_Blu_Offset	Float	0.0	设置对光线跟踪反射与折射结果进行虚化的程度，虚化过程中不考虑场景的远近，本属性单位为像素。可动画
<RaytraceMaterial>. Local_Blu_Aspect	Float	1.0	通过改变图像纵横的方式进行虚化处理。可动画
<RaytraceMaterial>. Local_Defocus	Float	0.0	散焦是基于场景距离的虚化，靠近表面的对象不进行虚化处理；远离表面的对象进行虚化处理。可动画
<RaytraceMaterial>. Local_Defocus_Aspect	Float	1.0	通过改变图像纵横比的方式进行散焦处理。可动画
<RaytraceMaterial>. ambientMap	TextureMap	undefined	为对象的阴影区指定图像贴图

(续表)

属性名称	数据类型	默认值	说明
<RaytraceMaterial>.ambientMapAmount	Float	100.0	指定阴影区贴图影响材质效果的百分比
<RaytraceMaterial>.ambientMapEnable	Boolean	False	设置是否使用为对象阴影区指定的图像贴图
<RaytraceMaterial>.bumpMap	TextureMap	undefined	为对象指定凹凸贴图
<RaytraceMaterial>.bumpMapAmount	Float	100.0	设置控制对象凹凸的深度
<RaytraceMaterial>.bumpMapEnable	Boolean	False	设置是否使用为对象指定的凹凸贴图
<RaytraceMaterial>.diffuseMap	TextureMap	undefined	为对象的过渡区指定图像贴图
<RaytraceMaterial>.diffuseMapAmount	Float	100.0	指定过渡区贴图影响材质效果的百分比
<RaytraceMaterial>.diffuseMapEnable	Boolean	False	设置是否使用为对象过渡区指定的图像贴图
<RaytraceMaterial>.displacementMap	TextureMap	undefined	为对象指定置换贴图
<RaytraceMaterial>.displacementMapAmount	Float	100.0	设置置换贴图影响材质效果的百分比
<RaytraceMaterial>.displacementMapEnable	Boolean	False	设置是否使用为对象指定的置换贴图
<RaytraceMaterial>.reflectionMap	TextureMap	undefined	为对象的反射区指定图像贴图
<RaytraceMaterial>.reflectionMapAmount	Float	100.0	指定反射贴图影响材质效果的百分比
<RaytraceMaterial>.reflectionMapEnable	Boolean	False	设置是否使用为对象指定的反射区图像贴图
<RaytraceMaterial>.refractionMap	TextureMap	undefined	为对象的折射区指定图像贴图
<RaytraceMaterial>.refractionMapAmount	Float	100.0	指定折射贴图影响材质效果的百分比
<RaytraceMaterial>.refractionMapEnable	Boolean	False	设置是否使用为对象指定的折射区图像贴图
<RaytraceMaterial>.glossinessMap	TextureMap	undefined	为对象的光泽度指定贴图
<RaytraceMaterial>.glossinessMapAmount	Float	100.0	指定对象的光泽度贴图影响材质效果的百分比
<RaytraceMaterial>.glossinessMapEnable	Boolean	False	设置是否使用为光泽度指定的贴图
<RaytraceMaterial>.specularLevelMap	TextureMap	undefined	为对象的高光级别指定贴图

(续表)

属性名称	数据类型	默认值	说明
<RaytraceMaterial>.specularLevelMapAmount	Float	100.0	指定对象的高光级别贴图影响材质效果的百分比
<RaytraceMaterial>.specularLevelMapEnable	Boolean	False	设置是否使用为高光级别指定的贴图
<RaytraceMaterial>.luminosityMap	TextureMap	undefined	为对象的发光度指定贴图
<RaytraceMaterial>.luminosityMapAmount	Float	100.0	指定对象的发光度贴图影响材质效果的百分比
<RaytraceMaterial>.luminosityMapEnable	Boolean	False	设置是否使用为发光度指定的贴图
<RaytraceMaterial>.transparencyMap	TextureMap	undefined	为对象的透明度指定贴图
<RaytraceMaterial>.transparencyMapAmount	Float	100.0	指定对象的透明度贴图影响材质效果的百分比
<RaytraceMaterial>.transparencyMapEnable	Boolean	False	设置是否使用为透明度指定的贴图
<RaytraceMaterial>.environmentMap	TextureMap	undefined	为对象的环境区指定贴图
<RaytraceMaterial>.environmentMapAmount	Float	100.0	指定对象的环境区贴图影响材质效果的百分比
<RaytraceMaterial>.environmentMapEnable	Boolean	False	设置是否使用为环境区指定的贴图
<RaytraceMaterial>.transEnvMap	TextureMap	undefined	为对象的透明环境区指定贴图
<RaytraceMaterial>.transEnvMapAmount	Float	100.0	指定对象的透明环境区贴图影响材质效果的百分比
<RaytraceMaterial>.transEnvMapEnable	Boolean	False	设置是否使用为透明环境区指定的贴图
<RaytraceMaterial>.iorMap	TextureMap	undefined	为对象的IOR区指定贴图
<RaytraceMaterial>.iorMapAmount	Float	100.0	指定对象的IOR区贴图影响材质效果的百分比
<RaytraceMaterial>.iorMapEnable	Boolean	False	设置是否使用为IOR区指定的贴图
<RaytraceMaterial>.translucencyMap	TextureMap	undefined	为对象的半透明度指定贴图
<RaytraceMaterial>.translucencyMapAmount	Float	100.0	指定对象的半透明度贴图影响材质效果的百分比
<RaytraceMaterial>.translucencyMapEnable	Boolean	False	设置是否使用为半透明度指定的贴图

(续表)

属性名称	数据类型	默认值	说明
<RaytraceMaterial>.extraLightingMap	TextureMap	undefined	为对象的附加灯光指定贴图
<RaytraceMaterial>.extraLightingMapAmount	Float	100.0	指定对象的附加灯光贴图影响材质效果的百分比
<RaytraceMaterial>.extraLightingMapEnable	Boolean	False	设置是否使用为附加灯光指定的贴图
<RaytraceMaterial>.flourescenceMap	TextureMap	undefined	为对象的 Flourescence 区指定贴图
<RaytraceMaterial>.flourescenceMapAmount	Float	100.0	指定对象的 Flourescence 区贴图影响材质效果的百分比
<RaytraceMaterial>.flourescenceMapEnable	Boolean	False	设置是否使用为 Flourescence 区指定的贴图
<RaytraceMaterial>.colorDensityMap	TextureMap	undefined	为对象的 colorDensity 区指定贴图
<RaytraceMaterial>.colorDensityMapAmount	Float	100.0	指定对象的 colorDensity 区贴图影响材质效果的百分比
<RaytraceMaterial>.colorDensityMapEnable	Boolean	False	设置是否使用为 colorDensity 区指定的贴图
<RaytraceMaterial>.fogColorMap	TextureMap	undefined	为对象的 fogColor 区指定贴图
<RaytraceMaterial>.fogColorMapAmount	Float	100.0	指定对象的 fogColor 区贴图影响材质效果的百分比
<RaytraceMaterial>.fogColorMapEnable	Boolean	False	设置是否使用为 fogColor 区指定的贴图
<RaytraceMaterial>.diffusionMap	TextureMap	undefined	为对象的 diffusion 区指定贴图
<RaytraceMaterial>.diffusionMapAmount	Float	100.0	指定对象的 diffusion 区贴图影响材质效果的百分比
<RaytraceMaterial>.diffusionMapEnable	Boolean	False	设置是否使用为 diffusion 区指定的贴图
<RaytraceMaterial>.specularMap	TextureMap	undefined	为对象的高光区颜色指定贴图
<RaytraceMaterial>.specularMapAmount	Float	100.0	指定对象的高光区颜色贴图影响材质效果的百分比
<RaytraceMaterial>.specularMapEnable	Boolean	False	设置是否使用为高光区颜色指定的贴图
<RaytraceMaterial>.Bounce_Coefficient	Float	1.0	指定对象在碰撞后的反弹高度，弹性系数值越高弹力越大，如果设定为 1，表示是理想的弹性碰撞，在每次撞击过程中都不会有能量的衰减。硬的金属和超级弹力球的弹性系数接近于 1；石墨的弹性系数接近于 0。可动画

(续表)

属性名称	数据类型	默认值	说明
<RaytraceMaterial>.Static_Friction	Float	0.0	指定让一个对象沿一个表面开始运动时施加需要的拉力。磁悬浮的静摩擦力为 0, 砂纸表面的静摩擦力在 0.5~0.8 之间; 高粘度的表面静摩擦力近似为 1。可动画
<RaytraceMaterial>.Sliding_Friction	Float	0.0	指定让一个对象沿一个表面持续运动时需要施加的拉力。一旦一个对象开始沿另一个对象表面滑动, 静摩擦力消失, 滑动摩擦力出现, 通常滚动摩擦力要比静态摩擦力小。可动画
<RaytraceMaterial>.Attenuation_Start	Float	0.0	设置光线开始 Attenuate 的距离
<RaytraceMaterial>.Attenuation_Exponent	Float	2.0	设置光线完全 Attenuate 的距离
<RaytraceMaterial>.Attenuation_Color_Mode	Integer	0	设置当光线 Attenuate 时光线的颜色: 0: Background 1: 使用属性. Attenuation_Color 指定的颜色
<RaytraceMaterial>.Attenuation_Color	Color	(color 0 0 0)	设置光线 Attenuate 时的颜色
<RaytraceMaterial>.Attenuation_Near	Float	1.0	设置光线在开始反射/折射时的强度。取值范围为[0.0 1.0]
<RaytraceMaterial>.Attenuation_Control_1	Float	0.6666	控制曲线在开始时的形状
<RaytraceMaterial>.Attenuation_Control_2	Float	0.3333	控制曲线在结束时的形状
<RaytraceMaterial>.Attenuation_Far	Float	0.0	设置光线在结束反射/折射时的强度。取值范围为[0.0 1.0]
<RaytraceMaterial>.Blur_Map	Boolean	False	当设置为 On 时, 系统使用指定的贴图确定虚化偏移的数量, 贴图白色的部位创建完全的虚化偏移效果, 贴图黑色的部分不进行虚化偏移处理。如: 假设贴图为一个 Checker 类贴图, 虚化偏移每隔一个方格才发生
<RaytraceMaterial>.Defocus_Map	Boolean	False	当设置为 On 时, 使用指定的贴图确定散焦处理的数量, 贴图白色的部位创建完全的散焦处理效果, 贴图黑色的部分不进行散焦处理
<RaytraceMaterial>.Enable_Reflection_Falloff	Boolean	False	设置是否使用反射衰减
<RaytraceMaterial>.Reflection_Falloff_Distance	Float	0.0	设置反射完全消失的距离
<RaytraceMaterial>.Enable_Refraction_Falloff	Boolean	False	设置是否使用折射衰减

(续表)

属性名称	数据类型	默认值	说明
<RaytraceMaterial>.Anisotropy	Float	50.0	控制高光区形状的各向异性。设置为 0，高光区是圆形的；设置为 100 高光区变成细长的椭圆形。可动画，百分比
<RaytraceMaterial>.Orientation	Float	0.0	指定各向异性高光长轴的方向。可动画，百分比
<RaytraceMaterial>.sampler_param_block	SubAnim	默认值: SubAnim:sampler_Param_Block	
		获取采样参数，只读	
<RaytraceMaterial>.sampler_param_block.sampler_on	Boolean	False	设置是否进行超级采样。可动画
<RaytraceMaterial>.sampler_param_block.sampler_quality	Float	0.5	设置超级采样质量值。在最小值 0.0 时，对每个像素采样 4 次。在最大值 1.0 时，对每个像素采样 40 次左右（根据明暗器是否为激活状态，对此值进行调整）。范围为 0.0~1.0。可动画

### 11.2.13 StandardMaterial: Material (标准材质)

#### 构造函数

StandardMaterial ...

Standard ...

#### 属性

属性名称	数据类型	默认值	说明
<Standard>.shaderType <Standard>.Shader_Type	Integer	1	设置材质的明暗方式： 0: Anisotropic 1: Blinn 2: Metal 3: Multi-Layer 4: Oren-Nayar-Blinn 5: Phong 6: Strauss 7: Translucent
<Standard>.shaderByName <Standard>.Shader_Name	String	“Blinn”	用字符串来设置材质的明暗方式。有效的字符串有：“anisotropic”、“blinn”、“metal”、“multi-layer”；“oren-nayar-blinn”、“phong”、“strauss”、“Translucent Shader” 注意：.shaderType 属性和.shaderByName 相互关联，改变其中一个属性的值会自动改变另一个的值

(续表)

属性名称	数据类型	默认值	说明
<Standard>.wire	Boolean	False	设置是否可将材质以结构线框方式进行渲染
<Standard>.twoSided <Standard>.Two_sided	Boolean	False	设置是否能将材质指定到造型的正反两面
<Standard>.faceMap <Standard>.Face_Map	Boolean	False	设置是否将材质指定到几何体的每个表面, 如果材质是贴图材质, 面贴图方式不需要贴图坐标, 贴图会自动指定到对象的每一个表面
<Standard>.faceted	Boolean	False	设置是否渲染对象的每一个小平面, 与平均的明暗渲染方式不同
<Standard>.adTextureLock <Standard>.Ambient_Diffuse_Texture_Lock	Boolean	False	设置是否将 Diffuse 色彩区与 Ambient 色彩区的贴图设置锁定在一起, 使这两个色彩区拥有相同的贴图
<Standard>.adLock <Standard>.Ambient_Diffuse_Lock	Boolean	False	设置是否将 Diffuse 色彩区与 Ambient 色彩区设置锁定在一起, 使这两个色彩区拥有相同颜色
<Standard>.opacityType <Standard>.Opacity_Type	Intege	0	设置透明类型: 0: Filter 1: Subtractive 2: Additive
<Standard>.opacity	Float	100.0	设置材质的不透明度百分比。可动画, 百分比
<Standard>.filterColor <Standard>.Filter_Color	Color	默认值: (color 127.5 127.5 127.5) 设置透明滤镜的色彩, 用于产生有色的透明对象(如蓝色的钻玻璃)效果。可动画	
<Standard>.opacityFallOffType <Standard>.Falloff_Type	Intege	0	
<Standard>.opacityFallOff <Standard>.Falloff	Float	0.0	设置在最外面或最里面边缘的透明度。可动画, 百分比
<Standard>.ior <Standard>.Index_of_Refraction	Float	1.5	设置在光线跟踪和折射贴图中使用的折射率(IOR)。如果指定折射率为 1.0(空气折射率), 在透明对象后面的其他对象不发生扭曲变形; 如果折射率设定为 1.5, 透明对象后的其他对象如同透过玻璃一样产生扭曲变形。可动画, 百分比
<Standard>.wireSize <Standard>.Wire_Size	Float	1.0	设置线框材质模式下线框的粗细。可动画
<Standard>.wireUnits <Standard>.Wire_Units	Integer	0	设置线框粗细度量单位: 0: Pixels; 1: Units
<Standard>.applyReflectionDimming <Standard>.Apply_reflection_Dimming	Boolean	False	当设置为 On 时, 表示应用反射模糊; 否则, 直射光线不会影响反射贴图材质

(续表)

属性名称	数据类型	默认值	说明
<Standard>.dimLevel <Standard>.Dim_Level	Float	0.0	设置在阴影中暗化含量。如果指定为 0.0，在阴影中的反射贴图是全黑的；如果指定为 0.5，反射贴图被暗化一半；如果指定为 1.0，反射贴图不暗化，如同取消 Apply 选项，可动画
<Standard>.reflectionLevel <Standard>.Reflection_Level	Float	3.0	设置影响不在阴影区的反射强度。加大该设置可以使反射强度提高，补偿反射暗化对反射贴图材质表面的影响。可动画
<Standard>.sampler <Standard>.Pixel_Sampler	Integer	3	设置采样类型： 0: Adaptive Halton (适应半色调) 1: Adaptive Uniform (适应统一) 2: Hammersley (空间离散采样) 3: Max 2.5 Star (2.5 版本星形)
<Standard>.samplerByName <Standard>.Sampler_Name	String	“Max 2.5 Star”	使用字符串设置采样类型，有效字符串有：“Adaptive Halton”、“Adaptive Uniform”、“Hammersley”、“Max 2.5 Star” 属性.sampler 和.samplerByName 相互关联，改变其中一个属性的值会自动改变另一个的值
<Standard>.samplerEnable <Standard>.Sampler_Enable	Boolean	False	设置是否为当前材质指定超级采样。可动画
<standard>.subSampleTextureOn <standard>.SubSample_Textures	Boolean	True	设置是否为当前材质的贴图指定超级采样。当设为 On 时，为材质贴图进行指定的超级采样处理；反之，使用像素平均值对贴图进行超级采样处理
<Standard>.samplerQuality <Standard>.Sampler_Enable	Float	0.5	设置采样质量，范围在 0.0~1.0 之间。采样质量为 0.0 表示只对当前像素周围的四个邻近像素进行采样；采样质量为 1.0 表示对当前像素周围的 36 到 40 个邻近像素进行采样，采样质量越高，所耗费的计算时间越长。可动画
<Standard>.samplerAdaptOn <Standard>.Adaptive_On	Boolean	True	设置是否为当前材质超级采样使用 adaptive。当设为 On 时，可以使用一个采样阈值的设置，如果像素色彩的变化小于阈值的设定时，会减少估样的数量。利用该选项可以有效地加快超级采样的计算中心时间
<Standard>.SamplerAdaptThreshold <Standard>.Adaptive_Threshold	Float	0.1	控制 Adaptive 选项的作用方式，取值范围 0.0 到 1.0 之间，如果指定为 0.0 等同如将属性.Adaptive_On 选项设为 False。当颜色的改变大于指定阈值时，adaptive 方法使用属性.SamplerQuality 指定的采样质量，如果颜色的改变小于指定阈值时，adaptive 方法使用小于属性.Sampler Quality 指定的采样质量，以节约计算时间
<Standard>.SamplerAdvancedOptions <Standard>.Advanced_Options	Boolean	True	本属性虽然被定义为一个属性，但并不能被 Standard 类材质使用

(续表)

属性名称	数据类型	默认值	说明	
<standard>.UserParam0 <standard>.Optional_Param0	Float	0.0	本属性虽然被定义为一个属性，但并不能被 Standard 类材质使用	
<standard>.UserParam1 <standard>.Optional_Param1	Float	0.0	本属性虽然被定义为一个属性，但并不能被 Standard 类材质使用	
以下三个贴图数组的长度都为 24，存储与材质贴图有关的信息				
<Standard>.maps	数据类型：ArrayParameter			
	默认值：#(undefined, undefined, ..., undefined, undefined)			
	存储每一通道的纹理贴图。每一通道的代表的含义取决于材质的明暗方式。比如对 Blinn 类材质，第 5 个元素代表 Glossiness 通道。各着色模式对应的贴图通道参见下表“材质各类着色模式对应的贴图通道”			
<Standard>.mapEnables <Standard>.Map_Enables	数据类型：ArrayParameter			
	默认值：#(False, False, ..., False, False)			
	设置是否使用属性.maps 定义的贴图			
<Standard>.mapAmounts <Standard>.Map_Amounts	数据类型：ArrayParameter			
	默认值：#(100.0, 100.0, ..., 100.0, 100.0)			
	设置贴图影响材质效果的百分比，例如一个过渡区贴图如果指定为 100%，它将完全覆盖基础材质；如果指定为 50%，则呈现半透明的效果，基础材质会从贴图下透过来；如果指定为 0，等同于关闭过渡区的贴图。可动画			
<Standard>.bounce <Standard>.Bounce_Coefficient	Float	1.0	指定对象在碰撞后的反弹的高度，弹性系数值越高弹力越大，如果设定为一，表示是理想的弹性碰撞，在每次撞击过程中都不会有能量的衰减硬的金属和超级弹力球的弹性系数接近于 1；石墨的弹性系数接近于 0。可动画	
<Standard>.staticFriction <Standard>.Static_Friction	Float	0.0	指定让一个对象沿另一个对象表面开始运动时施加的最大拉力。磁悬浮的静摩擦力为 0，砂纸表面的静摩擦力在 0.5~0.8 之间；高粘度的表面静摩擦力近似为 1。可动画	
<Standard>.SlidingFriction <Standard>.Sliding_Friction	Float	0.0	指定让一个对象沿另一个对象表面持续运动时施加的拉力。一旦一个对象开始沿另一个对象表面滑动，静摩擦力消失，滑动摩擦力出现，通常滚动摩擦力要比静态摩擦力小。可动画	
<Standard>.ambient <Standard>.Ambient_Color	Color	默认值：(color 25.5 25.5 25.5)		
		指定对象表面阴影区的色彩。可动画		
<Standard>.diffuse <Standard>.Diffuse_Color	Color	默认值：(color 127.5 127.5 127.5)		
		指定对象表面在最佳光照条件下呈现的本色。可动画		

(续表)

属性名称	数据类型	默认值	说明
<Standard>.specular <Standard>.Specular_Color	Color	默认值: (color 229.5 229.5 229.5)	
		指定对象在强光亮光点处的颜色。可动画	
<Standard>.dsLock <Standard>.Diffuse_Specular_Lock	Boolean	False	设置是否将 diffuse 色彩区与 specular 色彩区设置锁定在一起，使这两个色彩区拥有相同的颜色
<Standard>.useSelfIllumCol <Standard>.Use_Self_Illumination	Boolean	False	设置是否为材质指定一个特殊的自发光色彩；如果设为 Off，材质使用过度色作为自发光色彩
<Standard>.selfIllumAmount <Standard>.Self_Illumination	Float	0.0	设置材质自发光的强度。可动画，百分比
<Standard>.selfIllumColor <Standard>.Self_Illum_Color	Color	默认值: (color 0 0 0)	
		设置材质自发光的颜色。可动画	
<Standard>.specularLevel <Standard>.Specular_Level	Float	5.0	设置高光区的强度。可动画，百分比
<Standard>.glossiness	Float	25.0	设置高光区的尺寸，增大光泽数值，高光区变得小而锐。可动画，百分比
<Standard>.soften <Standard>.Softens_Level	Float	0.1	用于柔化高光的效果，特别对那些由斜光形成的高光。当.specularLevel 属性的值大而属性.glossiness 的值小，用户会在对象表面形成粗糙的背光，这时如果增大属性.soften 的值，可以减轻这种现象。本属性取值范围是 0~1.0。0 表示没有柔化高光的效果，1.0 表示最大程度应用柔化高光的效果。注意：本属性不能用于 Anisotropic 类着色模式。可动画
<Standard>.diffuseLevel <Standard>.Diffuse_Level	Float	100.0	控制材质过渡区的亮度。减小过渡区级别的设置会降低过渡区的亮度，但不会影响高光区域。可动画，百分比
<Standard>.diffuseRoughness <Standard>.Diff_Roughness	Float	50.0	控制材质过渡区与阴影区的融合程度。增加粗糙度，材质变得更加粗糙、灰暗、平面化。可动画，百分比
<Standard>.diffuseLevel <Standard>.Diffuse_Level	Float	100.0	控制材质过渡区的亮度。减小过渡区级别的设置会降低过渡区的亮度，但不会影响高光区域。可动画，百分比
<Standard>.anisotropy	Float	50.0	控制高光区形状的各向异性。设置为 0，高光区是圆形的；设置为 100 高光区变成细长的椭圆形。可动画，百分比
<Standard>.orientation	Float	0.0	指定各向异性高光长轴的方向。可动画，百分比
<Standard>.unused	Integer	1	本属性被定义，但不能用于各种着色模式

(续表)

属性名称	数据类型	默认值	说明
以下属性仅用于 Multi-Layer 类着色模式			
<Standard>.ambient <Standard>.Ambient_Color	Color	默认值: (color 25.5 25.5 25.5)	
		指定对象表面阴影区的色彩。可动画	
<Standard>.diffuse <Standard>.Diffuse_Color	Color	默认值: (color 127.5 127.5 127.5)	
		指定对象表面在最佳光照条件下呈现的本色。可动画	
<Standard>.useSelfIllumColor <Standard>.Use_Self_Illumination	Boolean	False	设置是否为材质指定一个自发光色彩。当设为 On 时, 可以为材质指定一个自发光色彩; 当设为 Off 时, 材质使用过度色作为自发光色彩
<Standard>.selfIllumAmount <Standard>.Self_Illumination	Float	0.0	设置材质自发光的强度。可动画, 百分比
<Standard>.selfIllumColor <Standard>.Self_Illum_Color	Color	默认值: (color 0 0 0)	
		设置材质自发光的颜色。可动画	
<Standard>.diffuseLevel <Standard>.Diffuse_Level	Float	100.0	控制材质过渡区的亮度。减小过渡级别的设置会降低过渡区的亮度, 但不会影响高光区域。可动画, 百分比
<Standard>.diffuseRoughness <Standard>.Diff_Roughness	Float	0.0	控制材质过渡区与阴影区的融合的速度。增加粗糙度, 材质变得更加粗糙、灰暗、平面化。可动画, 百分比
<Standard>.specular <Standard>.Color_1	Color	默认值: (color 229.5 229.5 229.5)	
		控制第一高光层的高光颜色。可动画	
<Standard>.specularLevel <Standard>.Level_1	Float	5.0	影响第一高光层的高光反射的强度, 增加高光级别可以使高光更为明亮。可动画, 百分比
<Standard>.glossiness <Standard>.Glossiness_1.	Float	25.0	影响第一高光层的高光区的尺寸, 增大光泽值, 高光区变得小而锐。可动画, 百分比
<Standard>.anisotropy <Standard>.Anisotropy_1	Float	0.0	控制第一高光层的高光各向异性或形状。当设为 0 时, 高光区为圆形; 当设为 100 时, 高光区变得小而锐。可动画
<Standard>.orientation	Float	0.0	指定第一高光层高光长轴的方向。可动画, 百分比
<Standard>.specular2 <Standard>.Color_2	Color	(color 229.5 229.5 229.5)	影响第二高光层的高光反射的强度, 增加高光级别可以使高光更为明亮。可动画
<Standard>.specularLevel2 <Standard>.Level_2	Float	0.0	影响第二高光层的高光反射的强度, 增加高光级别可以使高光更为明亮。可动画, 百分比
<Standard>.glossiness2 <Standard>.Glossiness_2	Float	25.0	影响第二高光层的高光区的尺寸, 增大光泽值, 高光区变得小而锐。可动画, 百分比

(续表)

属性名称	数据类型	默认值	说明
<Standard>.anisotropy2 <Standard>.Anisotropy_2	Float	0.0	控制第二高光层的高光各向异性或形状。当设置为0时，高光区为圆形；当设置为100时，高光区变得小而锐。可动画
<Standard>.orientation2	Float	0.0	指定第二高光层高光长轴的方向。可动画，百分比
以下属性用于 Strauss 类着色模式			
<Standard>.diffuse <Standard>.Diffuse_Color	Color  127.5 127.5 127.5	(color 127.5 127.5 127.5)	控制材质的颜色。本属性对应其他材质的过渡色，在 Strauss 类着色模式里，用户仅能控制该颜色，Ambient 和 specular 颜色都由着色模式自动计算。可动画
<Standard>.glossiness	Float	25.0	设置高光区的 Glossiness (光泽度)。该数值影响高光区的尺寸，增大光泽数值，高光区变得小而锐。可动画，百分比
<Standard>.metalness	Float	0.0	控制材质的金属效果，金属化的设置越高材质的金属效果越强烈。另外，只有同时增加光泽度的设置金属化参数变化的效果才显著。可动画，百分比

## 材质各类着色模式对应的贴图通道 (一)

	Anisotropic	Blinn	Metal	Multi-Layer
1	AmbientMap	AmbientMap	AmbientMap	AmbientMap
2	DiffuseMap	DiffuseMap	DiffuseMap	DiffuseMap
3	SpecularMap	SpecularMap	SpecularMap	DiffuseLevelMap
4	DiffuseLevelMap	SpecularLevelMap	SpecularLevelMap	DiffuseRoughnessMap
5	SpecularLevelMap	GlossinessMap	GlossinessMap	SpecularMap
6	GlossinessMap	SelfIllumMap	SelfIllumMap	SpecularLevelMap
7	AnisotropyMap	OpacityMap	OpacityMap	GlossinessMap
8	OrientationMap	FilterMap	FilterMap	AnisotropyMap
9	SelfIllumMap	BumpMap	BumpMap	OrientationMap
10	OpacityMap	ReflectionMap	ReflectionMap	specularMap2
11	FilterMap	RefractionMap	RefractionMap	SpecularLevelMap2
12	BumpMap	DisplacementMap	DisplacementMap	GlossinessMap2
13	ReflectionMap			AnisotropyMap2
14	RefractionMap			OrientationMap2
15	DisplacementMap			SelfIllumMap
16				OpacityMap
17				FilterMap
18				BumpMap
19				ReflectionMap
20				RefractionMap
21				DisplacementMap

### 材质各类着色模式对应的贴图通道 (二)

	Oren-Nayar-Blinn	Phong	Strauss	Translucent
1	AmbientMap	AmbientMap	DiffuseMap	AmbientMap
2	DiffuseMap	DiffuseMap	GlossinessMap	DiffuseMap
3	SpecularMap	SpecularMap	MetalnessMap	SpecularMap
4	GlossinessMap	SpecularLevelMap	OpacityMap	GlossinessMap
5	SpecularLevelMap	GlossinessMap	FilterMap	SpecularLevelMap
6	SelfIllumMap	SelfIllumMap	BumpMap	SelfIllumMap
7	OpacityMap	OpacityMap	ReflectionMap	OpacityMap
8	FilterMap	FilterMap	RefractionMap	FilterMap
9	DiffuseLevelMap	BumpMap	DisplacementMap	DiffuseLevel
10	DiffuseRoughnessMap	ReflectionMap		TranslucentColor
11	BumpMap	RefractionMap		BumpMap
12	ReflectionMap	DisplacementMap		ReflectionMap
13	RefractionMap			RefractionMap
14	DisplacementMap			DisplacementMap

### 11.2.14 Shellac: Material (胶合材质)

#### 构造函数

Shellac...

#### 属性

属性名称	数据类型	默认值	说明
<Shellac>.shellacMtl1 <Shellac>.Base_Material	Material	Standard	设置胶合材质中基础材质
<Shellac>.shellacMtl2 <Shellac>.Shellac_Material	Material	Standard	设置胶合材质中胶合子材质
<Shellac>.shellacColorBlend <Shellac>.Shellac_Color_Blend	Float	0.0	控制胶合材质中两种材质的混合量。如果设置为 0.0, 不显示胶合材质的效果。该参没有上限, 数值越大设置胶合材质和基础材质的混合度越高。可动画, 百分比

### 11.2.15 TopBottom: Material (顶/底材质)

#### 构造函数

TopBottom ...

TopBottomMat ...

#### 属性

属性名称	数据类型	默认值	说明
<TopBottom>.topMaterial <TopBottom>.Top_Standard	Material	Standard	设置顶/底材质中顶部子级材质
<TopBottom>.bottomMaterial <TopBottom>.Bottom_Standard	Material	Standard	设置顶/底材质中底部子级材质
<TopBottom>.map1Enabled <TopBottom>.Map_1_Enable	Boolean	True	设置是否使用顶/底材质中顶部子级材质
<TopBottom>.map2Enabled <TopBottom>.Map_2_Enable	Boolean	True	设置是否使用顶/底材质中底部子级材质
<TopBottom>.blend	Float	0.0	设置混合顶/底材质的相交边缘, 这是一个百分比数值, 取值范围在 0~100 之间。如果设置为 0, 在顶材质和底材质之间有一条清晰的分界线; 如果设置为 100, 顶材质和底材质完全混合在一起。可动画
<TopBottom>.position	Float	50.0	指定两种材质的分界线位置, 这是一个百分率数值, 取值范围在 0~100 之间。如果设置为 0, 分界线在对象底部, 只显示顶材质; 如果设置为 100, 分界线在对象的顶部, 只显示底材质。可动画
<TopBottom>.coordinates	Integer	0	设置顶材质和底材质的位置坐标系确定方式: 0: World (指定依据世界坐标系确定顶材质和底材质的位置, 旋转对象时, 顶/底材质不随同旋转) 1: Local (指定依据对象的局部坐标系确定顶材质和底材质的位置, 旋转对象时, 顶/底材质随同旋转)

### 11.3 TextureMap: Material (贴图)

#### 11.3.1 TextureMap 类通用属性和方法

TextureMap 类代表了二维贴图和三维贴图。用户可以把这些贴图赋给材质或某些对象的属性, 还可以创建各种贴图如 Bitmap、Dent 和 Swirl, 并存取其各种属性。

TextureMap 类由 Material 类派生而来, 并继承了 Material 类的所有属性和方法, 这些属性和方法参见本书 11.1 节。

##### 属性

属性名称	数据类型	默认值	说明
<TextureMap>.name	string		设置贴图名称

## 方法

### 1. assignName <TextureMap>

修改指定纹理贴图的名称，以保持其名称的惟一性。修改后的名称格式为“Map #1”，其中数字表示该纹理贴图在场景中的创建序号。

### 2. renderMap <TextureMap> [ into:<bitmap> ] \ [ size:<point2> ] \ [ filename:<string> ] \ [ scale:<Float> ] \ [ filter:<boolean> ] \ [ display:<boolean> ]

本函数可以实现 Material Editor 窗口里的 Render Map 功能，并返回一个 Bitmap 值，包含指定纹理贴图的渲染结果。如果有指定可选参数 into:，将渲染结果放在指定的位图文件里，如果位图文件已存在，位图大小和其他的一些属性与指定位图文件相同，否则系统用参数 size: 和 fileName: 创建一个新的位图文件，参数 size: 的默认值为 [200,200]。

参数 scale: 为应用于三维贴图的一个比例因子。指定三维空间里的表面映射到 UV 平面时的比例因子，实际控制在位图里出现的纹理数量，默认值为 1。

如果参数 filter: 为 True，位图会被过滤。默认值为 False。

如果参数 display: 为 True，渲染结果会用虚拟帧缓存显示出来，默认值为 False。

例如：

```
rm = renderMap $foo.Material.diffuseMap size:[640,480] \
fileName: "foodif.bmp"
save rm
close rm
```

上面脚本将一个贴图进行渲染，并将它存储到一个.bmp 文件里。

## 相关方法

### showTextureMap <Material> <TextureMap> <boolean>

本方法控制指定贴图在着色视窗里的可见性。指定的材质<Material>里必须包含有贴图，参数<TextureMap>指定了要控制的贴图，参数<boolean>控制了贴图是否可见。如：

```
showTextureMap $foo.Material $foo.Material.diffuseMap On
tm = checker()
mat = standardMaterial diffuseMap:tm
mm = multiMaterial()
mm[1] = mat
$Box01.Material = mm
showTextureMap mm[1] tm On
```

注意对 MultiMaterial 类材质，必须明确指定哪一个子材质（比如使用“[ ]”格式指定子材质序号）。

### 11.3.2 TextureMap 的三个共享类

下面的三个 Material 共享类对多个 TextureMap 通用。其对应 Material Editor 窗口里 TextureMap 材质的 Coordinates、Noise 和 Output 卷展栏下的控件。这些类不能由 MAXScript 构造，而是由 TextureMap 自动生成。这三个 Material 共享类分别为：

UVGenClass

StandardXYZGen

TextOutputClass

下面分节讨论 TextureMap 的这三个共享类。

### 11.3.3 UVGenClass: Material

#### 属性

属性名称	数据类型	默认值	说明
<textureMap.coordinates>.mappingType	Integer	0	设置二维贴图的方式： 0: Texture (纹理贴图); 1: Environment (环境贴图) 如果指定贴图的 Coordinates 卷展栏正被显示在 Material Editor 窗口，这时用户改变本属性的值，Texture/Environment radiobutton 控件和 Mapping dropdown 控件的显示值不会发生改变
<textureMap.coordinates>.mapping	Integer	varies	依据纹理贴图方式指定贴图坐标类型。如果贴图方式为纹理贴图，默认值为 0；可以选择以下几种坐标类型： 0: Explicit Map Channel (外部贴图通道) 1: Vertex Color Channel (节点色彩通道) 2: Planar from Object XYZ (基于对象 XYZ 的平面) 3: Planar from World XYZ (基于世界 XYZ 的平面) 如果贴图方式为环境贴图，默认值为 4；可以选择以下几种坐标类型： 0: Spherical (球体环境贴图) 1: Cylindrical (圆柱体环境贴图) 2: Shrink-Wrap (收缩包裹环境贴图) 3: Screen (在场景中以平面背景方式投影贴图) 每一种贴图类型的贴图源文件在系统内部是分开存储的，如果用户将某一贴图类型的贴图源文件改变，不会影响其他贴图类型的贴图源文件
<textureMap.coordinates>.mapChannel	Integer	1	指定贴图的 UV 坐标系
<textureMap.coordinates>.U_Offset	Float	0.0	设置贴图在贴图的 UV 坐标系中沿 U 轴方向的偏移量。可动画

(续表)

属性名称	数据类型	默认值	说明
<textureMap.coordinates>.V_Offset	Float	0.0	设置贴图在贴图的UV坐标系中沿V轴方向的偏移量。可动画
<textureMap.coordinates>.U_Tiling	Float	1.0	设置贴图在贴图的UV坐标系中沿U轴方向的重复拼接次数。可动画
<textureMap.coordinates>.V_Tiling	Float	1.0	设置贴图在贴图的UV坐标系中沿V轴方向的重复拼接次数。可动画
<textureMap.coordinates>.U_Angle	Float	0.0	设置贴图在贴图的UV坐标系中沿U轴方向的旋转角度。可动画, Angle
<textureMap.coordinates>.V_Angle	Float	0.0	设置贴图在贴图的UV坐标系中沿V轴方向的旋转角度。可动画, Angle
<textureMap.coordinates>.W_Angle	Float	0.0	设置贴图在贴图的UV坐标系中沿W轴方向的旋转角度。可动画, Angle
<textureMap.coordinates>.Blur	Float	1.0	依据贴图距离视图的距离对其进行虚化处理, 距离越远虚化程度越高。可动画, Angle
<textureMap.coordinates>.Blur_Offset	Float	0.0	对贴图进行虚化处理, 与贴图距离视图的距离无关, 利用该参数可以对贴图进行抗锯齿或散焦处理。可动画, Angle
<textureMap.coordinates>.Phase	Float	0.0	设置噪波的动画速度。可动画
<textureMap.coordinates>.Noise_Amount	Float	1.0	设置噪波的分形函数计算的强度。该参数为百分率类型, 如果设置为0, 不产生噪波效果; 如果设置为100, 贴图呈现纯噪波的效果。可动画
<textureMap.coordinates>.Noise_Levels	Integer	1	设置噪波的分形函数计算的的次数。级别参数受Noise_Amount的影响, Noise_Amount越大, Noise_Levels参数的作用效果越强烈。可动画
<textureMap.coordinates>.Noise_Size	Float	1.0	指定噪波函数相对于几何对象的比例关系, 数值越大噪波越平缓; 数值越小噪波越细碎。可动画
<textureMap.coordinates>.showMapOnBack	Boolean	True	设置是否显示背面贴图
<textureMap.coordinates>.U_Mirror	Boolean	False	设置贴图是否沿U轴方向重复镜像
<textureMap.coordinates>.V_Mirror	Boolean	False	设置贴图是否沿V轴方向重复镜像
<textureMap.coordinates>.U_Tile	Boolean	True	设置贴图是否沿U轴方向重复拼接
<textureMap.coordinates>.V_Tile	Boolean	True	设置贴图是否沿V轴方向重复拼接
<textureMap.coordinates>.UVW_Type	Integer	0	设置贴图坐标类型:0: UV; 1: VW; 2: WU
<textureMap.coordinates>.Noise_On	Boolean	False	噪波开关
<textureMap.coordinates>.Noise_Animate	Boolean	False	噪波动画开关

### 11.3.4 StandardXYZGen: Material

属性

属性名称	数据类型	默认值	说明
<textureMap.coords>.coordType	Integer	0	设置贴图坐标类型: 0: Object XYZ (使用基于对象 Local 坐标的平面贴图) 1: World XYZ (使用基于场景 World 坐标的平面贴图) 2: Explicit Map Channel (使用由属性 mapChannel 指定的贴图通道) 3: Vertex Color Channel (使用节点色彩通道)
<textureMap.coords>.mapChannel	Integer	1	当属性.coordType 为 2 时, 为贴图坐标指定贴图通道
<textureMap.coords>.offset	Point3	[0,0,0]	设置贴图在贴图的 UV 坐标系中的偏移量。贴图的移动受其自身尺寸的影响。可动画
<textureMap.coords>.Tiling	Point3	[1,1,1]	设置贴图沿各自轴向上的重复拼接次数。可动画
<textureMap.coords>.angle	Point3	[0,0,0]	设置贴图沿 U、V、W 轴方向的旋转角度。单位为度。可动画, Angle
<textureMap.coords>.blur	Float	1.0	依据贴图距离视图的距离对其进行虚化处理, 距离越远虚化程度越高。可动画
<textureMap.coords>.Blur_Offset	Float	0.0	对贴图进行虚化处理, 与贴图距离视图的距离无关, 利用该参数可以对贴图进行抗锯齿或散焦处理。可动画

### 11.3.5 TextOutputClass: Material

属性

属性名称	数据类型	默认值	说明
<texturemap.output>.Output_Amount	Float	1.0	控制贴图被混合到合成材质的程度, 影响贴图的饱和度和 Alpha 通道。可动画
<texturemap.output>.RGB_Offset	Float	0.0	在贴图的 RGB 色彩上加入指定的偏移量, 可以使整体变亮或变暗。可动画
<texturemap.output>.RGB_Level	Float	1.0	依据指定的参数倍增贴图的 RGB 色彩数值, 影响贴图的饱和度。增大参数设置, 使贴图色彩向高饱和、自发光的色彩方向转变; 减小参数设置, 使贴图色彩向低饱和、灰度图像方向转变。可动画
<texturemap.output>.Bump_Amount	Float	1.0	调整凹凸的深度, 该参数只作用于 Bump 贴图。可动画
<texturemap.output>.Mono_Color_Map	subAnim		包含贴图曲线的单色, 用户只有先在用户界面里勾选复选框 Enable Color Map 后才可以在 MAXScript 里创建、存取这个曲线

(续表)

属性名称	数据类型	默认值	说明
<texturemap.output>.RGB_Color_Map	subAnim		包含贴图曲线的 RGB 颜色。用户只有先在用户界面里勾选复选框 Enable Color Map 并选择 RGB radiobutton 后才可以在 MAXScript 里创建、存取这个曲线
<texturemap.output>.Mono_Color_Map>.curve_1	subAnim		curve_1 包含单色贴图曲线的控制点。用户不能创建曲线上的点，且在这些点被设置动画后才可以被存取。当这些点被设置动画后，用户可以用曲线的属性.Point_X 来存取点的位置，其中 X 表示点编号
<texturemap.output>.RGB_Color_Map>.curve_1 <texturemap.output>.RGB_Color_Map>.curve_2 <texturemap.output>.RGB_Color_Map>.curve_3	SubAnim		curve_1、curve_2、curve_3 分别对应贴图的 R、G、B 色彩图形曲线。用户不能创建曲线上的点，且在这些点被设置动画后才可以被存取。当这些点被设置动画后，用户可以用曲线的属性.Point_X 来存取点的位置，其中 X 表示点编号
<texturemap.output>.invert	Boolean	False	控制是否反转贴图的色调，使之类似彩色照片的底片
<texturemap.output>.clamp	Boolean	False	当设置为 On 时，参数会将颜色的值限制于不超过 1.0。当增加 RGB 级别时启用此选项，但此贴图不会显示出自发光。注意：如果在启用限制时将 RGB 偏移的值设置超过 1.0，所有的颜色都会变成白色
<texturemap.output>.alphaFromRGB	Boolean	False	当设置为 On 时，会根据在贴图中 RGB 通道的强度生成一个 Alpha 通道。黑色变得透明而白色变得不透明。中间值根据其强度变得半透明

## 11.4 贴图类型

下面列出了所有 3ds max 里的贴图类型：

- ◆ Adobe\_Photoshop\_Plug\_In\_Filter
- ◆ Adobe\_Premiere\_Video\_Filter
- ◆ BitmapTexture (位图贴图)
- ◆ Bricks (砖块贴图)
- ◆ Cellular (细胞贴图)
- ◆ Checker (方格贴图)
- ◆ Combustion (燃烧贴图)
- ◆ Composite (合成贴图)
- ◆ Dent (凹痕贴图)

- ◆ Falloff (衰减贴图)
- ◆ FalloffTextureMap (衰减纹理贴图)
- ◆ FlatMirror (平面镜贴图)
- ◆ Gradient (渐变贴图)
- ◆ Gradient\_Ramp (渐变坡度贴图)
- ◆ Marble (大理石贴图)
- ◆ Mask (遮罩贴图)
- ◆ Mix (混合贴图)
- ◆ Noise (躁波贴图)
- ◆ NoTexture (无纹理贴图)
- ◆ Output (输出贴图)
- ◆ Paint (绘制贴图)
- ◆ Particle\_Age (粒子年龄贴图)
- ◆ Particle\_Mblur (粒子运动模糊贴图)
- ◆ Perlin\_Marble (Perlin 大理石贴图)
- ◆ Planet (行星贴图)
- ◆ Raytrace (光线跟踪贴图)
- ◆ Reflect\_Refraet (反射和折射贴图)
- ◆ RGB\_Multiply (RGB 倍增贴图)
- ◆ RGB\_Tint (RGB 色彩贴图)
- ◆ Smoke (烟雾贴图)
- ◆ Speckle (斑点贴图)
- ◆ Splat (泼溅贴图)
- ◆ Stucco (灰泥贴图)
- ◆ Swirl (旋涡贴图)
- ◆ Thin\_Wall\_Refraction (薄壁折射贴图)
- ◆ Vertex\_Color (顶点颜色贴图)
- ◆ Water (波浪贴图)
- ◆ Wood (木材贴图)

### 11.4.1 Adobe\_Photoshop\_Plug\_In\_Filter: TextureMap

#### 构造函数

Adobe\_Photoshop\_Plug\_In\_Filter ...

#### 属性

属性名称	数据类型	默认值	说明
<Adobe_Photoshop_Plug_In_Filter>.blur	Float	0.0	可动画, 百分比
<Adobe_Photoshop_Plug_In_Filter>.Foreground_Color	Color	(color 0 0 0)	指定当前图像的前景色, 有些 Photoshop 外挂要使用该前景色。可动画
<Adobe_Photoshop_Plug_In_Filter>.Background_Color	Color	(color 0 0 0)	指定当前图像的背景色, 有些 Photoshop 外挂要使用该背景色。可动画
<Adobe_Photoshop_Plug_In_Filter>.coordinates	SubAnim		参见 UVGenClass
<Adobe_Photoshop_Plug_In_Filter>.output	StandardTextureOutput		参见 TextOutputClass

**注意** 在 3ds max 的 MAXScript 里不能存取界面的以下控件: Parameters 卷展栏里的 Use Alpha Planecheckbox 控件和 Bitmap Parameters 和 Time Parameters 卷展栏里的所有控件。

#### 11.4.2 Adobe\_Premiere\_Video\_Filter: TextureMap

##### 构造函数

Adobe\_Premiere\_Video\_Filter ...

##### 属性

属性名称	数据类型	说明
<Adobe_Premiere_Video_Filter>.coordinates	SubAnim	参见 UVGenClass
<Adobe_Premiere_Video_Filter>.output	StandardTextureOutput	参见 TextOutputClass

**注意** 在 3ds max 的 MAXScript 里不能存取界面 Time Parameters 卷展栏里的所有控件。

#### 11.4.3 BitmapTexture: TextureMap (位图贴图)

##### 构造函数

BitmapTexture ...

Bitmaptex ...

##### 属性

属性名称	数据类型	默认值	说明
<Bitmaptexture>.bitmap	Bitmap	Bitmap	返回贴图的位图文件。在脚本里可以对这个位图文件执行任何位图操作，并且这些修改会马上反映到贴图上。只读
<Bitmaptexture>.filename <Bitmaptexture>.file_name	String	“”	指定位图文件的文件名
<Bitmaptexture>.filtering	Integer	0	指定对位图进行抗锯齿处理的方式： 0: Pyramidal (可以创建渲染速度较快的较好的抗锯齿效果) 1: Summed Area (占用大量内存资源，但抗锯齿效果很好) 2: None (取消过滤处理)
<Bitmaptexture>.monoOutput	Integer	0	一些参数如 opacity(不透明度)、specular level (高光级别) 的贴图需要一个单独的通道(灰通道图像)进行控制，而不是 RGB 三个通道。该参数指定将 RGB 彩色贴图文件转换为图单通道灰度图像的方式： 0: RGB Intensity (使用位图文件红、绿、蓝三个通道的强度影响贴图效果，位图的色彩信息被忽略，只有像素的透明度数值产生作用，最后获得的结果是 0 (黑) ~255 (白) 灰度渐变级别) 1: Alpha (位图文件的 Alpha 通道强度影响贴图效果)
<Bitmaptexture>.RGBOutput	Integer	0	一些参数如 Ambient、Diffuse、Specular、Filter Color、Reflection、Refraction 的贴图需要 RGB 彩色贴图文件，该参数指定位图色彩输出的方式： 0: RGB (显示像素的全部色彩数值) 1: Alpha as Gray (基于位图 Alpha 通道的级别显示灰度色调)
<Bitmaptexture>.apply	Boolean	False	是否使用剪切/定位设置
<Bitmaptexture>.cropPlace	Integer	0	Crop/Place 的互斥开关
<Bitmaptexture>.clipu <Bitmaptexture>.clip_u_offset	Float	0.0	调整贴图沿 U 轴方向的位置。可动画
<Bitmaptexture>.clipv <Bitmaptexture>.clip_v_offset	Float	0.0	调整贴图沿 V 轴方向的位置。可动画
<Bitmaptexture>.clipw <Bitmaptexture>.clip_u_width	Float	1.0	调整位图图像的宽度。可动画
<Bitmaptexture>.cliph <Bitmaptexture>.clip_v_width	Float	1.0	调整位图图像的高度。可动画
<Bitmaptexture>.jitter <Bitmaptexture>.jitter_placement	Float	1.0	指定随机偏移的数量，如果指定为 0 不发生随机偏移。可动画

(续表)

属性名称	数据类型	默认值	说明		
<Bitmaptexture>.alphasource	Integer	0	设置Alpha通道的来源: 0: Image Alpha(使用图像自身的Alpha通道) 1: RGB Intensity(将位图的色彩转换为灰度色调数值,黑色区域完全透明;白色区域完全不透明) 2: None(不使用透明通道)		
<Bitmaptexture>.preMultAlpha	Boolean	True	指定如何处理位图中的Alpha通道。当设置为On时,Alpha通道进行自左乘处理;反之,Alpha通道不进行自左乘处理,所有RGB数值都被忽略		
<Bitmaptexture>.starttime	Time	0f	指定动画贴图开始回放的帧		
<Bitmaptexture>.playbackrate	Float	1.0	指定动画贴图回放的速率。指定为1.0时,以正常速率回放动画贴图;指定为2.0时,以双倍速率快速回放动画贴图;指定为0.333时,以1/3速率慢速回放动画贴图		
<Bitmaptexture>.endcondition	Integer	0	指定当回放到动画最后一帧时,如何继续显示该动画贴图: 0: Loop(返回到第一帧继续重复回放动画贴图向回放动画贴图,以此方式循环往复) 1: Ping Pong(从最后一帧向前反向回放动画贴图,以此方式循环往复) 2: Hold(将最后一帧保持为静态图像直到场景动画结束为止)		
<Bitmaptexture>.coords	StandardUVGen	参见UVGenClass			
<Bitmaptexture>.output	数据类型: StandardTextureOutput				
	说明: 参见TextOutputClass				

## 方法

1. <Bitmaptexture>.reload()  
重新装载位图文件。
2. <Bitmaptexture>.viewImage()  
查看源图像。

## 相关方法

1. enumerateFiles [<MAXWrapper\_obj>] <function> [<arg>] \ [ #inactive ] [ #videoPost ] [ #render ] [ #missing ] \ [ #localOnly ]

让用户对场景里或指定单个对象里用到的所有位图文件进行循环,也可以对位图文件进行过滤。

2. freeSceneBitmaps()

释放所有由位图缓存占用的内存。如果内存被许多位图文件分割成许多碎片，而用户又想将当前活动的位图文件重载，本方法非常有用。

### 3. usedMaps()

返回一个由当前场景所有使用过的贴图文件名组成的数组。

#### 11.4.4 Bricks: TextureMap (砖块贴图)

##### 构造函数

Bricks ...

##### 属性

属性名称	数据类型	默认值	说明
<Bricks>.Brick_Type	Integer		设置砖块类型： 0: Custom Brick 1: Running Bond 2: Common Flemish Bond 3: English Bond 4: 1/2 Running Bond 5: Stack Bond 6: Fine Running Bond 7: Fine Stack Bond  本属性预设砖块堆叠效果和行列编辑属性。 改变.brick_type 属性的值只改变这些属性(如果 Material Editor 窗口当前被打开，且本材质为活动材质)
<Bricks>.Show_Texture_Swatches	Integer	1	如果为 On，刷新 Material Editor 以显示贴图纹理。0: Off; 1: On
<Bricks>.Brick_Color	Color	默认值: (color 165.75 76.5 51)	设置砖块颜色。可动画
<Bricks>.Horizontal_Count	Float	3.0	设置每行砖块数目。可动画
<Bricks>.Vertical_Count	Float	8.0	设置每列砖块数目。可动画
<Bricks>.Color_Variance	Float	0.4	设置砖块颜色变化量。可动画
<Bricks>.Fade_Variance	Float	0.2	设置砖块褪色变化量。可动画
<Bricks>.Mortar_Color	Color	默认值: (color 211.65 196.35 183.6)	设置灰泥颜色。可动画
<Bricks>.Horizontal_Gap	Float	1.0	砖块之间水平灰泥宽度。可动画
<Bricks>.Vertical_Gap	Float	1.0	砖块之间垂直灰泥宽度。可动画
<Bricks>.Lock_Gap_Symmetry	Integer	1	如果设置为 On 时，属性.Horizontal_Gap 和.Vertical_Gap 互相锁定，改变其中一个，另一个的值会跟着改变。0: Off; 1: On

(续表)

属性名称	数据类型	默认值	说明
<Bricks>.Holes	Integer	0	砖块开洞比例，洞口处没有砖块而灰泥仍然穿过洞口。可动画
<Bricks>.Edge_Roughness	Float	0.0	定义灰泥边缘的粗糙程度。可动画
<Bricks>.Random_Seed	Integer	43304	贴图图案颜色变化量，只要调整本属性的值，可以生成完全不同的图案。可动画
<Bricks>.Line_Shift	Float	0.0	砖块隔行错开的距离。可动画
<Bricks>.Random_Shift	Float	0.0	砖块每行之间随机错开的距离。可动画
<Bricks>.Use_Row_Edit	Integer	0	开关行编辑：0: Off; 1: On
<Bricks>.Per_Row	Integer	2	设置每行里的砖块数
<Bricks>.Change_Row	Float	1.0	系统存储的每行里的砖块数
<Bricks>.Use_Column_Edit	Integer	0	开关列编辑：0: Off; 1: On
<Bricks>.Per_Column	Float	1.0	每列里的砖块数
<Bricks>.Change_Column	Float	1.0	系统存储的每行里的砖块数
<Bricks>.coordinates	SubAnim		参见 UVGenClass

#### 11.4.5 Cellular: TextureMap (细胞贴图)

##### 构造函数

Cellular ...

##### 属性

属性名称	数据类型	默认值	说明
<Cellular>.celcolor <Cellular>.Cell_Color	Color	(color 255 255 255)	设置细胞的颜色。可动画
<Cellular>.cellmap	textureMap	undefined	设置细胞的贴图，而不使用单一的颜色。可动画
<Cellular>.map1Enabled <Cellular>.Map1_On	Boolean	True	设置是否使用细胞的贴图
<Cellular>.variation	Float	0.0	为细胞颜色指定随机变化，数值越高随机效果越明显，取值范围从 0~100 之间。可动画
<Cellular>.divcolor1 <Cellular>.Division_Color1	Color	默认值：(color 127.5 127.5 127.5) 指定细胞侧壁的色彩。可动画	指定细胞侧壁的贴图
<Cellular>.divmap1 <Cellular>.DivisionMap1	textureMap		
<Cellular>.map2Enabled <Cellular>.Map2_On	Boolean	True	设置是否使用细胞侧壁的贴图

(续表)

属性名称	数据类型	默认值	说明
<Cellular>.divcolor2 <Cellular>.Division_Color2	Color	(color 0 0 0)	指定细胞间隙的色彩。可动画
<Cellular>.divmap2 <Cellular>.DivisionMap2	textureMap	undefined	指定细胞间隙的贴图
<Cellular>.map3Enabled <Cellular>.Map3_On	Boolean	True	设置是否使用细胞间隙的贴图
<Cellular>.type	Integer	0	设置细胞的形状和尺寸： 0: Circular (创建有机、饱满、泡状的细胞效果) 2: Chips (创建直线边界、碎片、马赛克的细胞效果)
<Cellular>.size	Float	5.0	指定整个细胞增殖贴图的尺寸，应将其指定到几何体对象相同的大小。可动画
<Cellular>.spread	Float	0.5	设置单个细胞的大小。可动画
<Cellular>.smooth <Cellular>.Bump_smoothing	Float	0.1	当细胞增殖贴图作为凹凸贴图时，该参数可以避免凹凸边界的锯齿、毛边效果。可动画
<Cellular>.fractal	Boolean	False	当设置为 On 时，创建分形化的细胞增殖贴图效果
<Cellular>.iteration <Cellular>.Iterations	Float	3.0	指定进行分形计算的重复次数。较低的设置可以产生更为平滑的效果。可动画
<Cellular>.adaptive	Boolean	True	当设置为 On 时，自动调整重复计算的次数，减少图像的锯齿和渲染时间
<Cellular>.roughness	Float	0.0	如果用户使用 Cellular 贴图作为凹凸贴图，本属性设置凹凸的粗糙度。取值范围为[0.0 1.0]，如果设为 0，每一次重复次数的强度和尺寸为上一次的一半。本属性数值越大，每一次重复次数的强度和尺寸与上一次越接近。当取最大值 1.0 时，每一次重复次数的强度和尺寸与上一次相等。本属性只有在属性.iteration 的值大于 1.0 时才有效。可动画
<Cellular>.lowthresh <Cellular>.Low	Float	0.0	控制细胞和间壁的相对尺寸。可动画
<Cellular>.midthresh <Cellular>.Mid	Float	0.5	控制细胞侧壁的尺寸。可动画
<Cellular>.highthresh <Cellular>.High	Float	1.0	控制整个细胞间隙的尺寸。可动画
<Cellular>.cords	StandardXYZGen		参见 StandardXYZGen
<Cellular>.output	StandardTextureOutput		参见 TextOutputClass

### 11.4.6 Checker: TextureMap (方格贴图)

#### 构造函数

Checker ...

#### 属性

属性名称	数据类型	默认值	说明
<Checker>.soften	Float	0.0	柔化方格拼接边缘。可动画
<Checker>.color1 <Checker>.Color_1	Color	(color 0 0 0)	指定方格构成元素的一个色彩。可动画
<Checker>.map1 <Checker>.Map_1	textureMap	undefined	指定方格构成元素的一个贴图
<Checker>.map1Enabled <Checker>.Map_1_Enable	Boolean	True	设置是否使用方格构成元素的一个贴图
<Checker>.color2 <Checker>.Color_2	Color	默认值: (color 255 255 255) 指定方格构成元素的另一个色彩。可动画	
<Checker>.map2 <Checker>.Map_2	textureMap	undefined	指定方格构成元素的另一个贴图
<Checker>.map2Enabled <Checker>.Map_2_Enable	Boolean	True	设置是否使用方格构成元素的另一个贴图
<Checker>.cords	StandardUVGen	参见 StandardXYZGen	

### 11.4.7 CompositeTextureMap: TextureMap (合成贴图)

#### 构造函数

CompositeTextureMap ...

CompositeTexture ...

#### 属性

属性名称	数据类型	默认值	说明
<CompositeTexturemap>.mapList <CompositeTexturemap>.maps	数据类型: ArrayParameter 默认值: #(undefined, undefined)		存储贴图的所有子级贴图
<CompositeTexturemap>.mapEnabled <CompositeTexturemap>.Map_1_Enable	数据类型: ArrayParameter 默认值: #(True, True)		存储贴图是否使用每一个子级贴图

#### 注意

上面两个数组初始长度都为 2, 对应 CompositeTextureMap 贴图的两个子级贴图。  
如果要改变数组的长度, 可直接将这两个数组的.count 属性设置成指定值, 比如下面的例子创建了一个包含 5 个子级贴图的 CompositeTexture Map 贴图。

```
c=compositeTextureMap()
c.mapList.count=5
```

### 11.4.8 Dent: TextureMap (凹痕贴图)

#### 构造函数

Dent ...

#### 属性

属性名称	数据类型	默认值	说明
<Dent>.size	Float	200.0	设置凹痕的相对尺寸，降低尺寸可以增加凹痕的数量。可动画
<Dent>.strength	Float	20.0	设置凹痕的数量，数值越大凹痕越密集，数值越小凹痕越稀疏。取值范围在 0~100 之间。可动画
<Dent>.iterations	Integer	2	设置迭代计算的重复次数，数值越大凹痕越密集，对象表面越复杂，耗费的渲染时间也越多。可动画
<Dent>.color1 <Dent>.Color_1	Color	(color 0 0 0)	设置凹痕的第一个颜色。可动画
<Dent>.map1	textureMap	undefined	设置凹痕的第一个贴图
<Dent>.map1Enabled <Dent>.MapOn1	Boolean	True	设置是否使用凹痕的第一个贴图
<Dent>.color2 <Dent>.Color_2	Color	默认值: (color 255 255 255)	
		设置凹痕的第二个颜色。可动画	
<Dent>.map2	textureMap	undefined	设置凹痕的第二个贴图
<Dent>.map2Enabled <Dent>.MapOn2	Boolean	True	设置是否使用凹痕的第二个贴图
<Dent>.cords	StandardXYZGen	参见 StandardXYZGen	

### 11.4.9 Falloff: TextureMap (衰减贴图)

#### 构造函数

Falloff ...

#### 属性

属性名称	数据类型	默认值	说明
<Falloff>.color1 <Falloff>.Color_1	Color	(color 0 0 0)	设置衰减的第一个颜色。可动画
<Falloff>.map1Amount <Falloff>.Map_Amount_1	Float	100.0	设置衰减贴图的第一个颜色相对强度。可动画
<Falloff>.map1 <Falloff>.Map_1	textureMap	undefined	设置衰减的第一个贴图

(续表)

属性名称	数据类型	默认值	说明
<Falloff>.map1On <Falloff>.Map_1_Enable	Boolean	True	当设置为 On 时, 使用衰减的第一个贴图; 否则, 使用衰减的第一个颜色
<Falloff>.color2 <Falloff>.Color_2	Color	(color 255 255 255)	设置衰减的第二个颜色。可动画
<Falloff>.map2Amoun <Falloff>.Map_Amount_2	Float	100.0	设置衰减贴图的第二个贴图相对强度。可动画
<Falloff>.map2 <Falloff>.Map_1	textureMap	undefined	设置衰减的第二个贴图
<Falloff>.map2On <Falloff>.Map_2_Enable	Boolean	True	当设置为 On 时, 使用衰减的第二个贴图; 否则, 使用衰减的第二个颜色
<Falloff>.type <Falloff>.Falloff_Type	Integer	1	设置衰减的类型。可动画 0: Towards/Away (依据对象面法线方向与当前视图呈 0 到 180 度角指定衰减变化的范围) 1: Perpendicular/Parallel (依据对象面法线方向与当前视图呈 0 到 90 度角指定衰减变化的范围) 2: Fresnel (依据对象的折射率参数指定衰减变化的范围。面对视图的表面暗化, 与视图呈 45 度夹角的表面亮化) 3: Light/Shadowed (依据对象表面的受光变化, 指定衰减变化的范围) 4: Distance Blend (依据远距与近距参数, 指定衰减变化的范围)
<Falloff>.mtlIOROverride <Falloff>.Mtl_IOR_Override_Enable	Boolean	True	当设为 On 时, 使用当前指定的折射率替代材质原先的折射率
<Falloff>.ior <Falloff>.Index_of_Refraction	Float	1.6	重新指定材质的折射率。可动画
<Falloff>.nearDistance <Falloff>.Near_Distance	Float	0.0	指定衰减开始的位置。可动画
<Falloff>.farDistance <Falloff>.Far_Distance	Float	100.0	指定衰减结束的位置。可动画
<Falloff>.extrapolateOn <Falloff>.Distance_Bind_Extrapolate	Boolean	False	设置是否依据远距与近距的参数设置, 进行渐变融合。本属性仅适用于 Distance Blend 类衰减

(续表)

属性名称	数据类型	默认值	说明
<Falloff>.direction <Falloff>.Falloff_Direction	Integer	0	设置衰减方向。可动画 0: Viewing Direction (依据摄影机的方向指定衰减变化, 改变对象的方向不影响衰减贴图) 1: Object (以指定的对象的局部坐标方向指定衰减变化, 改变对象的方向, 衰减效果随同改变) 2: Local X Axis/ Local Y Axis /Local Z Axis (依据贴图材质被指定到对象的局部坐标 X/Y/Z 轴方向作为衰减变化的方向, 改变对象的方向, 衰减效果随同改变) 3: World X Axis/ World Y Axis/ World Z Axis (依据场景世界坐标的 X/Y/Z 轴的方向指定衰减变化, 改变对象的方向不影响衰减贴图)
<Falloff>.Node <Falloff>.Falloff_Direction_Object	Node	undefined	当属性.Falloff_Direction = 1 时, 选择场景对象作为衰减变化的方向
<Falloff>.texture_output	StandardTextureOutput		参见 TextureOutputClass
<Falloff>.mixcurve	SubAnim	SubAnim:MixCurve	设置控制衰减灰度渐变的混合曲线
<Falloff.mixcurve>.curve_1	SubAnim	SubAnim:Curve_X	包含了混合曲线的各控制点。用户不能创建曲线上的点, 且在这些点被设置动画后才可以被存取。当这些点被设置动画后, 用户可以用曲线的属性.Point_X 来存取点的位置, 其中 X 表示点编号

#### 11.4.10 FalloffTextureMap: TextureMap (衰减纹理贴图)

falloffTextureMap 类贴图用来将 3ds max R2.5 里的 Falloff 类贴图转换到 3ds max R4 及以后更高版本中的 Falloff 类贴图, 在 MAXScript 里不能创建 FalloffTextureMap 类贴图。

##### 属性

属性名称	数据类型
<fallofftextureMap>.perpendicularValue <fallofftextureMap>.Perpendicular_Value	Float
<fallofftextureMap>.parallelValue <fallofftextureMap>.Parallel_Value	Float
<fallofftextureMap>.direction	Integer
<fallofftextureMap>.node	node

### 11.4.11 FlatMirror: TextureMap (平面镜贴图)

#### 构造函数

FlatMirror ...

Flat\_Mirror ...

#### 属性

属性名称	数据类型	默认值	说明
<FlatMirror>.applyBlur	Boolean	True	是否使用贴图虚化效果，同时也打开抗锯齿的效果（如果有设置的话）
<FlatMirror>.blurAmount	Float	1.0	依据被发射对象距离镜面的远近，指定发射虚化的强度，该参数还可以获得抗锯齿的效果。可动画
<FlatMirror>.frame	Integer	1	当设置为 0 时，仅在第一帧创建镜面自动反射；当设置为 1 时，依据指定的帧间隔，每隔指定帧数进行一次自动镜面反射计算
<FlatMirror>.nthFrame	Integer	1	当属性.frame =1 时，设置每隔指定帧数进行一次自动镜面反射计算
<FlatMirror>.useEnviroment	Boolean	True	设置镜面反射是否自动对环境贴图进行反射计算
<FlatMirror>.applyToFaceID	Boolean	False	当设为 On 时，依据对象表面的材质 ID 号码为面指定镜面反射贴图
<FlatMirror>.faceID	Integer	1	当属性.applyToFaceID=True 时，为需要镜面反射的面指定材质 ID 号码
<FlatMirror>.distortionType	Integer	0	设置模拟镜面不平整造成的反射扭曲效果： 0: None (不产生扭曲) 1: Use Bump Map (使用材质的凹凸贴图产生扭曲反射的效果) 2: Use Built-in Noise (依据在 noise 项目中的参数设置产生扭曲反射的效果)
<FlatMirror>.noiseType	Integer	0	当属性.distortionType=2 时，设置噪波的类型： 0: Regular (创建规则的噪波效果) 1: Fractal (创建以分形方式计算的噪波) 2: Turbulence (创建更为混乱的噪波效果)
<FlatMirror>.distortionAmount	Float	0.5	设置镜面反射的扭曲强度。这是惟一可以影响扭曲强度的属性。可动画
<FlatMirror>.phase	Float	0.0	设置噪波的相位数值，控制噪波函数的动画速度。单位时间内相位变化的数值越大，动画噪波的速度越快。可动画
<FlatMirror>.size	Float	10.0	设置噪波函数相对于几何对象的比例关系。数值越小噪波越细碎。可动画
<FlatMirror>.level	Float	2.0	设置分形函数重复计算的次数。可动画

## 11.4.12 Gradient: TextureMap (渐变贴图)

## 构造函数

Gradient ...

## 属性

属性名称	数据类型	默认值	说明
<Gradient>.color1 <Gradient>.Color_1	Color	(color 0 0 0)	指定渐变色三个构成元素的第一个颜色。可动画
<Gradient>.map1 <Gradient>.Map_1	TextureMap	undefined	指定渐变色三个构成元素的第一个贴图
<Gradient>.map1Enabled <Gradient>.Map_1_Enable	Boolean	True	设置是否使用第一个贴图
<Gradient>.color2 <Gradient>.Color_2	Color	默认值: (color 127.5 127.5 127.5) 指定渐变色三个构成元素的第二个颜色。可动画	
<Gradient>.map2 <Gradient>.Map_2	TextureMap	undefined	指定渐变色三个构成元素的第二个贴图
<Gradient>.map2Enabled <Gradient>.Map_2_Enable	Boolean	True	设置是否使用第二个贴图
<Gradient>.color3 <Gradient>.Color_3	Color	(color 255 255 255)	指定渐变色三个构成元素的第三个颜色
<Gradient>.map3 <Gradient>.Map_3	TextureMap	undefined	指定渐变色三个构成元素的第三个贴图
<Gradient>.map3Enabled <Gradient>.Map_3_Enable	Boolean	True	设置是否使用第三个贴图
<Gradient>.color2Pos <Gradient>.Color_2_Position	Float	0.5	控制中间色彩的位置, 取值范围在 0 到 1 之间。指定为 0, 色彩 2 将取代色彩 3; 设置为 1, 色彩 2 将取代色彩 1
<Gradient>.gradientType <Gradient>.Gradient_Type	Integer	0	设置渐变类型: 0: Linear (直线性的色彩渐变效果) 1: Radial (从中心向外的放射状色彩渐变效果)
<Gradient>.noiseAmount <Gradient>.Noise_Amount	Float	0.0	设置噪波计算的强度。取值范围为 [0.0 1.0], 如果指定为 0 不产生噪波效果; 如果指定为 1 贴图呈现纯噪波的效果
<Gradient>.noiseType <Gradient>.Noise_Type	Integer	0	设置噪波的类型: 0: Regular (创建规则的噪波效果) 1: Fractal (创建以分形方式计算的噪波) 2: Turbulence (创建更为混乱的噪波效果)

(续表)

属性名称	数据类型	默认值	说明
<Gradient>.noiseSize <Gradient>.Noise_Size	Float	1.0	设置噪波函数相对于几何对象的比例关系。数值越小噪波越细碎
<Gradient>.noisePhase <Gradient>.Noise_Phase	Float	0.0	设置噪波的相位数值。单位时间内相位变化的数值越大，动画噪波的速度越快
<Gradient>.noiseLevels <Gradient>.Noise_Levels	Float	4.0	设置分形函数重复计算的次数
<Gradient>.noiseThresholdLow <Gradient>.Low_Threshold	Float	0.0	指定噪波的最低阈值
<Gradient>.noiseThresholdHigh <Gradient>.High_Threshold	Float	1.0	指定噪波的最高阈值
<Gradient>.NoiseThresholdSmooth <Gradient>.Threshold_Smoothing	Float	0.0	依据参数阈值对进行光滑处理。如果设置为 0，不进行光滑处理。取值范围在 0~1 之间
<Gradient>.cords	StandardUVGen		参见 UVGenClass
<Gradient>.output	StandardTextureOutput		参见 TextOutputClass

#### 11.4.13 Gradient\_Ramp: TextureMap (渐变坡度贴图)

##### 构造函数

Gradient\_Ramp ...

##### 属性

属性名称	数据类型	默认值	说明
<Gradient_Ramp>.Gradient_Type	Integer	4	设置渐变类型： 0: 4_Corner (不对称线性颜色渐变) 1: Box 2: Diagonal (线性、对角线颜色渐变) 3: Lighting (光线密度值颜色渐变) 4: Linear (平滑线性对角线颜色渐变) 5: Mapped (光线亮度值重映射渐变) 6: Normal (摄影机对象连线与采样点表面法线之间角度颜色渐变) 7: Pong 8: Radial (从中心向外的放射状色彩渐变效果) 9: Spiral (平滑、螺旋形颜色渐变) 10: Sweep 11: Tartan (格子状颜色渐变)

(续表)

属性名称	数据类型	默认值	说明
<Gradient_Ramp>.Source_Map_On	Integer	1	当设置为 On 时, 使用渐变贴图
<Gradient_Ramp>.Noise_Type	Integer	0	设置 noise 的类型: 0: Regular (规则的噪波效果) 1: Fractal (分形方式计算的噪波) 2: Turbulence (更为混乱的噪波效果)
<Gradient_Ramp>.amount	Float	0.0	设置噪波计算的强度, 取值范围为[0.0 1.0]。如果指定为 0 不产生噪波效果; 如果指定为 1 贴图呈现纯噪波的效果。可动画
<Gradient_Ramp>.size	Float	1.0	设置噪波函数相对于几何对象的比例关系。数值越小噪波越细碎。可动画
<Gradient_Ramp>.phase	Float	0.0	设置噪波的相位数值。单位时间内相位变化的数值越大, 动画噪波的速度越快。可动画
<Gradient_Ramp>.Levels	Float	4.0	设置分形函数重复计算的次数。可动画
<Gradient_Ramp>.Low_Threshold	Float	0.0	指定噪波的最低阈值。可动画
<Gradient_Ramp>.High_Threshold	Float	1.0	指定噪波的最高阈值。可动画
<Gradient_Ramp>.Threshold_Smoothing	Float	0.0	依据参数阈值对进行光滑处理。如果设置为 0, 不进行光滑处理。取值范围在 0~1 之间。可动画
<Gradient_Ramp>.coordinates	SubAnim		参见 UVGenClass
<Gradient_Ramp>.output	数据类型: StandardTextureOutput 说明: 参见 TextOutputClass		
<Gradient_Ramp>.gradient_ramp	SubAnim		
<Gradient_Ramp.gradient_ramp>.flag_1	SubAnim		
<Gradient_Ramp.gradient_ramp>.flag_2	SubAnim		
<Gradient_Ramp.gradient_ramp>.flag_3	SubAnim		
<Gradient_Ramp.gradient_ramp.flag_1>.color	Point3	[255,0,0]	可动画
<Gradient_Ramp.gradient_ramp.flag_2>.color	Point3	[0,0,255]	可动画
<Gradient_Ramp.gradient_ramp.flag_3>.color	Point3	[0,255,0]	可动画
<Gradient_Ramp.gradient_ramp.flag_3>.position	Float	[0,255,0]	可动画

### 11.4.14 Marble: TextureMap (大理石贴图)

#### 构造函数

Marble ...

#### 属性

属性名称	数据类型	默认值	说明
<Marble>.size	Float	70	设置纹理的间距。可动画
<Marble>.width <Marble>.Vein_width	Float	0.025	设置纹理的宽度。可动画
<Marble>.color1 <Marble>.Color_1	Color	默认值: (color 51 51 25.5)	
		设置大理石贴图的纹理颜色。可动画	
<Marble>.map1 <Marble>.Map_1	TextureMap	undefined	设置大理石的纹理贴图
<Marble>.map1Enabled <Marble>.Map_1_Enable	Boolean	True	设置是否使用大理石的纹理贴图
<Marble>.color2 <Marble>.Color_2	Color	默认值: (color 209.1 209.1 153)	
		设置大理石贴图的背景颜色。可动画	
<Marble>.map2 <Marble>.Map_2	TextureMap	undefined	设置大理石的背景贴图
<Marble>.map2Enabled <Marble>.Map_2_Enable	Boolean	True	设置是否使用大理石的背景贴图
<Marble>.cords	StandardXYZGen		参见 StandardXYZGen

### 11.4.15 Mask: TextureMap (遮罩贴图)

#### 构造函数

Mask ...

MaskTex ...

#### 属性

属性名称	数据类型	默认值	说明
<Mask>.map	TextureMap	undefined	指定在遮罩图像的透明区域显示的贴图
<Mask>.mapEnabled <Mask>.Map_1_Enable	Boolean	True	设置是否使用在遮罩图像的透明区域显示的贴图
<Mask>.mask	TextureMap	undefined	指定遮罩贴图
<Mask>.maskEnabled <Mask>.Map_2_Enable	Boolean	True	设置是否使用遮罩贴图
<Mask>.maskInverted <Mask>.Invert	Boolean	False	当设为 On 时, 反转遮罩设置负片的效果

## 11.4.16 Mix: TextureMap (混合贴图)

## 构造函数

Mix ...

MixTextures ...

## 属性

属性名称	数据类型	默认值	说明
<Mix>.color1 <Mix>.Color_1	Color	(color 0 0 0)	设置混合贴图的第一个颜色。可动画
<Mix>.map1 <Mix>.Map_1	TextureMap	undefined	指定混合贴图的第一个贴图
<Mix>.map1Enabled <Mix>.Map_1_Enable	Boolean	True	设置是否使用混合贴图的第一个贴图
<Mix>.color2 <Mix>.Color_2	Color	(color 255 255 255)	设置混合贴图的第二个颜色。可动画
<Mix>.map2 <Mix>.Map_2	TextureMap	undefined	指定混合贴图的第二个贴图
<Mix>.map2Enabled <Mix>.Map_2_Enable	Boolean	True	设置是否使用混合贴图的第二个贴图
<Mix>.mixAmount	Float	0.0	设置两种混合色彩或贴图的呈现比例。 设置为 0 只显示 color1 的色彩或贴图； 设置为 1 只显示 color2 的色彩或贴图。 可动画，百分比
<Mix>.mask	TextureMap	undefined	使用混合量贴图的明度数值控制两种 混合色彩或贴图的呈现比例，而不使用 属性.mixAmount 来控制两种混合色彩 或贴图的呈现比例
<Mix>.maskEnabled <Mix>.MaskEnable	Boolean	True	设置是否使用混合量贴图
<Mix>.useCurve	Boolean	False	设置是否使用混合量曲线来控制两种 混合色彩或贴图在混合过程中的渐变 效果
<Mix>.lower	Float	0.3	设置混合区域的下限级别。可动画
<Mix>.upper	Float	0.7	设置混合区域的上限级别。如果上限与 下限相同，两个材质相交于一条清晰的 边界；上下限两个参数相差越大，创建 的渐变效果越好。可动画
<Mix>.output	StandardTextureOutput		参见 TextureOutputClass

## 11.4.17 Noise: TextureMap (躁波贴图)

## 构造函数

Noise ...

## 属性

属性名称	数据类型	默认值	说明
<Noise>.type	Integer	0	设置噪波的类型: 0: Regular (创建规则的噪波效果) 1: Fractal (创建以分形方式计算的噪波) 2: Turbulence (创建更为混乱的噪波效果)
<Noise>.size <Noise>.Noise_Size	Float	25.0	设置噪波函数相对于几何对象的比例关系。可动画
<Noise>.thresholdLow <Noise>.Low_Threshold	Float	0.0	指定噪波的最低阈值。可动画
<Noise>.thresholdHigh <Noise>.High_Threshold	Float	1.0	指定噪波的最高阈值。可动画
<Noise>.levels <Noise>.Noise_Levels	Float	3.0	设置分形函数重复计算的次数。可动画
<Noise>.phase	Float	0.0	设置噪波的相位数值。单位时间内相位变化的数值越大，动画噪波的速度越快。可动画
<Noise>.color1 <Noise>.Color_1	Color	默认值: (color 0 0 0)	
		设置噪波的第一个主要颜色。可动画	
<Noise>.map1 <Noise>.Map_1	Texture-Map	undefin-ed	设置替代噪波的第一个颜色的贴图
<Noise>.map1Enabled <Noise>.Map_1_Enable	Boolean	True	设置是否使用噪波的第一个贴图
<Noise>.color2 <Noise>.Color_2	Color	默认值: (color 255 255 255)	
		设置噪波的第二个主要颜色。可动画	
<Noise>.map2 <Noise>.Map_2	Texture-Map	undefin-ed	设置替代噪波的第二个颜色的贴图
<Noise>.map2Enabled <Noise>.Map_2_Enable	Boolean	True	设置是否使用噪波的第二个贴图
<Noise>.coords	StandardXYZGen	参见 StandardXYZGen	
<Noise>.output	StandardTextureOutput	参见 TextOutputClass	

## 11.4.18 NoTexture: TextureMap (无纹理贴图)

给对象赋给 NoTexture 类贴图相当于在 Material Editor 窗口里将贴图设为 None，也相当于给对象的贴图属性设为 undefined。如

```
$Box01.Material.bumpmap=noTexture()
```

或

```
$Box01.Material.bumpmap=undefined
```

#### 构造函数

NoTexture ...

#### 属性

NoTexture 没有额外的属性。

### 11.4.19 Output: TextureMap (输出贴图)

#### 构造函数

Output ...

#### 属性

属性名称	数据类型	默认值	说明
<output>.map1 <output>.map_1	TextureMap	undefined	设置要应用 output 的贴图
<output>.map1Enabled <output>.map_1_Enabled	Boolean	True	设置是否使用属性.map1 指定的贴图
<output>.output	StandardTextureOutput		参见 TextOutputClass

### 11.4.20 Paint: TextureMap (绘制贴图)

#### 构造函数

Paint ...

#### 属性

属性名称	数据类型	默认值	说明
<Paint>.height	Integer	0	可动画
<Paint>.width	Integer	0	可动画
<Paint>.coordinates	SubAnim		参见 UVGenClass

### 11.4.21 Particle\_Age: TextureMap (粒子年龄贴图)

#### 构造函数

Particle\_Age ...

#### 属性

属性名称	数据类型	默认值	说明
<Particle_Age>.color1 <Particle_Age>.Color_1	Color	(color 0 0 0)	设置粒子诞生时的颜色。可动画
<Particle_Age>.map1 <Particle_Age>.Map_1	TextureMap	undefined	设置粒子诞生时的贴图
<Particle_Age>. map1Enabled <Particle_Age>. Map_1_Enable	Boolean	True	设置是否使用粒子诞生时的贴图
<Particle_Age>.age1 <Particle_Age>.Age_1	Float	0.0	设置粒子颜色开始从 color1 开始变到 color2 时所处的周期位置。为粒子整个生存周期的百分比。可动画
<Particle_Age>.color2 <Particle_Age>.Color_2	Color	(color 127.5 127.5 127.5)	设置粒子运动到周期中点时的颜色。可动画
<Particle_Age>.map2 <Particle_Age>.Map_2	TextureMap	undefined	设置粒子运动到周期中点时贴图
<Particle_Age>. map2Enabled <Particle_Age>. Map_2_Enable	Boolean	True	设置是否使用粒子运动到周期中点时的贴图
<Particle_Age>.age2 <Particle_Age>.Age_2	Float	50.0	设置粒子颜色为 color2 时所处的周期位置。为粒子整个生存周期的百分比。可动画
<Particle_Age>.color3 <Particle_Age>.Color_3	Color	(color 255 255 255)	设置粒子运动到周期终点时的颜色。可动画
<Particle_Age>.map3 <Particle_Age>.Map_3	TextureMap	undefined	设置粒子运动到周期终点时贴图
<Particle_Age>. map3Enabled <Particle_Age>. Map_3_Enable	Boolean	True	设置是否使用粒子运动到周期终点时的贴图
<Particle_Age>.age3 <Particle_Age>.Age_3	Float	100.0	设置粒子颜色为 color3 时所处的周期位置。可动画
<Particle_Age>.output	StandardTextureOutput		参见 TextureOutputClass

#### 11.4.22 Particle\_MBlur: TextureMap (粒子运动模糊贴图)

构造函数

Particle\_MBlur ...

ParticleBlur ...

## 属性

属性名称	数据类型	默认值	说明
<Particle_MBlur>.color1 <Particle_Mblur>.color_1	Color	(color 255 255 255)	设置当粒子达到其最慢速度时的颜色，默认值为白色。可动画
<Particle_MBlur>.color2 <Particle_Mblur>.color_2	Color	(color 0 0 0)	设置当粒子达速度加快时的颜色，默认值为黑色。可动画
<Particle_MBlur>.sharp <Particle_Mblur>.sharpness	Float	2.0	控制粒子相对于速度的透明度。如果设为 0，全部粒子为模糊、透明的，不管它的运动速度有多慢

### 11.4.23 Perlin\_Marble: TextureMap (Perlin 大理石贴图)

#### 构造函数

Perlin\_Marble ...

PerlinMarble ...

#### 属性

属性名称	数据类型	默认值	说明
<Perlin_Marble>.size	Float	50.0	设置大理石纹理的尺寸。可动画
<Perlin_Marble>.level <Perlin_Marble>.Levels	Float	8.0	设置紊乱计算的重复次数，取值范围在 1.0~10.0 之间，数值越高创建的大理石图案越复杂。可动画
<Perlin_Marble>.color1 <Perlin_Marble>.Color_1	Color	默认值: (color 189.975 189.975 160.65)	设置大理石的第一个颜色。可动画
<Perlin_Marble>.saturation1 <Perlin_Marble>.Color_1_Saturation	Float	85.0	控制大理石的第一个颜色在贴图中的饱和度，不会影响色彩样本中的色彩，取值范围为 1 到 100。可动画
<Perlin_Marble>.map1	TextureMap	undefined	设置替代大理石的第一个颜色的贴图
<Perlin_Marble>.map1Enabled <Perlin_Marble>.EnableMap1	Boolean	True	设置是否使用大理石的第一个贴图
<Perlin_Marble>.color2 <Perlin_Marble>.Color_2	Color	默认值: (color 59.925 89.25 59.925)	设置大理石的第二个颜色。可动画
<Perlin_Marble>.saturation2 <Perlin_Marble>.Color_2_Saturation	Float	70.0	控制大理石的第二个颜色在贴图中的饱和度，不会影响色彩样本中的色彩。可动画
<Perlin_Marble>.map2	TextureMap	undefined	设置替代大理石的第二个颜色的贴图
<Perlin_Marble>.map2Enabled <Perlin_Marble>.EnableMap2	Boolean	True	设置是否使用大理石的第二个贴图
<Perlin_Marble>.coords	StandardXYZGen		参见 StandardXYZGen

## 11.4.24 Planet: TextureMap (行星贴图)

## 构造函数

Planet ...

## 属性

属性名称	数据类型	默认值	说明
<Planet>.color1 <Planet>.Color_1	Color	(color 10.2 20.4 79.05)	设置水域中心的色彩。可动画
<Planet>.color2 <Planet>.Color_2	Color	(color 10.2 30.6 79.05)	设置环绕水域中心色彩 color1 的色彩。可动画
<Planet>.color3 <Planet>.Color_3	Color	(color 10.2 40.8 79.05)	设置环绕 color2 的色彩并与陆地色彩相接。可动画
<Planet>.color4 <Planet>.Color_4	Color	(color 10.2 99.45 12.75)	这四个颜色设置了陆地的颜色，紧接着水域颜色，并依次连续排列：Color4 设置陆地海岸线的色彩，与水域相接；Color5 紧接 Color4，朝向陆地中心；Color8 为陆地中心的色彩。可动画
<Planet>.color5 <Planet>.Color_5	Color	(color 99.45 79.05 12.75)	
<Planet>.color6 <Planet>.Color_6	Color	(color 79.05 20.4 7.65)	
<Planet>.color7 <Planet>.Color_7	Color	(color 99.45 79.05 51)	
<Planet>.color8 <Planet>.Color_8	Color	(color 99.45 99.45 99.45)	
<Planet>.continentSize <Planet>.Continent_Size	Float	40.0	设置陆地分形噪波图案的尺寸，取值范围在 0~100 之间。较高的数值可以获得较大的陆地面积。可动画
<Planet>.islandFactor <Planet>.Island_Factor	Float	0.5	设置岛屿和高山分形噪波图案的尺寸，取值范围在 0~100 之间。较高的数值可以获得较崎岖的地形。可动画
<Planet>.oceanPercent <Planet>.Ocean_Percent	Float	60.0	设置陆地表面海洋面积所占的比率。可动画
<Planet>.randomSeed	Integer	12345	设置行星表面纹理图案的随机种子数，可以在不改变其他参数设置的情况下，变化行星表面的纹理效果
<Planet>.blendWaterLand	Boolean	True	设置是否融合水域与陆地色彩的边界。如果设为 True，水域与陆地之间呈现一种朦胧的效果，否则水域与陆地之间有一条明显的界线
<Planet>.cords	StandardXYZGen		参见 StandardXYZGen

## 11.4.25 Raytrace: TextureMap (光线跟踪贴图)

## 构造函数

Raytrace...

## 属性

属性名称	数据类型	默认值	说明
<Raytrace.parameters>.Trace_Mode	Integer	0	设置进行光线反射或折射的方式: 0: Auto Detect (自动对光线跟踪反射或折射进行计算。如果光线跟踪贴图作为材质的反射贴图，则进行反射计算；如果光线跟踪贴图作为材质的折射贴图，则进行折射计算) 1: Reflection (进行反射计算) 2: Refraction (进行折射计算)
<Raytrace.parameters>.Options_Raytracer_Enable	Boolean	True	开/关光线跟踪。即使关闭光线跟踪，光线跟踪材质和贴图仍然会反射和折射环境，包括3ds max 场景的环境贴图和赋给线跟踪材质的环境贴图。可动画
<Raytrace.parameters>.Options_Antialiasing_Enable	Boolean	True	开/关抗锯齿。可动画
<Raytrace.parameters>.Options_Self_Reflect_Refra	Boolean	True	开/关自反射和折射。可动画
<Raytrace.parameters>.Options_Raytrace_Atmospherics	Boolean	True	开/关大气效果光线跟踪。大气效果包括火、雾和体积灯光等。可动画
<Raytrace.parameters>.Options_Refra_Refra_Material_ID_s	Boolean	True	开/关反射和折射材质 ID 号码。可动画
<Raytrace.parameters>.Options_Raytrace_Objects_in_Glass	Boolean	True	开/关光线跟踪对象的内部对象的光线跟踪设置。可动画
<Raytrace.parameters>.Options_Raytrace_Atmospherics_in_Glass	Boolean	True	开/关光线跟踪对象的内部大气效果的光线跟踪设置。可动画
<Raytrace.parameters>.Options_Color_Density_Fog_Enable	Boolean	True	开/关密度雾的光线跟踪设置。可动画
<Raytrace.parameters>.Background_Mode	Integer	0	设置背景模式: 0: 使用当前场景的环境背景设置 1: 指定用于光线跟踪的环境背景色彩 2: 指定用于光线跟踪的环境背景贴图
<Raytrace.parameters>.Background_Color	Color	默认值: (color 0 0 0) 设置用于光线跟踪的环境背景色彩。可动画	当设置为 On 时，用户可以修改局部抗锯齿设置 当设置为 On 时，激活光线跟踪贴图的适配抗锯齿设置
<Raytrace.parameters>.Local_Antialias_OVERRIDE	Boolean		
<Raytrace.parameters>.Adaptive_Antialiasing	Boolean	False	

(续表)

属性名称	数据类型	默认值	说明
<Raytrace.parameters>.Local_Threshold	Float	0.1	设置适配计算的灵敏度, 取值范围为0~1。设置为0时总是计算最大光线数量; 设置为1时计算初始光线数量。可动画
<Raytrace.parameters>.Local_Min_Rays	Integer	4	设置每个像素进行光线计算的初始数量。可动画
<Raytrace.parameters>.Local_Max_Rays	Integer	32	设置每个像素进行光线计算的最大数量。可动画
<Raytrace.parameters>.Local_Blr_Offset	Float	0.0	对光线跟踪反射和折射结果进行虚化。虚化过程不考虑场景的远近, 数据设置基于像素。可动画
<Raytrace.parameters>.Local_Blr_Aspect	Float	1.0	通过改变图像纵横比的方式进行虚化处理。通常不需要改变该比率。可动画
<Raytrace.parameters>.Local_Defocus_Amount	Float	0.0	散焦是基于场景距离的虚化, 靠近表面的对象不进行虚化处理; 远离表面的对象进行虚化处理。可动画
<Raytrace.parameters>.Local_Defocus_Aspect	Float	1.0	通过改变图像纵横比的方式进行散焦处理。通常不需要改变该比率。可动画
<Raytrace.parameters>.Use_Blr_Map	Boolean	False	当设为On时, 使用指定的贴图确定虚化偏移的数量, 贴图白色的部位创建完全的虚化偏移效果; 贴图黑色的部分不进行虚化偏移处理
<Raytrace.parameters>.Use_Defocus_Map	Boolean	False	当设为On时, 使用指定的贴图确定散焦处理的数量, 贴图白色的部位创建完全的散焦处理效果; 贴图黑色的部分不进行散焦处理处理
<Raytrace.parameters>.Attenuation_Mode	Integer	0	设置光线跟踪的衰减模式: 0: Off (关闭衰减控制) 1: Linear (在start和end参数之间进行线性衰减计算) 2: Inverse Scale (使用start参数进行反转平方衰减计算) 3: Exponential (在start和end参数之间进行指数衰减计算) 4: Custom Falloff (依据自定义的衰减曲线进行衰减计算)
<Raytrace.parameters>.Attenuation_Start	Float	0.0	设置衰减开始的距离。可动画
<Raytrace.parameters>.Attenuation_End	Float	100.0	设置衰减结束的距离。可动画
<Raytrace.parameters>.Attenuation_Exponent	Float	2.0	设置从衰减开始到结束之间进行指数衰减计算。可动画
<Raytrace.parameters>.Attenuation_Color_Mode	Integer	0	指定光线跟踪在最后衰减至消失时的状态: 0: Background 1: Attenuation_Color (使用指定的光线跟踪最后衰减至消失时的色彩)

(续表)

属性名称	数据类型	默认值	说明
<Raytrace.parameters>.Attenuation_Color	Color	默认值: (color 0 0 0) 设置光线跟踪最后衰减至消失时的色彩。可动画	
<Raytrace.parameters>.Attenuation_Near	Float	1.0	指定光线跟踪在衰减开始距离中反射/折射的强度。取值范围在0.0~1.0之间。可动画
<Raytrace.parameters>.Attenuation_Control_1	Float	0.6666	指定曲线在开始位置的状态。可动画
<Raytrace.parameters>.Attenuation_Control_2	Float	0.3333	指定曲线在结束位置的状态。可动画
<Raytrace.parameters>.Attenuation_Far	Float	0.0	指定光线跟踪在衰减结束距离中反射/折射的强度，取值范围在0.0~1.0之间。可动画
<Raytrace.parameters>.Use_Reflectivity_Map	Boolean	False	设置是否使用反射率贴图。可动画
<Raytrace.parameters>.Reflectivity_Map_Amount	Float	1.0	控制光线跟踪贴图在材质中的作用。可动画
<Raytrace.parameters>.Basic_Tint_Enable	Boolean	False	设置是否使用基本染色处理
<Raytrace.parameters>.Tint_Amount	Float	1.0	设置染色处理的强度。可动画
<Raytrace.parameters>.Tint_Color	Color	默认值: (color 255 255 255) 指定反射的染色色彩。可动画	
<Raytrace.parameters>.Use_Tint_Map	Boolean	False	设置是否使用贴图控制染色的强度
<Raytrace.parameters>.Color_Density_Enable	Boolean	False	设置是否使用色彩密度
<Raytrace.parameters>.Color_Density_Color	Color	默认值: (color 255 255 255) 指定透明色彩。可动画	
<Raytrace.parameters>.Color_Density_Amount	Float	1.0	指定色彩密度的强度。取值范围在0~1.0之间。可动画
<Raytrace.parameters>.Color_Density_Start	Float	0.0	设置密度色彩的开始距离。可动画
<Raytrace.parameters>.Color_Density_End	Float	25.0	指定达到最大密度色彩时的对象厚度。只有对象的厚度大于开始距离参数，才呈现透明色彩。可动画
<Raytrace.parameters>.Use_Color_Density_Map	Boolean	False	设置是否使用贴图控制密度色彩的强度
<Raytrace.parameters>.Fog_Enable	Boolean	False	设置是否使用密度雾
<Raytrace.parameters>.Fog_Color	Color	默认值: (color 255 255 255) 设置密度雾的颜色。可动画	

(续表)

属性名称	数据类型	默认值	说明
<Raytrace.parameters>.Fog_Amount	Float	1.0	设置密度雾的强度，取值范围在 0~1.0 之间。减小该参数可以减小密度雾的效果，使其更为透明。可动画
<Raytrace.parameters>.Fog_Start	Float	0.0	设置密度雾的开始位置。可动画
<Raytrace.parameters>.Fog_End	Float	25.0	指定达到最大密度雾时的对象厚度。只有对象的厚度大于开始距离参数，才呈现密度雾。可动画
<Raytrace.parameters>.Use_Fog_Map	Boolean	False	设置是否使用贴图控制密度雾的强度
<Raytrace.parameters>.Use_Reflectivity_Map	Boolean	False	设置是否使用光线跟踪贴图。可动画
<Raytrace.parameters>.Bump_Multiplier	Float	0.2	设置凹凸乘法器数值
<Raytrace.parameters>.Filter_Density_Map_Amount	Float	1.0	设置过滤器密度贴图的数量值
<Raytrace.parameters>.Fog_Map_Amount	Float	1.0	设置雾贴图的数量值
<Raytrace.parameters>.Refractions_as_Glass_Enable	Boolean	True	设置是否使用玻璃折射效果
<Raytrace.parameters>.Tint_Map_Amount	Float	1.0	设置色彩贴图的数量值
下面属性不与任何界面控件对应。			
<Raytrace.parameters>.Interobject_Interphase_Fusing	Boolean	False	
<Raytrace.parameters>.IIF_Tolerance	Float	1.0	
<Raytrace.parameters>.Options__Color_Density__Fog_Enable	Boolean	True	可动画

**注意** 可以赋给 Raytrace 类贴图的子级贴图不能被 3ds max 的 MAXScript 存取，除非这些子级贴图被赋给 Bricks 类贴图。

#### 11.4.26 Reflect\_Refraet: TextureMap (反射和折射贴图)

##### 构造函数

```
Reflect_Refraet ...
ReflectRefraet ...
```

##### 属性

属性名称	数据类型	默认值	说明
<Reflect_Refraet>.source	Integer	0	指定六面贴图的来源: 0: Automatic (从对象的轴心点自动向场景六个方向创建六面贴图) 1: From File (为六面贴图指定图像文件)
<Reflect_Refraet>.size	Integer	100	指定反射/折射贴图的尺寸。数值越小，反射/折射的图像越不清楚。可动画
<Reflect_Refraet>.UseAtmosphericMap <Reflect_Refraet>.Use_Atmospheric	Boolean	True	设置是否对场景环境进行反射/折射的计算
<Reflect_Refraet>.apply	Boolean	True	设置是否对贴图进行虚化处理
<Reflect_Refraet>.blurOffset <Reflect_Refraet>.Blur_Offset	Float	0.0	对反射/折射结果进行虚化，虚化过程中不考虑场景的远近，数据设置基于像素。可动画
<Reflect_Refraet>.blur	Float	1.0	依据被反射/折射对象距离当前对象的远近，指定反射/折射虚化的强度，利用该参数还可以获得抗锯齿的处理效果。可动画
<Reflect_Refraet>.near <Reflect_Refraet>.Near_Value	Float	0.0	指定雾的近距范围。可动画
<Reflect_Refraet>.far <Reflect_Refraet>.Far_Value	Float	500.0	指定雾的远距范围。可动画
<Reflect_Refraet>.frametype	Integer	0	设置自动反射/折射计算的帧类型： 0: First Frame Only, 仅在第一帧进行自动反射/折射计算 1: Every Nth Frame, 依据指定的帧间隔，每隔指定帧数进行一次自动反射/折射计算
<Reflect_Refraet>.nthframe	Integer	1	当属性.frametype=1 时，为自动贴图指定帧间隔数
<Reflect_Refraet>.bitmapName	数据类型: ArrayParameter 默认值: #(undefined, ..., undefined) 为包含 6 个元素的数组，每个元素为源贴图文件的文件名字符串		
<Reflect_Refraet>.outputname <Reflect_Refraet>.Output_Name	String	undefined	

#### 11.4.27 RGB\_Multiply: TextureMap (RGB 倍增贴图)

构造函数

RGB\_Multiply ...

RGBMult ...

### 属性

属性名称	数据类型	默认值	说明
<RGB_Multiply>.color1 <RGB_Multiply>.Color_1	Color	默认值: (color 255 255 255)	
<RGB_Multiply>.map1 <RGB_Multiply>.Map_1		可动画	
<RGB_Multiply>.map1Enabled <RGB_Multiply>.Map_1_Enable	Boolean	True	设置是否使用第一个贴图
<RGB_Multiply>.color2 <RGB_Multiply>.Color_2	Color	(color 255 255 255)	可动画
<RGB_Multiply>.map2 <RGB_Multiply>.Map_2	TextureMap	undefined	
<RGB_Multiply>.map2Enabled <RGB_Multiply>.Map_2_Enable	Boolean	True	设置是否使用第二个贴图
<RGB_Multiply>.alphaFrom	Integer	2	设置贴图的 Alpha 通道产生方式: 0: Map #1 (使用第一个贴图的 Alpha 通道) 1: Map #2 (使用第二个贴图的 Alpha 通道) 2: Multiply Alphas (将两个贴图的 Alpha 通道相乘以生成一个新的 Alpha 通道)

### 11.4.28 RGB\_Tint: TextureMap (RGB 色彩贴图)

#### 构造函数

RGB\_Tint ...

RGBTint ...

#### 属性

属性名称	数据类型	默认值	说明
<RGB_Tint>.red	Color	(color 255 0 0)	图像的红色通道。可动画
<RGB_Tint>.green	Color	(color 0 255 0)	图像的绿色通道。可动画
<RGB_Tint>.blue	Color	(color 0 0 255)	图像的蓝色通道。可动画
<RGB_Tint>.map1 <RGB_Tint>.Map_1	TextureMap	undefined	指定被进行染色处理的贴图
<RGB_Tint>.map1Enabled <RGB_Tint>.Map_1_Enable	Boolean	True	设置是否使用染色贴图

## 11.4.29 Smoke: TextureMap (烟雾贴图)

## 构造函数

Smoke ...

## 属性

属性名称	数据类型	默认值	说明
<Smoke>.size	Float	40.0	设置烟雾贴图的雾团尺寸。可动画
<Smoke>.iterations	Integer	5	设置分形迭代计算的重复次数，数值越高产生越多的烟雾细节，耗费的渲染输出时间也越长。可动画
<Smoke>.phase	Float	0.0	指定紊乱烟雾图案的初始相位。为相位的变化指定动画，可以获得动态的烟雾效果。可动画
<Smoke>.exponent	Float	1.5	增大该参数，可以使 color2 所代表的烟雾纹理更为纤细、清晰。可动画
<Smoke>.color1 <Smoke>.Color_1	Color	(color 0 0 0)	设置无烟部分的颜色。可动画
<Smoke>.map1	TextureMap	undefined	为效果的无烟部分指定贴图
<Smoke>.map1On <Smoke>.Map1_On	Boolean	True	设置是否为效果的无烟部分使用贴图
<Smoke>.color2 <Smoke>.Color_2	Color	默认值: (color 229.5 229.5 229.5) 设置烟的色彩。可动画	
<Smoke>.map2	TextureMap	undefined	指定烟的贴图
<Smoke>.map2On <Smoke>.Map2_On	Boolean	True	设置是否为效果的烟使用贴图
<Smoke>.coords	StandardXYZGen		参见 StandardXYZGen

## 11.4.30 Speckle: TextureMap (斑纹贴图)

## 构造函数

Speckle ...

## 属性

属性名称	数据类型	默认值	说明
<Speckle>.size	Float	60.0	指定斑纹贴图的斑纹尺寸。可动画
<Speckle>.color1 <Speckle>.Color_1	Color	(color 51 127.5 255)	设置斑纹的色彩。可动画
<Speckle>.map1	TextureMap	undefined	为斑纹的色彩指定替代贴图
<Speckle>.map1On <Speckle>.Map1_On	Boolean	True	设置是否使用为斑纹指定的关联贴图

(续表)

属性名称	数据类型	默认值	说明
<Speckle>.color2 <Speckle>.Color_2	Color	(color 178.5 204 204)	指定斑纹贴图的背景色彩。可动画
<Speckle>.map2	TextureMap	undefined	为斑纹贴图的背景色彩指定替代贴图
<Speckle>.map2On <Speckle>.Map2_On	Boolean	True	设置是否使用为背景指定的关联贴图
<Speckle>.coords	StandardXYZGen		参见 StandardXYZGen

#### 11.4.31 Splat: TextureMap (泼溅贴图)

##### 构造函数

Splat ...

##### 属性

属性名称	数据类型	默认值	说明
<Splat>.size	Float	40.0	指定 splat 斑点尺寸。可动画
<Splat>.iterations	Integer	4	设置分形迭代计算的重复次数，数值越高产生越多的斑点细节，耗费的渲染输出时间也越长。可动画
<Splat>.threshold	Float	0.2	指定 color1 与 color2 的混合程度。如果设置为 0，仅显示 color1；如果设置为 1，仅显示 color2；可动画
<Splat>.color1 <Splat>.Color_1	Color	默认值：(color 178.5 204 204) 指定 splat 的背景色彩。可动画	
<Splat>.map1	TextureMap	undefined	为斑点贴图的背景色彩指定替代贴图
<Splat>.map1On <Splat>.Map1_On	Boolean	True	设置是否使用为背景指定的关联贴图
<Splat>.color2 <Splat>.Color_2	Color	默认值：(color 51 127.5 255) 指定 splat 的斑点色彩。可动画	
<Splat>.map2	TextureMap	undefined	为 splat 的斑点色彩指定替代贴图
<Splat>.map2On <Splat>.Map2_On	Boolean	True	设置是否使用为斑点色彩指定的关联贴图
<Splat>.coords	StandardXYZGen		参见 StandardXYZGen

#### 11.4.32 Stucco: TextureMap (灰泥贴图)

##### 构造函数

Stucco ...

### 属性

属性名称	数据类型	默认值	说明
<Stucco>.size	Float	20.0	指定灰泥贴图的灰泥斑点尺寸。可动画
<Stucco>.thickness	Float	0.15	设置两种颜色之间边界的虚化程度，取值范围为[0.0 1.0]，如果为0，两种颜色之间边界非常清晰；取值越大，两种颜色之间边界越模糊。如果将灰泥贴图作为凹凸贴图，将本属性设为0.5，两种颜色之间边界线非常模糊，如果再大一点，边界线会消失。可动画
<Stucco>.threshold	Float	0.57	指定color1与color2的混合程度，如果设置为0，仅显示color1；如果设置为1，仅显示color2；可动画
<Stucco>.color1 <Stucco>.Color_1	Color	(color 0 0 0)	指定灰泥贴图的灰泥斑点色彩。可动画
<Stucco>.map1	TextureMap	undefined	指定灰泥斑点色彩的替代贴图
<Stucco>.map1On <Stucco>.Map1_On	Boolean	True	设置是否使用灰泥斑点色彩的关联贴图
<Stucco>.color2 <Stucco>.Color_2	Color	(color 229.5 229.5 229.5)	指定灰泥贴图的底色色彩。可动画
<Stucco>.map2	TextureMap	undefined	指定灰泥贴图的底色色彩的替代贴图
<Stucco>.map2On <Stucco>.Map2_On	Boolean	True	设置是否使用底色色彩的替代贴图
<Stucco>.cords	StandardXYZGen		参见 StandardXYZGen

### 11.4.33 Swirl: TextureMap (旋涡贴图)

#### 构造函数

Swirl...

#### 属性

属性名称	数据类型	默认值	说明
<Swirl>.Base	Color	(color 229.5 147.9 76.5)	指定旋涡贴图的基础色彩。可动画
<Swirl>.Swirl	Color	(color 0 25.5 51)	指定旋涡贴图的旋涡色彩。可动画
<Swirl>.Color_Contrast	Float	0.4	控制基础与旋涡的色彩对比度。如果设置为0，旋转元素被进行虚化处理，取值范围在0~4.0之间。可动画
<Swirl>.Swirl_Intensity	Float	2.0	控制旋涡贴图的波动混合强度。如果设置为0，旋涡效果消失。取值范围为[-10 10]。可动画
<Swirl>.Swirl_Amount	Float	1.0	控制旋涡元素波动混合到基础元素的数量。如果设置为0，仅显示基础元素，取值范围在0~3.0之间。可动画

(续表)

属性名称	数据类型	默认值	说明
<Swirl>.Twist	Float	1.0	指定旋涡盘旋扭曲的圈数。如果指定为 0，两个构成元素只随机分布但不产生扭曲，负值可以改变旋涡盘旋扭曲的方向，取值范围在 -20.0~20.0 之间。可动画
<Swirl>.Constant_Detail	Integer	4	控制旋涡贴图的细节级别，如果指定为 0，旋涡贴图的所有细节全部消失，取值范围在 0~10 之间。可动画
<Swirl>.Center_Position_Y	Float	-0.5	指定旋涡中心在对象表面的位置。可动画
<Swirl>.Center_Position_X	Float	-0.5	指定旋涡中心在对象表面的位置。可动画
<Swirl>.Lock_Background	Integer	1	设置是否锁定 X/Y 的位置参数，如果设置为 On，则调整一个参数时，另一个参数会随同变化
<Swirl>.Random_Seed	Float	23638.4	指定旋涡贴图效果的随机种子数，可以在相同的参数设置下，创建不同的旋涡效果。可动画
<Swirl>.coordinates	SubAnim		参见 StandardXYZGen

**注意**

Base 和 Swirl 的子贴图只有在被指定给 Swirl TextureMap 后，才可以在 MAXScript 里像属性一样被存取。而且属性名称取决于被赋给的子贴图类型。比如：如果砖块和灰泥的子贴图类型分别为 Checker 和 Cellular，属性名分别类似于 Swirl\_Map\_7\_Checker 和 Swirl\_Map\_6\_Cellular。

#### 11.4.34 Thin\_Wall\_Refraction: TextureMap (薄壁折射贴图)

##### 构造函数

```
Thin_Wall_Refraction ...
```

```
ThinWallRefraction ...
```

##### 属性

属性名称	数据类型	默认值	说明
<Thin_Wall_Refraction>.applyBlur	Boolean	True	设置是否对贴图执行虚化处理
<Thin_Wall_Refraction>.blur	Float	1.0	依据折射/反射对象距离当前对象的远近，指定反射/折射虚化的强度，利用该参数还可以获得抗锯齿的处理效果。可动画
<Thin_Wall_Refraction>.frame	Integer	0	设置自动薄壁折射计算的帧类型： 0: First Frame Only (仅在第一帧进行自动薄壁折射计算) 1: Every Nth Frame (依据指定的帧间隔，每隔指定帧数进行一次自动薄壁折射计算)

(续表)

属性名称	数据类型	默认值	说明
<Thin_Wall_Refraction>.nthFrame	Integer	0	当属性.frame=1 时, 为自动薄壁折射计算指定帧间隔数
<Thin_Wall_Refraction>.useEnvironment	Boolean	True	设置薄壁折射是否自动对场景环境进行折射计算。
<Thin_Wall_Refraction>.thicknessOffset <Thin_Wall_Refraction>.Thickness_Offset	Float	0.5	指定折射偏移的大小, 如果指定为 0 不产生折射偏移, 取值范围在 0.0~10.0 之间。可动画
<Thin_Wall_Refraction>.bumpMapEffect <Thin_Wall_Refraction>.Bump_Map_Effect	Float	1.0	指定凹凸贴图效果对折射的影响, 减小该数值减小二级折射的效果; 增加该数值增加二级折射的效果。如果当前材质没有指定凹凸贴图, 该参数没有作用效果。可动画

#### 11.4.35 Vertex\_Color: TextureMap (顶点颜色贴图)

##### 构造函数

Vertex\_Color ...

VertexColor ...

##### 属性

属性名称	数据类型	默认值	说明
<Vertex_Color>.map <Vertex_Color>.Map_ID	Integer	0	指定贴图的通道
<Vertex_Color>.subid	Integer	0	设置贴图子通道: 0: All; 1: Red; 2: Green; 3: Blue

#### 11.4.36 Water: TextureMap (波浪贴图)

##### 构造函数

Water ...

Waves ...

##### 属性

属性名称	数据类型	默认值	说明
<Water>.numWaveSets <Water>.Num_Wave_Sets	Integer	10	指定波浪贴图中的波纹数量。可动画
<Water>.waveRadius <Water>.Wave_Radius	Float	800.0	指定波浪贴图的分布半径。数值越小波纹越密集。可动画

(续表)

属性名称	数据类型	默认值	说明
<Water>.waveLenMax	Float	50.0	指定波浪贴图中的最大波长。可动画
<Water>.waveLenMin	Float	5.0	指定波浪贴图中的最小波长。在最大波长和最小波长范围内随机分布波长数值，如果两个参数设置得比较接近，可以创建规则的波浪贴图效果；如果两个参数相差很大，可以创建更为无序的波浪贴图效果。可动画
<Water>.amplitude	Float	1.0	通过调整波纹两种色彩的对比度，指定波纹的强度或深度。可动画
<Water>.phase	Float	0.0	指定波纹震荡的开始相位，为相位参数的变化创建动画，可以创建动态的波浪贴图效果。可动画
<Water>.distribution	Integer	0	设置波浪贴图的波纹分布方式： 0：3D（三维类型的波浪贴图，波纹以虚拟的球体分布波浪贴图） 1：2D（二维类型的波浪贴图，波纹以虚拟的圆形平面分布波浪贴图）
<Water>.randomSeed	Integer	30159	设置波浪贴图纹理的随机种子数，可以在不改变其他参数设置的情况下，变化波浪贴图的纹理效果
<Water>.color1 <Water>.Color_1	Color	默认值：(color 198.9 198.9 239.7) 指定波浪贴图的波谷色彩。可动画	
<Water>.map1	TextureMap	undefined	指定波浪贴图的波谷色彩的替代贴图
<Water>.map1On <Water>.Map1_On	Boolean	True	设置是否使用波浪贴图的波谷色彩的关联贴图
<Water>.color2 <Water>.Color_2	Color	默认值：(color 25.5 25.5 198.9) 指定波浪贴图的波峰色彩。可动画	
<Water>.map2	TextureMap	undefined	指定波浪贴图的波峰色彩的替代贴图
<Water>.map2On <Water>.Map2_On	Boolean	True	设置是否使用波浪贴图的波峰色彩的关联贴图
<Water>.cords <Water>.Coordinates	StandardXYZGen		参见 StandardXYZGen

#### 11.4.37 Wood: TextureMap (木材贴图)

##### 构造函数

Wood ...

##### 属性

属性名称	数据类型	默认值	说明
<Wood>.thickness <Wood>.Grain_Thickness	Float	7.0	指定木材贴图的条纹宽度。可动画
<Wood>.radialNoise <Wood>.Radial_Noise	Float	1.0	沿中心放射方向，指定木材贴图随机噪波变化。可动画
<Wood>.axialNoise <Wood>.Axial_Noise	Float	1.0	沿垂直轴向，指定木材贴图随机噪波变化。可动画
<Wood>.color1 <Wood>.Color_1	Color	默认值: (color 201.45 175.95 68.85)	指定木材贴图的底色色彩。可动画
<Wood>.map1	TextureMap	undefined	指定木材贴图的底色色彩的替代贴图
<Wood>.map1On <Wood>.Map1Enabled	Boolean	True	设置是否使用木材贴图的底色色彩的替代贴图
<Wood>.color2 <Wood>.Color_2	Color	默认值: (color 130.05 81.6 12.75)	指定木材贴图的条纹色彩。可动画
<Wood>.map2	TextureMap	undefined	指定木材贴图的条纹色彩的替代贴图
<Wood>.map2Enabled <Wood>.MapOn2	Boolean	True	设置是否使用木材贴图的条纹色彩的替代贴图
<Wood>.cords <Wood>.Coordinates	StandardXYZGen		参见 StandardXYZGen

# 第 12 章 动画控制器

## 12.1 Controller (控制器) 类

在 3ds max 里所有动画都是通过一种叫 Controller (控制器) 的类来实现的，用户可以在 Track View 视窗和 Motion (运动) 命令面板里看到这些类，还可以在按下 Animate 按钮给对象设置关键帧时由 3ds max 自动赋给。大多数可以制作关键帧的 Controller 类将动画值存储在其属性.keys 里。

Controller 类的组织层次分超类 (SuperClass) 和类 (Class) 两种。每种超类里包含的全部类都使用同一种数据类型 (如 Float 或 Point3)。在 3ds max 里每一个可动画的属性都有它自己的数据类型，如果某一超类里包含的类使用的数据类型与这个属性的数据类型一致，我们就可以把这种 Controller 类赋给这个属性。例如：超类 Float 下的类 Linear\_float 和 Noise\_float 都可以被赋给 Box 对象的属性.height，因为该属性的数据类型也为 Float。我们可以用函数 apropos() 来列出所有控制器超类，其调用格式如下：

```
apropos "controller:super"
```

我们还可以用函数 showClass() 来列出所有可用的 Controller 类名，其调用格式如下：

```
showClass "*.*controller*"
```

将列出超类名里包含字符串 Controller 的所有类。这些类至少包含以下几种：3ds max 内嵌的 Controller 类和所有第三方插件 Controller 类。在有些情况下，复杂插件里包含的 Controller 也会在函数 showClass() 的输出结果中看得到 (如 Character Studio 和 HyperMatter)，但用户不能单独创建并使用这些 Controller，否则会导致系统运行错误。

## 12.2 控制器通用属性

### 通用属性

所有控制器类都有下面的属性：

属性名称	数据类型	说明
<controller>.keys	MAXKeyArray	返回控制器<controller>的关键帧数组。只读
<controller>.value	不定	存取控制器<controller>的当前值
<controller>.Ease_Curve	Float	控制器<controller>的 Ease Curve 值。如果控制器被赋给一个 Ease Curve，该子属性可以被存取
<controller>.Multiplier_Curve	Float	控制器<controller>的 Multiplier Curve 值。如果控制器被赋给一个 Multiplier Curve，该子属性可以被存取

其中属性.keys 和.value 存取控制器的关键帧和值信息。如果一个控制器不能创建关键帧，其属性.keys 会返回一个空的 MAXKeyArray 值。

属性.value 的值与当前时间和 animate 关联语句有关，这样用户可以用它来获取不同时间下控制器的不同值，并可以用来为那些支持关键帧的控制器加入关键帧。

如果一个控制器支持关键帧，用户只能将一个值赋给该控制器的.value 属性。但也有一个例外：我们可以将一个 Matrix3 值赋给一个 PRS 类控制器的.value 属性，MAXScript 会自动根据 Matrix3 值设置好三个子控制器 Position、Rotation 和 Scale 的.value 属性值。

下面是两个使用属性.keys 和.value 的例子：

```
globaltracks.float.mycontroller.value
globaltracks.float.mycontroller.keys[2]
```

### 相关属性

在 3ds max 里所有可动画的（标记为“可动画”）属性都可以引用下面这些与控制器有关的子属性：

属性名称	数据类型	说明
<property>.controller		属性的控制器
<property>.track		与属性.controller 一样
<property>.isAnimated	Boolean	返回指定属性是否有设置动画
<property>.keys	MAXKeyarray	获取属性控制器的关键帧数组，只读
<property>.supportsKeys	Boolean	如果控制器能够创建关键帧，返回 True。只读
<property>.keyable	Boolean	如果控制器能够设置关键值，返回 True。只读
<property>.isHSV	Boolean	如果控制器表示 HSV 颜色，返回 True。只读

如果一个可动画的属性尚没有设置动画，属性.isAnimated 返回 False，并且属性.controller、.track 和.keys 都会返回值 undefined。例如：

```
$foo.pos.controller
bend_mod1.angle.isAnimated
$bar.taper.gizmo.scale.keys[2]
```

用户可以通过调用一个函数来创建一个控制器，就像创建其他 3ds max 对象一样，并可以将它们赋给那些可动画的属性，只要将刚创建的控制器赋给这些属性的.controller 子属性就可以了。如：

```
c = bezier_position()
$foo.pos.controller = c
$baz.bend.gizmo.rotation.controller = tcb_rotation()
$Box01.length = linear_float()
```

## 12.3 控制器通用方法

1. getPointControllers {<editable\_mesh\_node> | <editable\_spline\_node>}

返回一个数组，包含指定的 Mesh 或 Spline 对象顶点的控制器。如果某一顶点此时还没有赋给控制器，该顶点对应的元素值为 undefined。对 Editable Spline 类对象，每个结点 (knot) 由三个矢量组成：in 矢量、knot 矢量和 out 矢量。本函数返回的控制器数组为当前时间下的各顶点控制器的“快照”(Snapshot)，如果执行本方法后某一顶点的控制器又被改变，这种改变不会反映到这个数组里；如果添加或删除了一个顶点，本数组的长度也不会改变。

### 2. getPointController <editable\_mesh\_node><vertex\_index\_integer>

返回 Mesh 对象里指定顶点的控制器，如果该顶点还没有被赋给控制器，返回 undefined。

### 3. getPointController <editable\_spline\_node> \

#### <spline\_index\_integer><vertex\_index\_integer>

返回 Editable Spline 对象指定顶点的控制器，如果该顶点还没有被赋给控制器，返回 undefined。

#### 注意

如果将一个控制器赋给某一属性，系统会将赋值前的属性值作为控制器在 0 帧的值。而如果控制器在被赋给属性之前已有关键值，所有的关键值会加上控制器原来在 0 帧的值与属性值之间的差值。下面的例子说明了这一点：

```
a=bezier_float()
addnewkey a 0
addnewkey a 10
a.keys[1].value=10
a.keys[2].value=100
b=Box()
b.height
b.height.controller=a
b.height
at time 10 b.height
输出：
Controller:Bezier_Float      -- 第 1 行输出结果
#Bezier Float key(1 @ 0f)    -- 第 2 行输出结果
#Bezier Float key(2 @ 10f)   -- 第 3 行输出结果
10                           -- 第 4 行输出结果，在 0 帧时的帧值
100                          -- 第 5 行输出结果，在 0 帧时的帧值
$Box:Box01 @ [ 0.0,0.0,0.0] -- 第 6 行输出结果
25.0                         -- 第 7 行输出结果，属性默认值
Controller:Bezier_Float      -- 第 8 行输出结果
25.0                         -- 第 9 行输出结果，帧值由 10 变为 25，差值为 15
115.0                        -- 第 10 行输出结果，帧值被加上 15
```

当把同一个控制器赋给多个对象属性时，会发生一个有趣的现象：实际存储在控制器里的值可能与显示在 Track View 或命令面板 Create、Modify 里的值不一样，也可能与在 MAXScript 里该属性的返回值不一样。这是因为一部分对象属性从控制器里读写数据时，系统会对这些数据进行比例缩放。如属性.slice\_to 和.slice\_from 在很多基本类几何体 (Primitive Geometry) 对象里都有用到，它在 Modify 面板里显示单位为度，而实际在控制

器里存储单位为弧度，其缩放因子由系统内部控制，而非控制器的一个特征。**3ds max** 和 **MAXScript** 都会自动对这些属性施加缩放操作，这样这种缩放实际上对用户是不可见的。

如果同一个控制器被赋给两个拥有不同缩放因子的属性，会产生用哪个缩放因子来计算控制器输出值的问题。假设用户通过其中一个对象属性来存取控制器的值，与该属性相连的缩放因子被施加于控制器，**MAXScript** 会在内部存储最后施加于控制器值的那个缩放因子。此时如果用户直接存取控制器的.value 属性，或控制器的某一帧的.value 属性，系统会使用存储在 **MAXScript** 里的缩放因子，这样可能会导致在脚本里不同时间存取控制器的值得到不同的结果，其返回值取决于控制器最后一次被存取的方式。因此，建议用户不要将同一个控制器赋给多个拥有不同缩放因子的对象属性。各属性的缩放因子（如果有的话）在各 **MAXWrapper** 类的属性说明中均有说明。

#### 4. displayControlDialog <controller> <string>

显示所有模式的 Controller 对话框（如果有）。变量<string>指定了对话框标题。如果指定<controller>有关键帧，会为选中的帧显示 Key Info 对话框。如果没有选择关键帧，不会显示对话框。

## 12.4 与控制器时间有关的方法

### 1. supportsTimeOperations <ctrl>

如果指定控制器<ctrl>支持下面的时间操作，返回 True；否则，返回 False。

### 2. deleteTime <controller> <interval> [ #incLeft ] [ #incRight ] [ #noSlide ]

从指定控制器里删除一个时间段，将该时间段里的所有关键帧清除，并且在默认情况下，将时间段右边的关键帧左移至时间段的起点。可选参数有以下几种选项：

- ◆ #incLeft 包括时间段起始处的任何关键帧；
- ◆ #incRight 包括时间段结尾处的任何关键帧；
- ◆ #noSlide 不将时间段后面的关键帧左移以填补被清除的空缺，而是仅仅删除时间段内的所有关键帧；
- ◆ <interval> 可以按 Interval 类值的格式来指定：用两个数值或 Time 类值指定起始和终止时间。如果用数值，其单位为帧。

### 3. reverseTime <controller> <interval> [ #incLeft ] [ #incRight ]

反转指定时间段，实际上是将时间段内的关键帧进行交换。有关变量的说明参见函数 **deleteTime()**。

### 4. scaleTime <controller> <interval> <float\_scale>

将指定时间段里关键帧的时间进行缩放。变量<interval>的说明详见函数 **deleteTime()**。

### 5. insertTime <controller> <at\_time> <amount\_time>

在指定时间处插入一段时间，这样可以将后面的关键帧向后移一段时间。变量<at\_time>和<amount\_time>数据类型可以为 Number 或 Time 类值。如果为数值，其单位为帧。

### 6. getTimeRange <controller> [ #selOnly ] [ #allKeys ] [ #children ]

返回一个时间段，这个时间段涵盖了指定范围的关键帧。可选的参数有：

- ◆ #selOnly 返回 Track View 视窗里当前选定的关键帧涵盖的时间段；
- ◆ #allKeys 默认值，Controller 里全部关键帧涵盖的时间段；
- ◆ #children 返回时间段包括所有子 Controller 关键帧涵盖的时间段。

#### 7. setTimeRange <controller> [ <interval> ] [ #linkToKeys ]

为指定控制器设置一个时间段，取代控制器已有关键帧涵盖的时间段，主要用于 out-of-range 方法生效时。如果指定了可选变量#linkToKeys，移动时间段起点、终点处的关键帧会改变指定控制器的时间段。如果调用函数 setTimeRange()时仅使用单个变量#linkToKeys，会把当前起始、终止关键帧的时间间隔设为所有相关控制器的时间段，这样做与在 Track View 视窗里 Position Ranges 模式下执行 Recouple Ranges 函数具有相同的效果。如：

```
setTimeRange $Box* #linkToKeys
```

下面的例子演示了与控制器时间有关函数的一些使用方法：

```
b=Box height:10
at time 5 animate on b.height=50
at time 10 animate on b.height=100
bhc=b.height.controller
bhk=bhc.keys
supportstimeoperations bhc
deletetime bhc 4 5
bhk
deletetime bhc 4 5 #incLeft
bhk
deletetime bhc 1 4 #noslide
bhk
at time 5 animate on b.height=50
deletetime bhc(interval 5 8)#incLeft
bhk
at time 10 animate on b.height=150
for k in bhk do format "%:%\n" k.time k.value
reversetime bhc 5 15 #incLeft #incRight
for k in bhk do format "%:%\n" k.time k.value
insertTime bhc 12 5
bhk
getTimeRange bhc
```

## 12.5 与控制器关键帧有关的方法

### 1. addNewKey <controller> <time> [ #select ]

在指定时间处给控制器添加一个关键帧。新关键帧的值为在指定时间处的插入值。如果指定了参数#select，新关键帧会被选中。如果在指定时间处已存在一个关键帧，addNewKey()函数不会再添加一个关键帧，在这种情况下，函数返回值为指定时间处的关键帧。本函数返回一个 MAXKey 类值，代表该关键帧，用户可以设置它的各种属性。

2. `deleteKeys <controller> [ #allKeys ] [ #selection ]`

根据提供的参数删除控制器的关键帧：

- ◆ `#allKeys` 默认值，删除指定 Controller 的所有关键帧；

- ◆ `#selection` 删除当前被选中的关键帧。

3. `deleteKey <controller> <index>`

删除指定序号的关键帧。关键帧序号从 1 开始。

4. `selectKeys <controller> [ <interval> | <time> ]`

选择指定的关键帧。如果指定`<interval>`值，在指定时间段内的全部关键帧会被选中；如果指定一个单独的时间`<time>`，此时间位置的关键帧被选中；如果仅指定`<controller>`，表示选择全部关键帧。

5. `deselectKeys <controller> [ <interval> | <time> ]`

取消对指定多个关键帧的选择。参数说明同函数 `selectKeys()`。

6. `selectKey <controller> <index_integer>`

选择指定序号的关键帧，关键帧序号从 1 开始。

7. `isKeySelected <controller> <index_integer>`

如果指定序号的当前关键帧被选中，返回 `True`；否则，返回 `False`。

8. `deselectKey <controller> <index_integer>`

取消指定序号关键帧的选择。

9. `moveKeys <controller> <time> [ #selection ]`

将指定关键帧移动指定时间。如果有指定参数`#selection`，移动当前被选中的关键帧；否则，为全部关键帧。

10. `moveKey <controller> <index_integer> <time>`

将指定序号的关键帧移动指定时间。

11. `numKeys <controller>`

返回指定 Controller 里的关键帧数量。如果参数`<controller>`不支持关键帧，返回 -1。

12. `getKey <controller> <index_integer>`

返回一个 MaxKey 值，代表指定索引的关键帧。

13. `getKeyTime <controller> <index_integer>`

返回指定序号关键帧的时间。

14. `getKeyIndex <controller> <time>`

返回指定时间处关键帧的序号。

15. `numSelKeys <controller>`

返回在 Track View 视窗里当前被选中关键帧的数量。

16. `sortKeys <controller>`

根据时间对关键帧重新排序。某些 MAXKey 操作可能会打乱关键帧的时间顺序，本函数可以纠正这种无序的时间排列。

17. `createLockKey <controller> <time> <rot_or_pos_integer>`

在指定时间处创建一个“锁定”的关键帧。系统先在指定时间处创建一个新关键帧，

新关键帧的值会锁定前一关键帧的值，并与其始终匹配。本方法同时也会调整关键帧的参数，以便新建关键帧与前一关键帧的差值保持常数。如对 TCB 类，控制器前一关键帧和新关键帧的continuity 属性会被设为 0；对 Bezier 类 Controller 前一关键帧的 out-tangent 类型被设为 Linear；而新关键帧的 in-tangent 类型被设为 Linear。如果参数<controller>指定的是一个 Transform 级控制器，变量<rot\_pos\_integer>指定是在该控制器的 Position 还是在 Rotation（或 roll\_angle）子控制器（Controller）上创建新关键帧。如果 rot\_pos\_integer=0，新关键帧在 Position 子控制器上创建；如果为其他值，新关键帧在 Rotation（或 roll\_angle）子控制器上创建。如果参数<controller>不是一个 Transform 级控制器，参数<rot\_pos\_integer>无需指定。如果新关键帧创建成功，返回 True；否则，返回 False。

下面脚本程序显示了与控制器关键帧有关函数的一些使用方法：

```
b=Box height:10
at time 5 animate on b.height=50
at time 10 animate on b.height=100
bhc=b.height.controller
bhk=bhc.keys
addnewkey bhc 7
addnewkey bhc 9
for k in bhk do format "%:%\n" k.time k.value
selectKeys bhc(interval 7 9)
deleteKeys bhc #selection
bhk
addnewkey bhc 7
addnewkey bhc 9
selectKeys bhc(interval 7 9)
deleteKeys bhc #selection #slide
bhk
addnewkey bhc 7
addnewkey bhc 9
selectKeys bhc(interval 7 9)
deleteKeys bhc #selection #slide #rightToLeft
bhk
addnewkey bhc 8
i=getKeyIndex bhc 8
selectKey bhc i
moveKey bhc i 10
bhk
getKeyTime bhc 4
b.width.controller=noise_float()
numkeys b.width.controller
```

## 12.6 与控制器 ORT 有关的方法

### 1. getBeforeORT <controller>

获取指定 Controller 在第一个关键帧之前的 ORT（Out-of-Range）类型。返回值为下面

Name 值之一: #constant、#cycle、#loop、#pingPong、#linear、#relativeRepeat。

2. getAfterORT <controller>

获取指定 Controller 在最后一个关键帧之后的 ORT 类型。返回值同函数 getBeforeORT() 的返回值。

3. setBeforeORT <controller> <ORT\_type\_name>

设置指定 Controller 在第一个关键帧之前的 Out-of-Range (ORT) 类型。变量<ORT type name>必须为下面 Name 值之一:#constant、#cycle、#loop、#pingPong、#linear、#relativeRepeat。

4. setAfterORT <controller> <ORT\_type\_name>

设置指定 Controller 在最后一个关键帧之后的 ORT 类型。变量<ORT\_type\_name>说明同函数 setBeforeORT()。系统不会对其有效性作检查。

5. enableORTs <controller> <boolean>

启用或禁用控制器的 ORT 处理, 如果<boolean>设为 False, 表示禁用 ORT 处理, 控制器会自动使用 Constant 类型 ORT。

## 12.7 与控制器 Ease 曲线和 Multiplier 曲线有关的方法

1. addEaseCurve <controller> [ <float\_controller> ]

给指定控制器<controller>添加一个 Ease 曲线。用户可提供一个可选参数<float\_controller>以指定曲线, 否则本函数会产生一个默认的 Float 类控制器。如果用户在 Track View 视窗里添加一个 Ease 曲线, 系统会自动在当前动画范围的起点和终点创建一个关键帧。但用函数 addEaseCure()添加 Ease 曲线不能自动产生这些关键帧, 用户需自己向 Ease 曲线控制器添加这些关键帧。

2. deleteEaseCurve <controller> <index\_integer>

从指定控制器<controller>里删除指定序号的 Ease 曲线。序号从 1 开始, 并按 Ease 曲线被添加的顺序排序。

3. numEaseCurves <controller>

返回指定控制器<controller>当前的 Ease 曲线数量。

4. applyEaseCurve <controller> <time>

将 Ease 曲线在指定的时间赋给指定 Controller。

5. addMultiplierCurve <controller> [ <float\_controller> ]

向指定控制器<controller>添加一个 Multiplier 曲线。用户可以提供一个可选参数<float controller>来指定该曲线, 否则本函数会为 Controller 生成一个默认 Float Controller。

如果用户在 Track View 视窗里添加一个 Multiplier 曲线, 系统会自动在当前动画范围的起点和终点创建一个关键帧。但用方法 addMultiplierCurve()添加 Multiplier 曲线不能自动产生这些关键帧, 用户需要自己向 Multiplier 曲线控制器添加这些关键帧。

6. deleteMultiplierCurve <controller> <index\_integer>

从指定控制器里删除指定序号的 Multiplier 曲线。序号从 1 开始, 并按 Multiplier 曲线

被添加的顺序排序。

7. numMultiplierCurves <controller>

返回指定 Controller 当前的 Multiplier 曲线数量。

8. getMultiplierValue <controller> <time>

返回指定 Controller 在指定时间处 Multiplier 曲线的结合值（数据类型为 Float）。

**注意** 在 Ease 曲线控制器里存储的时间格式为整型时间 tick。

下例演示了与控制器 Ease 曲线和 Multiplier 曲线有关方法的用法：

```
b=Box height:10
at time 5 animate on b.height=50
at time 10 animate on b.height=100
bhc=b.height.controller
ec=bezier_float()
at time 0 animate on ec.value = 0
at time 100 animate on ec.value=100*ticksPerFrame
setBeforeORT ec #linear
setAfterORT ec #linear
addEaseCurve bhc ec
applyEaseCurve bhc 50
addEaseCurve bhc ec
numEaseCurves bhc
deleteEaseCurve bhc 2
numEaseCurves bhc
at time 50 bhc.ease_curve.value
mc=bezier_float()
at time 0 animate on mc.value = 1
at time 100 animate on mc.value = 2
addMultiplierCurve bhc mc
at time 50 bhc.multiplier_curve
```

## 12.8 与控制器关键帧衰减有关的方法

reduceKeys <controller> <threshold> <step> [ <range> ]

将指定<controller>的指定时间段内的关键帧进行衰减。

其中参数<threshold>为衰减后的关键帧与原来关键帧的接近程度；参数<step>为采样的步长；可选参数<range>定义了要进行关键帧衰减的时间段，可以用两个时间值或一个 Interval 值来定义，默认值为当前活动的动画时间段。例如：

```
reduceKeys $foo.pos.controller 0.5 1f
```

将对象 foo 的 position 控制器进行关键帧衰减，保留 0.5 倍精度，对每帧进行采样处理。

下面的例子将对象 baz 的弯转角度控制器进行关键帧衰减，保留 0.1 倍精度，对每半帧进行采样处理，衰减时间段为从 20 帧到 75 帧。

```
reduceKeys $baz.bend.angle.controller 0.1 0.5f(interval 20 75)
```

和在 Track View 视窗里操作衰减器一样，本函数运行时会显示一个进度栏和一个 Cancel 按钮。

## 12.9 与对象层级有关的时间和关键帧方法

下面这些控制器时间和关键帧方法的参数`<controller>`不仅可以是单个控制器，还可以是任何 3ds max 对象或对象集合。使用后一种方式调用时，这些方法会逐一作用于指定对象和子对象的控制器以及这些对象所包含的子控制器。从这一点看，其工作方式和 Track View 里的 Object 级轨道类似，两者均可以操作其下面所有子对象和子控制器的关键帧和时间。

```
deleteTime
reverseTime
scaleTime
insertTime
setTimeRange
addNewKey
deleteKeys
selectKeys
deselectKeys
moveKeys
sortKeys
reduceKeys
addEaseCurve
deleteEaseCurve
setBeforeORT
setAfterORT
enableORTs
```

以上这些方法的参数`<controller>`都可以是任何 3ds max 对象集合（如一个带匹配符路径名或一个对象数组），请读者参考下面示例：

```
--Box01 对象的 xform Modifier 里所有控制器在 0 帧以后的关键帧向后移 10 帧
insertTime $Box01.xform 0f 10f
--Box01 对象全部 0 帧以后的所有关键帧向后移动 10 帧。
insertTime $Box01 0f 10f
--选择 0~100 帧范围里全部关键帧
selectKeys $Box02.xform.gizmo.rotation.controller 0f 100f
--删除所有名字以 Box 开头的对象所有控制器的第 10 个关键帧
deleteTime $Box* 10f 10f
--将对象 Box03 所有子对象的所有控制器 0 帧以后的关键帧后移 10 帧
insertTime $Box03.children 0f 10f
--将对象 Box01 的所有 Modifier 施加关键帧衰减
reduceKeys $Box01.modifiers 0.5 1f
```

## 12.10 控制器类型

### 12.10.1 控制器超类级

下面清单是全部控制器超类:

超类名	包含子类	说明
FloatController	AudioFloat、Bezier_Float、Block、Float_Expression、Float_List、Float_Motion_Capture、Float_Reactor、Float_Script、Linear_Float、LOD_Controller、Noise_Float、On_Off、SlaveFloat、TCB_Float、Waveform_Float	可以被赋给数据类型为 Float 或 Integer 属性的控制器
Matrix3Controller	Link_Control、Link_Transform、LookAt、PRS IK_ControllerMatrix3Controller、Slave_Controller	可以被赋给 Transform 或 Position-Rotation-Scale (位置-转角-缩放) 类参数的控制器
MorphController	Barycentric_Morph_Controller Cubic_Morph_Controller	可以赋给 Morph 类对象的控制器的超类
Point3Controller	AudioPoint3、Bezier_Color、Bezier_Point3、Color_RGB、Noise_Point3、Point3_Expression、Point3_List、Point3_Motion_Capture、Point3_Reactor Point3_Script、Point3_XYZ、Slave_Point3、TCB_Point3	可以被赋给 Point3 类参数的控制器的超类
PositionController	Attachment、Audiposition、Bezier_Position、Dynamic_Position_Controller、Linear_Position、Noise_Position、Path、Position_Expression、Position_List、Position_Motion_Capture、Position_Reactor、Position_Script、Position_XYZ、Surfaceposition、TCB_Position	可以被赋给位置 (Position) 类参数的控制器的超类
RotationController	AudioRotation、Bezier_Rotation、Dynamics_Rotation_Controller、Euler_XYZ、Linear_Rotation、Local_Euler_XYZ、Noise_Rotation、Rotation_List、Rotation_Motion_Capture、Rotation_Reactor、Rotation_Script、SlaveRotation、TCB_Rotation	可以被赋给转角 (Rotation) 类参数的控制器的超类
ScaleController	AudioScale、Bezier_Scale、Linear_Scale、Noise_Scale、Scale_Expression、Scale_List、Scale_Motion_Capture、Scale_Reactor、Scale_Script、ScaleXYZ、SlaveScale、TCB_Scale	可以被赋给缩放 (Scale) 类参数的控制器的超类
QuatController	目前还没有与 QuatController 超类相关的控制器	
MasterBlockController	Block_Control、MasterBlock	与 Block Control Track View 节点对象有关的控制器

### 12.10.2 Attachment: PositionController (附着点约束控制器)

#### 构造函数

Attachment...

#### 属性

属性名称	数据类型	默认值	说明
<attachment>.keys	MAXKeyArray		
<attachment>.node	Node		
<attachment>.align	Boolean	True	当设置为 On 时, 调整结合对象的方向使其与目标对象表面对齐; 否则, 结合对象的方向不受目标对象表面方向的影响
<attachment>.manupdate	Boolean	False	是否使 Update 按钮可用

#### 方法

1. AttachCtrl.getKey <attachment> <index\_integer>

返回一个 MAXAKey 值, 表示指示控制器的指定关键帧。

2. AttachCtrl.addNewKey <attachment> <time>

在指定时间处为 Controller 添加一个关键帧。

3. AttachCtrl.update <attachment>

如果指定<attachment>的属性.manupdate 为 True, 会更新控制器。

#### 注意

用.keys[i]的格式存取 Attachment 控制器的关键帧会返回一个 MAXKey 值, 通过这种方式获取的关键帧只能存取属性.time 和.selected。而方法 AttachCtrl.getKey()返回一个 MAXAKey 类值, 用户可以通过它存取该控制器关键帧的所有属性。所有通用于控制器的函数如 deleteKey、selectKeys、movekeys 等都可以用于 Attachment 类控制器。

#### 与附着点约束控制器控制器关键帧有关的属性

Attachment 控制器的关键帧数据类型为 MAXAKey, 这一点和绝大多数其他控制器的关键帧数据类型为 MAXKey 不同, 这些关键帧有下面一些属性:

属性名称	数据类型	说明
<MAXAKey>.face	Integer	存取关键帧附着的面序号, 序号从 0 开始。
<MAXAKey>.coord	Point2	存取关键帧附着面的重心坐标。
<MAXAKey>.time	Time	Time 类值或数值(以帧为单位)
<MAXAKey>.selected	Boolean	指定关键帧是否被选中
<MAXAKey>.value	不定	数据类型取决于包含它控制器
<MAXAKey>.tension	Float	存取关键帧的张力
<MAXAKey>.continuity	Float	存取关键帧的连续性
<MAXAKey>.bias	Float	存取关键帧的偏移值
<MAXAKey>.easeTo	Float	指定当运动接近关键帧时, 减缓运动的速度
<MAXAKey>.easeFrom	Float	指定当运动离开关键帧时, 减缓运动的速度

下面的脚本是一个使用Attachment控制器的例子。

```
c = cylinder height:100 heightsegs:10 --创建一个Cylinder对象
addModifier c (Bend()) --给Cylinder对象施加一个Bend Modifier
s = geosphere() --用默认参数创建一个Geosphere对象
actrl = Attachment() --创建一个Attachment controller
--将控制器赋给Geosphere对象的position属性
s.position.controller = actrl
actrl.node = c --将Cylinder对象设为附着对象
addNewKey actrl 0f --在0帧添加一个关键帧
theAKey = AttachCtrl.getKey actrl 1 --获取第一个关键帧
theAKey.face = c.mesh.numfaces - 1
theAKey.coord = [1,0]
with animate on at time 100 --在100帧处设置动画
(
c.height = 200 --animate the height of the cylinder
c.bend.angle = 90 --and the bend angle
)
```

### 12.10.3 Audio Controller (音频控制器)

#### 类型

audiofloat: FloatController  
audiopoint3: Point3Controller  
audioposition: PositionController  
audiorotation: RotationController  
audioscale: ScaleController

#### 构造函数

audiofloat...  
audiopoint3...  
audioposition...  
audiorotation...  
audioscale...

#### 属性

Audio Controller 没有额外的属性。

### 12.10.4 Barycentric\_Morph\_Controller: MorphController (重心变形控制器)

#### 构造函数

createMorphObject <source\_node>

将指定场景对象变为一个Morph类复合对象。如果参数<source\_node>指定的对象已经

是一个 Morph 对象，系统什么也不会执行。返回的 Morph 对象会自动生成一个 Morph 控制器，可以用属性.morph 来存取这个控制器，如：

```
c = $foo.morph      -- 获取 Morph 类 Controller
mk1 = c.keys[1]     -- 获取第一个 Morph 关键帧
```

### 方法

#### 1. addMorphTarget <morph\_controller> <target\_node> <add\_method>

给指定 Morph 类控制器添加一个 Morph 目标对象。变量<add\_method>定义了目标对象的添加方法，并且必须是 1~4 之间的一个整数：

- ◆ 1 reference (参照)
- ◆ 2 copy (复制)
- ◆ 3 move (移动)
- ◆ 4 instance (实例)

返回值为一个整数，代表所添加目标对象的序号。

#### 2. setMorphTarget <morph\_controller> <target\_index\_integer>|<target\_node> <add\_method>

用新指定的场景对象取代原来的目标对象。变量说明同函数 addMorphTarget()。

#### 3. getMKTTargetNames <morph\_controller>

返回指定 Morph 类控制器里由所有目标对象名字组成的数组。

#### 4. deleteMorphTarget <morph\_controller> <target\_index\_integer>

删除指定序号的 Morph 目标对象。

#### 5. setMorphTargetName <morph\_controller> <target\_index\_integer> <name\_string>

改变指定目标对象的.name 属性。

#### 6. getMKTTargetWeights <morph\_controller> <time> <dest\_array>

快速获取指定时间处指定 Morph 类控制器目标对象的权重。目标对象的权重放在变量<dest\_array>里，该变量必须是一个长度刚好为目标对象个数的数组。用户可以通过函数 getMKTTargetNames()返回数组的长度来获取目标对象的个数。

#### 7. getMKKey <morph\_controller> <time>

返回 Morph 类控制器在指定时间处的关键帧。如果控制器在指定时间没有关键帧，返回值为 undefined。

#### 8. getMKKeyIndex <morph\_controller> <time>

返回指定时间处关键帧的序号。如果指定时间没有关键帧，返回值为 undefined。

**注意** 如果用函数 addnewkey()来创建 Barycentric\_Morph\_Controller 类控制器的一个关键帧，其关键帧的.time 属性的值经常会不准确。

### 和关键帧有关的属性

#### 1. <key>.time: Time 或 Number (以帧为单位)，只读

## 2. &lt;key&gt;.selected: Boolean

存取指定关键帧是否被选中，可读写

## 和重心变形控制器关键帧有关的方法

## 1. getMKTime &lt;morph\_key&gt;

返回指定 Morph 关键帧的时间。

## 2. setMKTime &lt;morph\_key&gt; &lt;time&gt;

为指定 Morph 关键帧设置时间。

## 3. getMKWeight &lt;morph\_key&gt; &lt;target\_index\_integer&gt;

获取指定 Morph 关键帧里指定序号目标对象的权重，返回值为一个百分比，目标对象的序号与 Morph 类关键帧在 Track View 视窗里标题为 Target 的列表框列出的顺序相同，都是按其被添加的顺序排列。

## 4. setMKWeight &lt;morph\_key&gt; &lt;target\_index\_integer&gt; &lt;pcnt\_float&gt; &lt;keep100%\_boolean&gt;

为指定 Morph 关键帧的指定目标对象设置权重，单位为百分比。最后一个变量为布尔值，如果设为 True，表示会调整其他目标对象的权重，使最大权重为 100%。

示例：

```

sel1 = sphere radius:30          --目标对象 1
sel2 = sphere radius:50          --目标对象 2
createMorphObject sel1           --为目标对象 1 创建 Morph 对象
mobj1= sel1.morph                --获取对象的 Morph 控制器
--将对象 sel2 添加为目标对象，创建关键帧，目标权重分别为 0 和 100
addmorphTarget mobj1 sel2 3
k=mobj1.keys[1]                  --获取关键帧
setMKWeight k 1 100 True          --将目标 1 的权重设为 100%，保持总权重为 100%
addnewkey mobj1 50               --在第 50 帧处创建一个关键帧
k=getMKKey mobj1 50              --获取关键帧
--将目标 2 的权重设为 100%，保持总权重为 100%
setMKWeight k 2 100 True

```

输出结果：

```

$Sphere:Sphere01 @ [0.0,0.0,0.0]          --第 1 行结果
$Sphere:Sphere02 @ [0.0,0.0,0.0]          --第 2 行结果
$Editable_Mesh:Sphere01 @ [0.0,0.0,0.0]    --第 3 行结果
Controller:Barycentric_Morph_Controller     --第 4 行结果
2                                         --第 5 行结果
#Barycentric Morph Controller key(1 @ 0f) --第 7 行结果
1                                         --第 8 行结果
#Barycentric Morph Controller key(0 @ 50f) --第 9 行结果
#Barycentric Morph Controller key(2 @ 50f) --第 10 行结果
2

```

### 12.10.5 Bezier Controller (贝塞尔控制器)

#### 类型

bezier\_color: Point3Controller  
 bezier\_float: FloatController  
 bezier\_point3: Point3Controller  
 bezier\_position: PositionController  
 bezier\_rotation: RotationController  
 bezier\_scale: ScaleController

#### 构造函数

bezier\_color...  
 bezier\_float...  
 bezier\_point3...  
 bezier\_position...  
 bezier\_rotation...  
 bezier\_scale...

#### 属性

<bezier\_controller>.keys MAXKeyArray

#### 和贝塞尔控制器关键帧有关的属性

属性名称	数据类型	默认值	说明
<key>.time	Time		Time 类值或数字（以帧为单位）
<key>.selected	Boolean		指定关键帧是否被选择
<key>.value	不定		关键帧的值、数值类型取决于其 Controller 的类型
<key>.inTangent	不定 (Float 或 Point3)		
<key>.outTangent	不定 (Float 或 Point3)		
<key>.inTangentType	Name		指定关键帧切线类型
<key>.outTangentType	Name		
<key>.x_locked	Boolean	True	设置是否锁定 X 轴的放缩比例
<key>.y_locked	Boolean	True	设置是否锁定 Y 轴的放缩比例
<key>.z_locked	Boolean	True	设置是否锁定 Z 轴的放缩比例
<key>.constantVelocity	Boolean	False	设置是否使用匀速运动方式

属性.inTangent 和.outTangent 的数据类型有两种可能：如果控制器类型为 Float，数据类型为 Float；如果控制器为其他类型，数据类型为 Point3。如果为 Float 类控制器，其默认值为 0，否则其默认值为 [0,0,0]。

属性.inTangentType 和.outTangentType 数据类型为 Name，代表六种可能的切线类型，

这些类型可在 Bezier Key property 对话框里的下拉菜单里看到，分别为：#smooth（默认值）、#linear、#step、#fast、#slow、#custom。

示例：

```
c = bezier_position()          -- 创建和指定一个新的 Controller
$bar.pos.controller = c
k = addNewKey c 0f            -- 在 0 帧处插入关键帧
k.value = [10,0,0]             -- 设置关键帧值
k.outTangentType = #slow      -- 设置 outTangent 类型
k = addNewKey c 100f          -- 在 100 帧处插入关键帧
k.value = [200,10,0]           -- 设置关键帧值
k.inTangentType = #custom     -- 设置 inTangent 类型
k.inTangent = [0.2, 0.02, 0.112] -- 设置 inTangent 值
k.x_locked = False            -- 句柄解锁
```

### 12.10.6 Block: FloatController (块控制器)

Block 类控制器不能用 MAXScript 来创建，而只能在 Track View 视窗里通过 Global 轨迹下的 Block Control 节点里的 Masterblock 来创建。

### 12.10.7 Block\_Control: MasterBlockController

不能用 MAXScript 来创建 Block\_Control 类控制器。

**属性**

```
<Block_Control>.weight: Float 数组
<Block_Control>.average: Boolean
```

### 12.10.8 Dynamics Controller (运动控制器)

**类型**

Dynamics\_Position\_Controller: PositionController

Dynamics\_Rotation\_Controller: RotationController

Dynamics 类控制器只能在 Dynamics 工具栏里被创建，而不能用 MAXScript 来创建。

### 12.10.9 Expression Controller (表达式控制器)

**类型**

```
float_expression: FloatController
point3_expression: Point3Controller
position_expression: PositionController
scale_expression: ScaleController
```

### 构造函数

```
float_expression...
point3_expression...
position_expression...
scale_expression...
```

### 属性

表达式控制器没有额外的属性。

## 12.10.10 IK\_ControllerMatrix3Controller: Matrix3Controller (反向动力学控制器)

本节讲述的函数提供了存取 IK 类控制器的某些 IK 参数的途径。在 3ds max 里 IK 类控制器主要用于 Bone System。在 3ds max 里，用户不能在 MAXScript 里创建一个 Bone System，但是本节里讲述的这些方法可以对已有的 Bone 系统的 IK 类控制器进行存取操作。实际上 IK 系统由两种类型控制器组成：主 IK 控制器和从 IK 控制器。在 IK 系统里，从 IK 控制器用于所有节点对象的转换运算，而主 IK 控制器用来控制单个的从 IK 控制器。所有与 IK 相关的方法前面都必须有 ik. 前缀，并与 Motion 面板上的 IK Controller Parameters 卷展栏里的参数相对应。如果对一个控制器类型不是 IK 的对象调用这些函数，会返回值 undefined。存取这些函数的实际上是主 IK 控制器。存取当前场景的 IK 系统里任何对象的 IK 控制器实际上会存取同一个主 IK 控制器。

**注意** 下面的 IK 方法仍然存在，但在 3ds max 4 以及更高版本里已不再使用，如果用这些方法存取数据，均会返回值 undefined。

```
GetRotThreshold
SetRotThreshold
GetPosThreshold
SetPosThreshold
GetIterations
SetIterations
```

下面是与 IK 相关的方法：

1. ik.GetPosThreshold <node>  
ik.SetPosThreshold <node> <float>  
存取指定 Node 对象的 Position 阈值。
2. ik.GetRotThreshold <node>  
ik.SetPosThreshold <node> <float>  
存取指定 Node 对象的 Rotation 阈值。
3. ik.GetIterations <node>  
ik.SetIterations <node> <integer>

存取指定 Node 对象的 Iterations 阈值。

4. ik.GetStartTime <node>

    ik.SetStartTime <node> <time>

存取指定 Node 对象的开始时间。

5. ik.GetEndTime <node>

    ik.SetEndTime <node> <time>

存取指定 Node 对象的结束时间。

6. ik.getPinNode <node>

获取指定 Node 对象在 IK 面板里被绑定的 Node 对象。如果指定对象还没有被绑定，返回 undefined。

7. ik.setPinNode <node>(<node> | undefined)

设置指定 Node 对象在 IK 面板里被绑定的 Node 对象。如果设置为 undefined，表示指定对象被解除绑定；如果绑定成功，返回 True；否则，返回 False。

8. ik.getPrecedence <node>

返回一个整数，表示指定 Node 对象在 IK 面板里的优先级。

9. ik.setPrecedence <node> <number>

为指定 Node 对象设置在 IK 面板里的优先级。如果设置成功，返回 True；否则，返回 False。

10. ik.getIsTerminator <node>

    ik.setIsTerminator <node> <boolean>

存取是否为指定 Node 对象打开 Terminator 开关。

11. ik.getBindPos <node>

    ik.setBindPos <node> <boolean>

存取是否为指定 Node 对象打开 Bind Position 开关。

12. ik.getBindOrient <node>

    ik.setBindOrient <node> <boolean>

存取是否为指定 Node 对象打开 Bind Orientation 开关。

## 12. 10. 11 Linear Controller (线性控制器)

### 类型

linear\_float: FloatController

linear\_position: PositionController

linear\_rotation: RotationController

linear\_scale: ScaleController

### 构造函数

linear\_float...

linear\_position...

linear\_rotation...

linear\_scale...

#### 属性

<controller>.keys MAXKeyArray

和关键帧有关的属性

- ◆ <key>.time Time 或 Number 类值（以帧为单位）
- ◆ <key>.selected Boolean，存取关键帧是否被选择
- ◆ <key>.value 类型不定，关键帧值，数据类型取决于包含的控制器

### 12.10.12 Link\_Control: Matrix3Controller (链接控制器)

#### 构造函数

link\_control...

#### 属性

<link\_control>.transform Matrix3Controller

返回链接控制器包含的 Transform 控制器。

#### 方法

1. linkCtrl.getLinkCount <link\_control>

返回一个整数，表示与 Link 控制器相连的 Node 对象数目。

2. linkCtrl.getLinkNode <link\_control> <index\_integer>

返回一个<Node>值，表示与序号<index\_integer>对应的 Node 对象。

3. linkCtrl.setLinkNode <link\_control> <index\_integer> <node>

为指定序号的 Link 设置节点对象。指定序号的 Link 必须已事先存在。

4. linkCtrl.getLinkTime <link\_control> <index\_integer>

返回一个 Time 类值表示指定序号 Link 的时间。

5. linkCtrl.setLinkTime <link\_control> <index\_integer> <time>

为指定索引的 Link 设置时间。指定的 Link 必须已事先存在。

6. linkCtrl.addLink <link\_control> <node> <time>

在指定时间将指定 Node 对象作为一个 Link 添加到<link-control>。

7. linkCtrl.deleteLink <link\_control> <index\_integer>

删除指定 Link。

### 12.10.13 List Controller (列表控制器)

#### 类型

float\_list: FloatController

```
point3_list: Point3Controller
position_list: PositionController
rotation_list: RotationController
scale_list: ScaleController
```

### 构造函数

```
float_list...
point3_list...
position_list...
rotation_list...
scale_list...
```

### 属性

List 类控制器的属性包括这些控制器里的子控制器和一个属性.available。属性.available 有一个子属性.controller，这个子属性包含一个“投影”控制器：如果要向 List 类控制器里添加一个子控制器，将其赋给属性.available.controller 就可以了。下面的脚本创建一个 List 类控制器，并向它添加一些子控制器。

```
p=float_list()           -- 创建一个 List Controller
showProperties: p         -- 显示其属性
p.available.controller   -- 投影控制器
p1=bezier_float()        -- 创建一个新的 Bezier Float Controller
p.available.controller=p1 -- 将其添加到 List Controller
p2=bezier_float()        -- 再创建一个新的 Bezier Float Controller
p.available.controller=p2 -- 再将其添加到 List Controller
getpropnames p            -- 显示 List Controller 属性名
showproperties p          -- 显示 List Controller 的属性
p.numSubs                 -- List Controller 里 subAnims 的个数
getsubanimnames p         -- 显示 List Controller 里 subAnim 的名称
p[2].object               -- 获取第二个 Bezier Float Controller
```

### 输出：

```
Controller:Float_List      -- 第 1 行结果
.available: float          -- 第 2 行结果
OK                         -- 第 3 行结果
Controller:                -- 第 4 行结果
Controller:Bezier_Float    -- 第 5 行结果
Controller:Bezier_Float    -- 第 6 行结果
Controller:Bezier_Float    -- 第 7 行结果
Controller:Bezier_Float    -- 第 8 行结果
#(#bezier_float, #available) -- 第 9 行结果
.bezier_float: float        -- 第 9 行结果
.bezier_float: float        -- 第 9 行结果
.available: float          -- 第 9 行结果
OK                         -- 第 9 行结果
3                          -- 第 10 行结果
```

```
#(#bezier_float, #bezier_float, #available) --第 11 行结果
Controller:Bezier_Float --第 12 行结果
```

如果要显示 List 控制器里的全部子控制器的名称，用户必须用上例中最后一行的方法：getsubAnimNames()。

目前尚没有一种方法可以从 List 控制器里移走一个控制器。如果用户要这样做，最简单的办法就是：创建一个新 List\_Controller，将旧 list Controller 里其他的子控制器复制到新控制器里，然后用新控制器代替旧控制器。下面的脚本就是这方面的一个例子：

```
--从旧 Controller 创建新的 List Controller，删除指定子控制器，
--返回新的 ListController。
fn deleteFromListController list_cont index =
(
--用传送来的 List Controller 的类创建一个新 List Controller
new_cont=execute((classof list_cont as string)+ "()")
--对 List Controller 的每一 SubAnim 循环（最后一个除外，因为是
--投影控制器）
for i=1 to(list_cont.numSubs-1)do
--如果不是要删除的子控制器
if i != index do
--复制子控制器（包含在 subAnim 的 object 属性里）到新
--List Controller 里
    new_cont.available.controller=list_cont[i].object
--返回新 list Controller
    return new_cont
)
--调用实例，假设属性 obj.pos 已经被赋予 List Controller
obj.pos.controller = deleteFromListController obj.pos.controller 2
```

### 方法

1. listCtrl.getName <list\_controller> <index\_integer>

返回指定子控制器的名称。

2. listCtrl.setName <list\_controller> <index\_integer> <string>

为指定子控制器设置名称。如果变量<string>为一个空字符串（“”），使用默认的控制器名称。

3. listCtrl.getActive <list\_controller>

返回指定 List Controller 当前活动的子控制器的序号。

4. listCtrl.setActive <list\_controller> <index\_integer>

将指定子控制器设置为当前活动的子控制器。

## 12.10.14 LOD\_Controller: FloatController

### 构造函数

LOD\_Controller 不能由 MAXScript 来创建，而仅能用 Utilities 面板里的 Level of Detail 工具进行创建。

## 属性

LOD\_Controller 没有额外的属性。

### 12.10.15 LookAt: Matrix3Controller (注视约束控制器)

#### 构造函数

LookAt...

#### 属性

<LookAt>.position: Bezier\_Position, Position Controller

<LookAt>.roll\_angle: Bezier\_Float, Roll Controller

<LookAt>.scale: Bezier\_Scale, Scale Controller

目前尚没有办法用来直接存取 LookAt 控制器里的目标对象，如果要设置目标对象，要先把 LookAt 控制器赋给一个节点对象，然后将该对象的.LookAt 属性设为目标对象。如果该节点对象的 Transform 控制器不是一个 LookAt 控制器，当把目标对象赋给该节点对象的.LookAt 属性，系统会自动把一个 LookAt 控制器赋给节点对象。如下脚本：

```
$Box01.lookAt=$Box02
```

与下面两句脚本等效：

```
$Box01.transform.controller=lookAt()  
$Box01.lookAt=$Box02
```

### 12.10.16 MasterBlock: MasterBlockController

#### 构造函数

MasterBlock 不能在 MAXScript 里创建，而只能在 Track View 视窗的 Global Tracks 轨迹下的 Block Control 子轨迹里创建。如果当前有多个 MasterBlock，用户必须要用 trackviewnodes.global-tracks.block\_control[i] 的格式进行来存取，因为所有 MasterBlock 都有一个同样的名称：MasterBlock。

#### 属性

一旦一个 MasterBlock 在 Track View 里被创建，下面的属性可以被存取：

<MasterBlock>.blend: Float

**注意** 如果向 MasterBlock 里添加一个轨迹，一个 Block Controller 会同时添加到 MasterBlock 里，该 Block Controller 的名称就是用户为 Block 指定的名称。然后该 Block Controller 就可以作为 MasterBlock 的一个属性进行存取。

下面的例子中，假设已有一个名为 BoxHeight 的 MasterBlock，它包含一个 Height Controller，用来控制 Box 对象：

```
mb=trackviewnodes.global_tracks. \
```

```
block_control[1]          --返回 SubAnim:MasterBlock  
getpropnames mb           --返回#(#Blend, #height)  
h=mb.height                --返回 SubAnim:height  
h.object                   --返回 Controller:Block  
getpropnames h              --返回#(#Box01_height)  
h.Box01_height             --返回 157.083, Box 对象的 height 属性  
h.Box01_height.controller  --返回 Controller:Bezier_Float
```

### 12.10.17 Motion Capture Controllers (运动捕捉控制器)

#### 类型

```
float_motion_capture: FloatController  
point3_motion_capture: Point3Controller  
position_motion_capture: PositionController  
rotation_motion_capture: RotationController  
scale_motion_capture: ScaleController
```

#### 构造函数

```
float_motion_capture...  
point3_motion_capture...  
position_motion_capture...  
rotation_motion_capture...  
scale_motion_capture...
```

#### 属性

<motion\_capture>.data Controller

Data 控制器类型取决于 Motion 控制器的数据类型，该控制器是一个典型的 Bezier 类控制器。

### 12.10.18 Noise Controllers (噪波控制器)

#### 类型

```
noise_float: floatController  
noise_point3: point3Controller  
noise_position: positionController  
noise_rotation: rotationController  
noise_scale: scaleController
```

#### 构造函数

```
noise_float...  
noise_point3...
```

noise\_position...  
noise\_rotation...  
noise\_scale...

### 属性

属性名称	数据类型	默认值	说明
下面的属性适用于所有 Noise Controllers			
<noise_controller>.seed	Integer	0	存取噪波控制器的根值
<noise_controller>.frequency	Float	0.5	存取噪波控制器的频率
<noise_controller>.fractal	Boolean	True	存取是否打开控制器的分形噪波设置
<noise_controller>.roughness	Float	0.0	存取噪波控制器的粗噪度
<noise_controller>.rampin	Time	0f	存取噪波控制器淡入时间
<noise_controller>.rampout	Time	0f	存取噪波控制器淡出时间
下面的属性仅适用于 Noise float Controller			
<noise_controller>.noise_strength	Float	50	可动画
<noise_controller>.positive	Boolean	False	存取是否打开控制器的正值约束设置
下面的属性仅适用于 Position、Point3、Rotation 和 Scale Controller			
<noise_controller>.noise_strength	Point3	默认值: Position 和 point3 类: [50,50,50]; Rotation 类: [0.785398,0.785398,0.785398]; Scale 类: [0.5,0.5,0.5] 可动画	
<noise_controller>.x_positive	Boolean	False	存取是否打开控制器的 X 轴向的正值约束设置
<noise_controller>.y_positive	Boolean	False	存取是否打开控制器的 Y 轴向的正值约束设置
<noise_controller>.z_positive	Boolean	False	存取是否打开控制器的 Z 轴向的正值约束设置

#### 注意

当用户使用这些属性时，必须显式地列出控制器的名称，如：

```
$Box01.position.controller.x_strength = 100
```

或者：

```
c = $Box01.rotation.controller
c.seed = random 0 100
c.fractal = off
c.frequency = random 0.1 0.2
```

## 12.10.19 On\_Off: FloatController (开关控制器)

## 构造函数

On\_off...

## 属性

&lt;on\_off&gt;.keys MAXKeyArray

## 和关键帧有关的属性

&lt;key&gt;.time Time 或 Number (以帧为单位) 类值, 只读

&lt;key&gt;.selected Boolean, 设置关键帧是否被选择, 可读写

## 12.10.20 Path: PositionController (路径约束控制器)

## 构造函数

path...

## 属性

属性名称	数据类型	默认值	说明
<path>.allowUpsideDown <path>.Allow_Upside_Down	Boolean	False	当设置为 True 时, 可以避免沿垂直方向的路径运动时对象发生翻转
<path>.axis	Integer	0	0: X; 1: Y; 2: Z
<path>.axisFlip <path>.Axis_Flip	Boolean	False	将对象的方向沿路径翻转 180°
<path>.bank	Boolean	False	当设置为 True 时, 对象通过路径的曲线段时会倾斜
<path>.bankAmount <path>.Bank_Amount	Float	0.5	对象倾斜的大小, 其数值的正负决定了倾斜的方向。可动画
<path>.constantVel <path>.Constant_Velocity	Boolean	False	当设置为 True 时, 对象会以同一速度通过路径。否则, 对象沿路径的速度取决于路径上顶点之间的距离
<path>.follow	Boolean	False	设置是否在对象跟随轮廓运动的同时将对象指定给轨迹
<path>.loop	Boolean	False	当设置为 True 时, 对象到达路径的终点后, 会自动回到路径的起点循环移动
<path>.path	Node	undefined	指定选定对象的运动路径
<path>.pathlist	Array	#()	对象将沿本属性指定的 resultant 路径运动, 该路径由本数组里各路线的加权平均形成。每一条路径的权重由属性<path>.weightlist 指定
<path>.percent	Float	0.0	对象沿路径放置的百分比。可动画, 百分比
<path>.relative	Boolean	False	如果设为 True, 由路径控制的对象会保持从其原始位置到路径起点之间的位置偏移量
<path>.smoothness	Float	0.5	控制对象在经过路径中转弯时翻转角度改变

			的快慢程度。较小的值使对象对曲线的变化反应更灵敏，而较大的值则会消除突然的转折。此默认值对沿曲线的常规阻尼是很适合的。当值小于 2 时往往会使动作不稳定，但是值在 3 附近时对模拟出某种程度的真实的不稳定很有效果
<path>.weightlist	Array	#()	对应于属性<path>.Pathlist 的一个数组，定义了每一个路径的权重

下面的脚本显示了如何创建一个 Path Controller，并为其设置了动画。

```

thePath=circle radius:50           --创建一个 shape 对象作为路径
theObj=cone radius1:6 radius2:0 height:15   --创建一个对象沿路径运动
--将一个 path Controller 赋给对象
theObj.pos.controller=path follow:true
PosCont=theObj.pos.controller      --获取 path Controller
PosCont.path=thePath              --将 shape 对象设为路径
PosCont.axis=2                   --将 local 坐标系的 z 轴设为沿 path 创建关键帧
animate on
(  at time 30 PosCont.percent=25    --30 帧时，在路径的 25% 的位置上
  at time 100 PosCont.percent=95   --100 帧时，在路径的 95% 的位置上
)

```

### 12.10.21 PRS: Matrix3Controller (PRS 控制器)

#### 构造函数

PRS...

#### 属性

<PRS>.position: Bezier\_Position, position Controller  
<PRS>.rotation: TCB\_Rotation, rotation Controller  
<PRS>.scale: Bezier\_Scale, scale Controller

**注意** 我们可以将一个 Matrix3 值赋给一个 PRS Controller 的.value 属性，MAXScript 会自动将与该值对应的 Position、Rotation 和 Scale 分量分别赋给三个对应的子属性。

### 12.10.22 Reactor Controller (连锁反应控制器)

#### 类型

float\_Reactor: floatController  
point3\_Reactor: point3Controller  
position\_Reactor: positionController  
rotation\_Reactor: rotationController  
scale\_Reactor: scaleController

### 构造函数

float\_Reactor...  
 point3\_Reactor...  
 position\_Reactor...  
 rotation\_Reactor...  
 scale\_Reactor...

### 方法

1. createReaction <reactor\_controller>

为指定<reactor\_controller>创建一个新的连锁反应。

2. deleteReaction <reactor\_controller> <index\_integer>

删除指定连锁反应。变量<index\_integer>为要删除的连锁反应序号。各连锁反应的序号与其创建顺序相同，用户可以在 Reactor Parameters 对话框里看到。

3. reactTo <reactor\_controller>(<controller> | <node> )

为指定<reactor\_controller>设置连锁反应控制器。用户可以指定一个<controller>，也可以指定一个 Node 对象，表示反应针对该对象的 World 空间下的位置。

4. getReactionCount <reactor\_controller>

返回指定<reactor\_controller>的连锁反应数目。

5. selectReaction <reactor\_controller> <index\_integer>

选择指定的连锁反应。

6. getSelectedReactionNum <reactor\_controller>

返回当前选中的连锁反应数目。

7. getReactionFalloff <reactor\_controller> <index\_integer>

返回指定连锁反应的衰减值。

8. setReactionFalloff <reactor\_controller> <index\_integer> <float>

为指定连锁反应设置衰减值。

9. getReactionInfluence <reactor\_controller> <index\_integer>

返回指定连锁反应的影响范围。

10. setReactionInfluence <reactor\_controller> <index\_integer> <float>

为指定连锁反应设置影响范围。

11. getReactionStrength <reactor\_controller> <index\_integer>

返回指定连锁反应的强度。

12. setReactionStrength <reactor\_controller> <index\_integer> <float>

为指定连锁反应设置强度。

13. getReactionState <reactor\_controller> <index\_integer>

返回指定连锁反应的状态。

14. setReactionState <reactor\_controller> <index\_integer> <value>

为指定连锁反应设置状态。

15. `getReactionValue <reactor_controller> <index_integer>`

返回指定连锁反应的值。返回值的数据类型与连锁反应针对的 Controller 的类型一致。

16. `setReactionValue <reactor_controller> <index_integer> <value>`

为指定连锁反应设置反应值。

17. `setReactionName <reactor_controller> <index_integer> <string>`

为指定连锁反应设置名称。

18. `getReactionName <reactor_controller> <which>`

返回指定连锁反应的名称。

示例：

```
b1 = Box name:"Box01" pos:[-32.5492,-21.2796,0]
b2 = Box name:"Box02" pos:[51.3844,-17.2801,0]
--建立场景，创建两个 Box 对象
animate on at time 100 b1.pos = [-48.2522,167.132,0]
--为其中一个 Box 对象的.pos 属性设置动画
cont = b2.pos.controller = position_Reactor()
--生成一个 reactor，拾取连锁反对象，创建一个 reaction
reactTo cont b1.pos.controller
--可以设置 reactor 针对 Controller
reactTo cont b1
--也可以设置 reactor 针对一个 node
slidertime = 100
createReaction cont
slidertime = 50
createReaction cont
deleteReaction cont 3
setReactionState cont 2 [65.8385,174.579,0]
selectReaction cont 1
setReactionInfluence cont 1 100
setReactionStrength cont 1 1.2
setReactionFalloff cont 1 1.0
setReactionValue cont 1 [-40.5492,-20.0,0]
setReactionName cont 1 "My Reaction"
--设置 reaction 参数
getReactionInfluence cont 1
getReactionStrength cont 1
getReactionFalloff cont 1
getReactionState cont 1
getReactionValue cont 1
getSelectedReactionNum cont
getReactionCount cont
getReactionName cont 1
--存取 reaction 参数
```

### 12.10.23 Script Controller (脚本控制器)

#### 类型

```
float_script: FloatController
point3_script: Point3Controller
position_script: PositionController
rotation_script: RotationController
scale_script: ScaleController
```

#### 构造函数

```
float_script...
point3_script...
position_script...
rotation_script...
scale_script...
```

#### 属性

<script\_controller>.script: String  
存取脚本控制器的脚本文本。如：

```
$foo.position.controller=position_script()
$foo.position.controller.script = "$baz.pos - [20,20,35]"
```

脚本控制器和表达式控制器有点类似，都会出现一个对话框。用户可以向其中输入脚本，并计算控制器的值。与表达式控制器相比，脚本控制器有如下优点：

1. 在脚本控制器里可以使用 MAXScript 的全部语法，包括循环、定义函数、路径等；
2. 在脚本控制器里几乎场景里所有对象的任何属性都可以用来计算控制器的值，包括 Mesh 对象的顶点、任一关键帧处属性值和其他可在表达式控制器里可用但不可动画的属性；
3. 在脚本控制器里可以使用 MAXScript 全局变量来与其他 3ds max 里的控制器和脚本进行沟通和协调。

当用户将一个脚本控制器赋给某一个参数时，可以在 Track View 视窗里用右击鼠标或窗口工具栏的 Properties 按钮打开一个 Properties 对话框，该对话框包含两个文本框和几个按钮，详细说明如下：

- ◆ **Script** 文本框 用户可以向其中输入脚本。详细说明参见下面章节；
- ◆ **Result** 文本框 显示脚本的一次求值结果或脚本里的语法错误导致的错误信息；
- ◆ **Evaluate** 按钮 引起对脚本的一次求值，并将结果显示在 Result 文本框里。求值针对 3ds max 时间滑标当前显示的时间位置进行；
- ◆ **Load/Save** 按钮 载入或存储脚本文件；
- ◆ **Close** 按钮 编译并检测 Controller 里的脚本，如果一切正常，关闭 Properties 对话框；如果在检测中发生任何问题，系统会提示用户“Whether you really want to close

the Box with an incorrect script”, 如果用户按 Yes, 这时 3ds max 如果正对该控制器求值, 控制器会得到一个空值 (0、[0, 0, 0]等)。

### 写入控制器脚本

3ds max 将用户在 Script 文本框里输入的文本看作一个 MAXScript 块表达式, 因此用户可以向文本框里输入任意多的表达式, 且行数没有限制, 系统会对它们依次进行求值, 最后一个表达式的值会作为控制器的值。这个值的数据类型必须与控制器的类型相对应, 如 Float 类控制器必须返回 Float 值, Position 类控制器返回 Point3 值, Rotation 类控制器返回 Quat 值。

由脚本控制器返回的值可能与在 Track View 或命令面板里看到的不一致, 也可能与在 MAXScript 里存取控制器对应的属性值不一致。有一些 MAXWrapper 属性定义了一个缩放因子, 这个缩放因子会在系统存取控制器值时被施加。如属性.slice\_to 和.slice\_from 显示单位为度, 而实际存储在控制器里的值是以弧度为单位。这些缩放因子是 MAXWrapper 属性的内在特性, 而不是控制器的内在特性。在 3ds max 和 MAXScript 里都会在存取属性时自动施加这些缩放因子, 所以一般来讲, 这些缩放因子对用户而言是不可见的, 这样在脚本控制器或表达式控制器中返回的数据都必须是未经缩放的, 因为 3ds max 会为这些输出值施加缩放系数。在 MAXWrapper 类的说明中, 显示了将施加于属性的缩放因子, 如果没有列出缩放因子, 表示该属性没有缩放因子, 比如 Capsule 类的部分属性说明如下:

- ◆ <Capsule>.radius Float, 默认值: 0.0, 可动画
- ◆ <Capsule>.height Float, 默认值: 0.0, 可动画
- ◆ <Capsule>.slice\_from Float, 默认值: 0.0, 可动画, Angle
- ◆ <Capsule>.slice\_to Float, 默认值: 0.0, 可动画, Angle

对脚本控制器而言, .radius 和.height 属性返回的值是一个未经缩放的值; .slice\_from 和.slice\_to 属性, 其单位为弧度, 缩放类型、存储值意义和缩放值如下:

- ◆ Angle 角度, 以弧度为单位存储, 输出时乘以 57.29578
- ◆ Percentage 百分数, 以小数格式 (0~1) 存储, 输出时乘以 100

如果属性类型为 Color, 控制器输出时会自动乘以 255。如果一个 Point3 类脚本控制器被赋给这样的属性, 每一个元素值都应该以小数格式 (0~1) 存储。用户应注意: 在 MAXScript 里将 Color 值转换成一个 Point3 值不会自动施加缩放因子, 如将系统变量 Red 转为 Point3, 会返回 [255, 0, 0]。这样, 如果将一个 Point3 类脚本控制器的输出值作为 Color 值, 用户必须自己对它们乘以 255。

脚本控制器的正文由任何合法的 MAXScript 表达式组成, 这些表达式最后计算出合适的数据类型, 并可以包含全局和局部变量、函数和结构定义。脚本在它自己的作用域里进行编译, 局部变量仅在脚本控制器的作用域里可见, 且是基于 Heap 的。

这些局部变量与一般的局部变量 (基于 Stack 的) 有点不一样。一般的局部变量在当前作用域下可见, 并有与文本作用域相同的生存期。而基于 Heap 的局部变量同样在当前作用域可见, 但是其生存期却与定义它的脚本里最高一级表达式的生存期一样。在脚本控制

器里的局部变量在脚本第一次被执行时创建，并被放置在 Heap 里保持其值不变，直到用户重新定义脚本。这意味着用户可以在一次执行求值时将值存在局部变量里，而在下次执行脚本求值时这些值仍然存在。用户不能在 Listener 窗口或别的脚本工具里存取脚本控制器的局部变量，因为它们不在脚本控制器的作用域里执行。

用户还可以用命令 `return <expr>` 来中途退出脚本。脚本控制器总是在一个特定的时间值下被求值。这个时间值可能是某个关键帧的时间。

在脚本控制器里，被求值的时间总是会自动形成一个 `at time` 关联语句，这样用户存取属性值时都会得到当前 Controller 执行时刻的正确的值。这意味着在脚本里用户不用作任何特别的设置，就可以在正确的时间下进行操作。用户可以用标准的 MAXScript 变量 `CurrentTime` 来获取脚本求值时的时间。用户还可以引用别的时间下场景的属性值，只需要在脚本里显式地使用 `at time` 表达式。记住在 MAXScript 里，我们可以将一个命令写成多行。

下面是一个脚本控制器的例子，当场景对象发生移动时，它会让对象保持在其他对象组成的中心位置上。

```
local pos = [0,0,0]
for o in objects where o != $foo do
    pos += o.pos
pos / (objects.count - 1)
```

上面脚本计算除当前对象之外（本例中为 `foo`）的全部对象的中心位置：首先设置一个局部变量，该变量将遍历所有对象（除 `foo` 以外）来累积一个总的位置矢量，然后在最后一行计算其平均值，并将值作为脚本的最后结果。

下面的 Position 类脚本会将一个对象附着在指定对象的最高顶点处。

```
local high_index = 1, high_z =(getVert $foo 1).z
for i in 2 to $foo.numVerts do
    if(getVert $foo i).z > high_z then
    (
        high_index = i
        high_z =(getVert $foo i).z
    )
getVert $foo high_index
```

上面脚本对 `foo` 的每个顶点循环，并记录下 Z 坐标最高顶点的序号，然后返回该顶点的坐标值。

**注意** 如果用户直接在场景下修改对象，与其相应的脚本控制器不会自动刷新。

如果用户移动时间滑标或将上述的修改设置成动画，然后按下 Play 按钮，这些修改方式会自动反映出来。因为在脚本里可以引用其他对象，这种引用可能是非直接的方式或因果关系方式，对 MAXScript 而言，不可能自动识别与脚本相关的对象。

#### 12.10.24 Slave\_Control: Matrix3Controller (附属控制器)

`Slave_Control` 用于 RingArray 系统里 Box 对象的 Transform 控制器。在 3ds max 里用户不能用 MAXScript 来创建 RingWave 系统，但是可以在 MAXScript 里进行存取。一个

RingArray 系统实际由两类控制器组成：Master 控制器和 Slave\_Control 控制器。Slave\_Control 控制器用于 RingWave 系统里 Box 对象的 Transform 控制器；而 Master 控制器用来控制单个的 Slave\_Control Controller。改变 RingWave 系统某一 Slave\_Control 的属性会改变系统里所有 Slave\_Control 的同一属性的值。

#### 属性

```
<Slave_Control>.radius: Float, 默认值: 100.0, 可动画
<Slave_Control>.cycles: Float, 默认值: 3.0, 可动画
<Slave_Control>.amplitude: Float, 默认值: 20.0, 可动画
<Slave_Control>.phase: Float, 默认值: 1.0, 可动画
```

### 12.10.25 Slave Controller (附属控制器)

#### 类型

```
slavefloat: FloatController
slave_point3: Point3Controller
slavepos: PositionController
slaverotation: RotationController
slavescale: ScaleController
```

#### 构造函数

Slave Controller 不能在 MAXScript 里创建，只能通过在 Track View 视窗里 Global 轨迹的 Block Control 下创建一个 Master Block 来创建。

### 12.10.26 Surface\_position: PositionController (表面约束控制器)

#### 构造函数

Surface\_position...

#### 属性

1. <Surface\_position>.surface: Node, 默认值: undefined  
存取表面约束控制器的目标约束对象。
2. <Surface\_position>.u: Float, 默认值: 0.0, 可动画  
存取当前对象在目标约束对象 U 向上的位置。
3. <Surface\_position>.v: Float, 默认值: 0.0, 可动画  
存取当前对象在目标约束对象 V 向上的位置。
4. <Surface\_position>.align: Integer, 默认值: 1  
指定对齐方式:

◆ 1 无对齐

- ◆ 2 与 U 轴对齐
- ◆ 3 与 V 轴对齐

5. <Surface\_position>.flip: Boolean, 默认值: False

## 12.10.27 TCB Controller (TCB 控制器)

### 类型

TCB\_float: FloatController  
 TCB\_point3: Point3Controller  
 TCB\_position: PositionController  
 TCB\_rotation: RotationController  
 TCB\_scale: ScaleController

### 构造函数

TCB\_float...  
 TCB\_point3...  
 TCB\_position...  
 TCB\_rotation...  
 TCB\_scale...

### 属性

<controller>.keys MAXKeyArray

<tcb\_rotation>.rotWindup : boolean: 可读可写, 仅用于 TCB\_rotation 型控制器

### 和 TCB 控制器关键帧有关的属性

对 TCB 控制器关键帧而言, 除通用属性以外, 下面几个额外的属性可被存取:

属性名称	数据类型	默认值	说明
<key>.time	Time		Time 类值或数值 (以帧为单位)
<key>.selected	Boolean		指定关键帧是否被选中
<key>.value	不定		数据类型取决于包含它控制器
<key>.tension	Float	25.0	存取关键帧的张力
<key>.continuity	Float	25.0	存取关键帧的连续性
<key>.bias	Float	25.0	存取关键帧的偏移值
<key>.easeTo	Float	0.0	指定当运动接近关键帧时, 减缓运动的速度
<key>.easeFrom	Float	0.0	指定当运动离开关键帧时, 减缓运动的速度

## 12.10.28 Waveform\_Float: FloatController (波形控制器)

### 构造函数

waveform\_float...

### 属性

除通用属性以外，Wavefrom\_Float 控制器没有额外的属性。

## 12.10.29 XYZ Controller (XYZ 控制器)

### 类型

Color\_RGB: Point3Controller  
 Euler\_XYZ: RotationController  
 Local\_Euler\_XYZ: RotationController  
 Point3\_XYZ: Point3Controller  
 Position\_XYZ: PositionController  
 ScaleXYZ: ScaleController

### 构造函数

Color\_RGB...  
 Euler\_XYZ...  
 Local\_Euler\_XYZ...  
 Point3\_XYZ...  
 Position\_XYZ...  
 ScaleXYZ...

### 属性

XYZ Controller 的属性名与特定的控制器类型有关：

#### 1. Color\_RGB (颜色 RGB 控制器):

<Color\_RGB>.r: Float, 默认值: 0.0, 可动画  
 <Color\_RGB>.g: Float, 默认值: 0.0, 可动画  
 <Color\_RGB>.b: Float, 默认值: 0.0, 可动画

#### 2. Euler\_XYZ (Euler XYZ 旋转控制器):

<Euler\_XYZ>.x\_rotation: Float, 默认值: 0.0, 可动画  
 <Euler\_XYZ>.y\_rotation: Float, 默认值: 0.0, 可动画  
 <Euler\_XYZ>.z\_rotation: Float, 默认值: 0.0, 可动画  
 <Euler\_XYZ>.axisOrder: Integer, 默认值: 1

存取轴向角施加的顺序，有效值及其代表的意义如下：

◆ 1 XYZ

- ◆ 2 XZY
- ◆ 3 YZX
- ◆ 4 YXZ
- ◆ 5 ZXY
- ◆ 6 ZYX
- ◆ 7 XYX
- ◆ 8 YZY
- ◆ 9 ZXZ

3. Local\_Euler\_XYZ (局部 Euler XYZ 控制器):

<Local\_Euler\_XYZ>.local\_x\_rotation: Float, 默认值: 0.0, 可动画

<Local\_Euler\_XYZ>.local\_y\_rotation: Float, 默认值: 0.0, 可动画

<Local\_Euler\_XYZ>.local\_z\_rotation: Float, 默认值: 0.0, 可动画

<Local\_Euler\_XYZ>.axisOrder: Integer, 默认值: 1

存取轴向角施加的顺序, 有效值及其代表的意义同<Euler\_XYZ>.axisOrder。

4. Point3\_XYZ (Point3 XYZ 控制器):

<Point3\_XYZ>.x: Float, 默认值: 0.0, 可动画

<Point3\_XYZ>.y: Float, 默认值: 0.0, 可动画

<Point3\_XYZ>.z: Float, 默认值: 0.0, 可动画

5. Position\_XYZ (位置 XYZ 控制器):

<Position\_XYZ>.x\_position: Float, 默认值: 0.0, 可动画

<Position\_XYZ>.y\_position: Float, 默认值: 0.0, 可动画

<Position\_XYZ>.z\_position: Float, 默认值: 0.0, 可动画

6. ScaleXYZ (缩放 XYZ 控制器):

<ScaleXYZ>.x\_scale: Float, 默认值: 0.0, 可动画

<ScaleXYZ>.y\_scale: Float, 默认值: 0.0, 可动画

<ScaleXYZ>.z\_scale: Float, 默认值: 0.0, 可动画

## 方法

getXYZControllers <controller>

返回一个数组, 表示指定控制器下的 X、Y、Z 子控制器。

# 第 13 章 Atmospheric (环境效果)

Atmospheric 类可用来在 MAXScript 里设置体积类渲染效果。用户可以创建一些 Atmospheric 类对象如爆炸和雾，可以对其属性和 gizmo 数组进行存取、修改操作。

直接由类 Atmospheric 派生的子类在本书 13.2 节中详细描述。

Atmospheric 类是由 MAXWrapper 类派生而来的，并继承了 MAXWrapper 类的属性和方法。

## 13.1 Atmospheric 类通用属性和方法

Atmospheric 类对象有下面一些通用的属性和方法。

### 属性

1. <atmos>.name: String, 默认值: 不定

指定对象在 Environment 对话框列表框里显示的名称，如果在创建对象时没有指定参数 name:，类名会作为默认的对象名。

2. <atmos>.numGizmos: Integer, 只读

返回赋给指定对象的 gizmo 数目。用方法 getGizmo() 获取某一 gizmo 时可能会返回 undefined，表示该 gizmo 在被添加之后又被删除了。Atmospheric 包含了一个 gizmo 数组，但并不是每一数组元素都会被用到，属性.numGizmos 是数组的长度，而不是所用到的数组元素的数目。

### 方法

1. isActive <atmos>

如果指定 Atmospheric 对象被设为活动的，返回 True；否则，返回 False。

2. getGizmo <atmos> <index\_integer>

从 Atmospheric 里获取指定索引的 gizmo 对象，gizmo 对象可能是一个 Light 对象或 Atmospheric Helper 对象，序号从 1 开始。

3. deleteGizmo <atmos> <index\_integer>

删除指定序号的 gizmo。

4. appendGizmo <atmos> <node>

将指定对象添加到指定 Atmospheric 的 gizmo 数组里。对象类型必须与 Atmospheric 匹配，如对 VolumeLight 必须是 Light 对象；对 Fire，必须为 Helper 对象。

5. setActive <atmos> <boolean>

设置指定 Atmospheric 状态是否为活动的。

### 相关方法

用户可以在全局的渲染环境里使用下面几个函数来管理 Atmospherics 清单：

1. addAtmospheric <atmos>

将指定 Atmospheric 添加到 Atmospherics Effect 清单里。

2. getAtmospheric <index\_integer>

获取 Atmospherics Effect 清单里指定序号的 Atmospheric。

3. setAtmospheric <index\_integer> <atmos>

将 Atmospheric Effects 清单里指定索引的 Atmospheric 设为指定 Atmospheric。如果调用本方法时 Atmospherics Effect 对话框正被显示，显示的 Atmospherics Effect 清单不会被刷新。

4. deleteAtmospheric <index\_integer>

删除 Atmospherics Effect 清单里指定序号的 Atmospheric。

5. editAtmosphere <atmos> <node>

打开 Environment 对话框，并打开指定的 Atmospheric 卷展栏（如果它已被添加到 Environment 清单里）。如果参数<node>为 Atmospheric 的 gizmo，该 gizmo 在 gizmo 清单里被选中。

6. editAtmospheric <atmos> [ gizmo: <node> ]

打开 Environment 对话框，并打开指定 Atmospheric 的卷展栏（如果它已被添加到 Environment 清单里）。如果有指定可选参数 gizmo:，而且指定 Node 对象为 Atmospheric 的 gizmo，该 gizmo 在 gizmo 清单里被选中。

**注意** 和大多数 MAXWrapper 对象不一样，用户不能复制一个 Atmospheric 类对象。

例子：

```
atmos = combust density:20 outer_color:red
appendGizmo atmos $SphereGizmo01
```

下面的 3ds max 系统变量可以存取当前的渲染环境（详细说明参见 4.5.2 节“3ds max 系统变量”）。

```
backgroundColor
backgroundColorController
ambientColor
ambientColorController
environmentMap
useEnvironmentMap
lightTintColor
lightTintColorController
lightLevel
lightLevelController
numAtmospherics
```

下面的方法用来改变当前的渲染环境设置:

1. `getUseEnvironmentMap()`

`setUseEnvironmentMap <boolean>`

存取 Rendering | Environment 对话框下 Environment 选项卡里复选框 Use Map 的设置。

2. `getBackGroundController()`

返回 Rendering | Environment 对话框下 Environment 选项卡里的 Background Color 的控制器。

3. `setBackGround <time> <point3>`

为 Rendering | Environment 对话框下 Environment 选项卡里的 Background Color 的控制器在指定时间设置颜色, 数据类型为 Point3, 每一元素的取值范围为 0~255。

4. `setBackGroundController( <color_controller> | <point3_controller> )`

将指定控制器赋给 Rendering | Environment 对话框下 Environment 选项卡里的 Background Color 的控制器。

## 13.2 Atmospheric Effect (环境效果类型)

下面列出了所有 3ds max 环境效果类型:

- ◆ Fire\_Effect (火焰环境效果)
- ◆ Fog (雾效果)
- ◆ Volume\_Fog (体积雾环境效果)
- ◆ Volume\_Light (体积环境光效果)

## 13.3 Fire\_Effect: Atmospheric (火焰环境效果)

### 构造函数

`Fire_Effect...`

### 属性

属性名称	数据类型	默认值	说明
<code>&lt;Fire_Effect&gt;.Inner_Color</code>	Color	(color 252 202 0)	设置火焰效果最密集部分的颜色, 对一般的火焰而言, 这个颜色代表了火焰的最热部分。可动画
<code>&lt;Fire_Effect&gt;.Outer_Color</code>	Color	(color 225 30 30)	设置火焰效果的最稀薄部分的颜色。对一般的火焰而言, 这个颜色代表了火焰的边缘部分。火焰效果的颜色为从 Inner_color 开始的渐变, 密集区使用 Inner_color 颜色, 然后慢慢过渡到 Outer_color 的颜色, 到火焰的边缘使用 Outer_color 的颜色。可动画

(续表)

属性名称	数据类型	默认值	说明
<Fire_Effect>.Smoke_Color	Color	默认值: (color 25.5 25.5 25.5) 如果有使用 Explosion 选项, 本属性设置烟雾的颜色。如果没有选择 Explosion 选项, 本设置无效。可动画	
<Fire_Effect>.Flame_Type	Integer	0	设置火焰类型: 0: Fire Ball (创建圆形膨胀火焰, 本选项适合于爆炸) 1: Tendril (创建有方向性的锐角火舌火焰, 火焰的方向指向火焰装置的 Z 轴, 适合用来创建营火状的火焰)
<Fire_Effect>.Stretch	Float	1.0	沿火焰装置的 Z 轴方向缩放火焰。本设置主要与 Tendril 类型火焰配合使用, 但用户也可以将它用在 Fireball 类型火焰上, 会产生椭圆形的火球。如果拉伸参数小于 1.0, 可以压缩火焰, 使火焰变粗变短, 如果拉伸参数大于 1.0, 可以拉伸火焰, 使火焰变长变细。可动画
<Fire_Effect>.Regularity	Float	0.2	取值范围为 1.0~0.0。如果其值为 1.0 时, 火焰效果完全填充到装置中, 在装置的边缘附近火焰效果逐渐消退, 但其形状仍然非常清晰。如果其值为 0.0, 火焰效果非常不规则, 偶尔会触及装置的线框, 但通常火焰都会很小。可动画
<Fire_Effect>.Flame_Size	Float	35.0	设置火焰装置内部单个火焰的尺寸, 火焰装置的大小也会影响火焰的大小, 装置越大时火焰尺寸就要越大一些。其取值范围在 15.0~30.0 之间火焰效果最好。对 Fireball 类型火焰本参数要设得大一些。而对 Tendril 火焰而言, 本参数要设得小一些。可动画
<Fire_Effect>.Flame_Detail	Float	3.0	控制火焰颜色改变大小及火焰边缘的清晰度。较小的值会创建平滑、模糊的火焰, 且渲染速度会快一些, 较大的值会创建形状清晰的火焰效果, 但渲染速度会慢一些。可动画
<Fire_Effect>.Density	Float	15.0	设置火焰的不透明度和亮度。火焰装置的大小会影响火焰的密度。较大的装置如果火焰密度设置相同与较小的装置相比, 看起来会更加不透明且更明亮。减小本设置会使火焰变得更透明, 并使用更多的外部颜色 (Outer color)。加大本设置会使火焰变得不透明且更明亮, 并会逐渐将内部颜色 (Inner color) 用炽热的白色代替。本设置值越大, 火焰中部越白。可动画
<Fire_Effect>.Samples	Integer	15	设置火焰效果的采样速率。本设置值越大, 获得的火焰效果越逼真, 但渲染时间也会更长。可动画
<Fire_Effect>.phase	Float	0.0	控制火焰变化的速度。可动画
<Fire_Effect>.Drift	Float	0.0	设置火焰效果在渲染时沿火焰装置的 Z 轴升起的快慢。可动画

(续表)

属性名称	数据类型	默认值	说明
<Fire_Effect>.Explosion	Integer	0	如果为 1, 系统会根据属性.phase 的值自动为尺寸参数、密度参数、颜色参数设置动画。 0: Off; 1: On
<Fire_Effect>.Smoke	Integer	1	设置爆炸时是否产生烟: 0: Off; 1: On
<Fire_Effect>.Fury	Float	1.0	指定属性.phase 变化的剧烈程度。其大于 1.0 会产生剧烈燃烧的效果, 小于 1.0 会产生缓慢燃烧的效果

示例:

在 Explosion 对话框里的 Start 和 End 两个微调器可用来为爆炸设置关键帧的时间及值。用户也可以用脚本来实现这一点。在下面的脚本里爆炸开始和结束的时间分别为 20 和 75:

```
c = getAtmospheric 1
--建立一个新的.phase.
pc = bezier_float()
c.phase.controller = pc
--在 20f 处添加关键帧 1 并为其设置属性
k = addNewKey pc 20
k.value = 0
k.inTangentType = #slow
k.outTangentType = #fast
--在 75f 处添加关键帧 2 并为其设置属性
k = addNewKey pc 75
k.value = 300
k.inTangentType = #slow
k.outTangentType = #fast
--如果指定时间处已有关键帧, 改变时间
c.phase.keys[1].time = 20
c.phase.keys[2].time = 75
```

## 13.4 Fog: Atmospheric (雾环境效果)

构造函数

fog...

属性

属性名称	数据类型	默认值	说明
<Fog>.Fog_Color	Color	默认值: (color 255 255 255)	
		指定雾的色彩。可动画	
<Fog>.Use_Color_Map	Integer	0	指定是否使用环境色彩贴图: 0: Off; 1: On
<Fog>.Use_Opacity_Map	Integer	0	指定是否使用环境不透明贴图: 0: Off; 1: On

(续表)

属性名称	数据类型	默认值	说明
<Fog>.Fog_Background	Integer	1	指定是否使用雾化背景: 0: Off; 1: On
<Fog>.Fog_Type	Integer	0	设置雾的类型: 0: 标准雾; 1: 层雾
<Fog>.Exponential	Integer	0	如果本设置为 1, 依据与摄像机之间的距离, 以指数方式增加雾的密度; 如果为 0, 以线性方式增加雾的密度
<Fog>.Near	Float	0.0	设置摄像机近距范围雾百分比
<Fog>.Far	Float	100.0	设置摄像机远距范围内雾密度
<Fog>.Top	Float	100.0	设置层雾的上限范围
<Fog>.Bottom	Float	0.0	设置层雾的下限范围
<Fog>.Density	Float	50.0	设置层雾的密度
<Fog>.falloff	Integer	2	设置层雾衰减效果: 0: 自上顶部开始衰减 1: 自底部开始衰减 2: 不衰减
<Fog>.Horizon_Noise	Integer	0	设置是否为层雾水平方向的密度加入噪波: 0: Off; 1: On
<Fog>.size	Float	20.0	设置水平噪波的尺寸
<Fog>.angle	Float	5.0	设置层雾相对于地平线的角度
<Fog>.phase	Float	0.0	设置层雾的相位, 相位大于 0 可以创建升腾的层雾效果, 相位小于 0 可以创建沉积的层雾效果

**注意** 目前尚不能在 MAXScript 里将纹理贴图来作为色彩和不透明度的贴图, 如果以后实现了这一点, 它们可以作为雾效果的 SubAnims 来进行存取。

## 13.5 Volume\_Fog: Atmospheric (体积雾环境效果)

### 构造函数

Volume\_fog...

### 属性

属性名称	数据类型	默认值	说明
<Volume_Fog>.Softens_Gizmo_Edges	Float	0.2	取值范围在 0~1.0, 对体积雾的边界进行羽化处理, 其值越高效果边界越柔和。可动画
<Volume_Fog>.Fog_Color	Color	默认值: (color 255 255 255)	设置体积雾的颜色。可动画

(续表)

属性名称	数据类型	默认值	说明
<Volume_Fog>.Exponential	Integer	0	如本属性为1, 将依据与摄像机时间的相对距离以指数方式增加体积雾的密度; 如本属性为0, 以线性方式增加体积雾的密度
<Volume_Fog>.Density	Float	20.0	设置体积雾的密度, 取值范围为0~20.0。可动画
<Volume_Fog>.Step_Size	Float	4.0	指定雾采样的步幅。数值越小, 体积雾越细腻; 数值越大, 体积雾越粗糙
<Volume_Fog>.Max_Steps	Integer	100	设置采样计算的最大步幅, 避免采样计算无休止地进行
<Volume_Fog>.Fog_Background	Integer	1	指定是否将雾功能应用于场景的背景
<Volume_Fog>.Noise_Type	Integer	0	指定噪波效果类型: 0: 标准; 1: 分形; 2: 湍流
<Volume_Fog>.invert	Integer	0	指定是否反转体积雾的噪波效果: 0: Off; 1: On
<Volume_Fog>.High_Threshold	Float	1.0	设置噪波的最高阈限。可动画
<Volume_Fog>.Low_Threshold	Float	0.0	设置噪波的最底阈限。可动画
<Volume_Fog>.Uniformity	Float	0.0	设置效果的均匀度, 数值越小, 雾的颗粒越细, 体积雾的效果越透明。可动画
<Volume_Fog>.Levels	Float	3.0	设置噪波重复迭代的次数。取值范围在1~6之间, 本设置仅对分形和湍流噪波有效。可动画
<Volume_Fog>.size	Float	20.0	设置体积雾噪波的大小, 较高的设置可加大体积雾的卷曲碎片。可动画
<Volume_Fog>.phase	Float	0.0	设置体积雾动态噪波的相位。可动画
<Volume_Fog>.Wind_Strength	Float	0.0	设置体积雾在风力吹动下的运动速度
<Volume_Fog>.Wind_Direction	Integer	0	指定风力的方向: 0: 前方; 1: 后方; 2: 左方; 3: 右方; 4: 顶部; 5: 底部

## 13.6 Volume\_Light: Atmospheric

### 构造函数

volume\_light...

### 属性

属性名称	数据类型	默认值	说明
<Volume_Light>.Fog_Color	Color	默认值: (color 255 255 255)	
		指定体积光效果的雾色彩。可动画	
<Volume_Light>.attenuation_color	Color	(color 0 0 255)	设置体积雾的衰减色彩。可动画
<Volume_Light>.Exponential	Integer	0	设置体积光的衰减色彩
<Volume_Light>.Density	Float	5.0	体积光的密度。可动画
<Volume_Light>.Max_Light	Float	90.0	设置体积积光可以达到的最大发光亮度。可动画
<Volume_Light>.Min_Light	Float	0.0	设置体积积光可以达到的最小发光亮度。可动画
<Volume_Light>.Use_Attenuation_Color	Integer	0	设置是否使用衰减色彩: 0: Off; 1: On
<Volume_Light>.Attenuation_Color_Multiplier	Float	2.0	设置调整衰减色彩的倍增系数。可动画
<Volume_Light>.Filter_Shadows	Integer	3	设置过滤阴影的方式: 0: 低, 不过滤 1: 中, 对相邻像素进行采样与均衡处理 2: 高, 对相邻像素和对象像素进行采样与均衡处理 3: 使用灯光采样范围
<Volume_Light>.Samples	Integer	20	体积光的采样体积速率
<Volume_Light>.Auto_Sample	Integer	1	是否使用自动采样体积百分比: 0: Off; 1: On
<Volume_Light>.Atten_Start	Float	100.0	设置体积光开始衰减的位置, 与灯光对象的衰减参数相对应。可动画
<Volume_Light>.Atten_End	Float	100.0	设置体积光结束衰减的位置, 与灯光对象的衰减参数相对应。可动画
<Volume_Light>.Noise_On	Integer	0	是否激活噪波效果: 0: Off; 1: On
<Volume_Light>.Noise_Amount	Float	0.0	指定体积光噪波效果的强度, 取值范围0~1, 如果为0, 表示取消噪波效果。可动画
<Volume_Light>.Link_To_Light	Integer	0	是否将噪波效果链接到灯光对象: 0: Off; 1: On
<Volume_Light>.Noise_Type	Integer	0	指定噪波效果类型: 0: 标准; 1: 分形; 2: 湍流
<Volume_Light>.invert	Integer	0	指定是否反转体积光的噪波效果: 0: Off, 1: On

(续表)

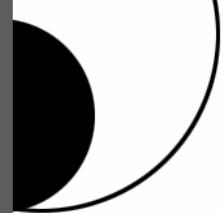
属性名称	数据类型	默认值	说明
<Volume_Light>.High_Threshold	Float	1.0	设置噪波的最高阈限。可动画
<Volume_Light>.Low_Threshold	Float	0.0	设置噪波的最低阈限。可动画
<Volume_Light>.Uniformity	Float	0.0	设置效果的均匀度, 数值越小, 雾的颗粒越细, 体积雾的效果越透明。可动画
<Volume_Light>.Levels	Float	3.0	设置噪波重复迭代的次数。取值范围在1~6之间, 本设置仅对分形和湍流噪波有效。可动画
<Volume_Light>.size	Float	20.0	设置体积光噪波的大小, 较高的设置可加大体积雾的卷曲碎片。可动画
<Volume_Light>.phase	Float	0.0	设置体积光动态噪波的相位。可动画
<Volume_Light>.Wind_Strength	Float	0.0	设置体积光在风力吹动下的运动速度。可动画
<Volume_Light>.Wind_Direction	Integer	0	指定风力的方向: 0: 前方; 1: 后方; 2: 左方; 3: 右方; 4: 顶部; 5: 底部

## 13.7 使用 Atmospheric 的示例

下面的示例演示了如何创建体积雾和体积光效果:

```
-- 创建一个 geosphere、sphere gizmo、omnilight 和 targeted camera
geos=geosphere radius:24
sgizmo=spheregizmo radius:40
omni=omnilight farAttenStart:20 farAttenEnd:100 \
useFarAtten:True v=255
cam=targetcamera pos:[150,-50,100] target:(targetobject())
-- 为 geosphere 对象指定材质
geos.material=standard()
geos.material.selfIllumColor = color 238 241 2
-- 创建一个体积雾, 为它指定名称, 这样在 Environment 对话框里可以看到
vf=volume_fog name: "VolFog" soften_gizmo_edges:0.7 \
exponential:1 density:75
vf.fog_color=color 255 157 0
vf.noise_type=2
vf.size=4
appendGizmo vf sgizmo
addAtmospheric vf
-- 创建一个体积光, 为它指定名称, 这样在 Environment 对话框里可以看到
vl=volume_light name: "VolLight" exponential:1 density:20
vl.fog_color=color 255 30 0
vl.attenuation_color= color 255 162 0
```

```
vl.Use_Attenuation_Color=1  
vl.noise_on=1  
vl.Noise_Amount=1  
vl.size=5  
vl.noise_type=2  
appendGizmo vl omni  
addAtmospheric vl  
renderWidth=320  
renderHeight=200  
render camera:cam
```



第  
3  
部  
分

用 MAXScript  
创建实用工具、  
用户界面

# 第 14 章 RenderEffect (渲染效果)

RenderEffect 类可以用来在 MAXScript 里创建、设置渲染效果，如 Blur、Color Balance 和 Film Grain 等。

RenderEffect 类是从 MAXWrapper 类派生而来，并继承了 MAXWrapper 类的属性和方法。

## 14.1 渲染效果通用属性和方法

Render Effects 类有下面的通用属性和方法。

### 属性

`<renderEffect>.name: String, 默认值: 不定`

指定对象显示在 Rendering | Effects 对话框里的名字。当创建一个 renderEffect 对象时，如果没有指 name 参数，系统使用类名作为默认的对象名。

### 方法

1. `addEffect <renderEffect>`

向 Rendering | Effects 对话框的清单里添加一个 RenderEffect。

2. `setActive <renderEffect> <boolean>`

将指定 Render Effect 设置为活动的或不活动的。

用户可以使用下面的全局变量和函数来管理 Rendering | Effects 对话框里的 Render Effects 清单。

3. `IsActive <renderEffect>`

如果 Render Effect 当前状态为活动的，返回 True；否则，返回 False。

4. `numEffects`

获取 Rendering | Effects 对话框清单里指定 Render Effect 的序号，只读的全局变量。

5. `getEffect <index_integer>`

获取 Rendering | Effects 对话框清单里指定序号的 Render Effect，序号从 1 开始。

6. `editEffect <renderEffect> [ gizmo: <node> ]`

打开 Rendering | Effects 对话框，并打开指定 renderEffect 的卷展栏（如果它已被添加到对话框的清单里），如果指定了可选参数 gizmo:，而且该指定 Node 对象是 RenderEffect 的一个 gizmo，该 gizmo 会被选中。

7. `setEffect <index_integer> <renderEffect>`

将 Rendering | Effects 对话框的 Render Effect 清单指定序号位置设为指定 Render Effect，如果调用本方法时 Rendering | Effects 对话框正被打开，清单不会被刷新。

#### 8. deleteEffect <index\_integer>

删除 Rendering | Effects 对话框清单里指定位置的 RenderEffect。

**注意** 和绝大多数 MAXWrapper 对象不同，RenderEffect 类对象不能被复制。

## 14.2 渲染效果类型

下面是 3ds max 里所有 RenderEffect 类对象：

- ◆ Blur Effect (模糊渲染效果)
- ◆ Brightness\_and\_Contrast (亮度和对比度渲染效果)
- ◆ Color\_Balance Effect (颜色平衡渲染效果)
- ◆ Depth\_of\_Field (景深渲染效果)
- ◆ File\_Output Effect (文件输出渲染效果)
- ◆ Film Grain (胶片颗粒渲染效果)
- ◆ Lens\_Effects (镜头渲染效果)
- ◆ Motion\_Blur (运动模糊渲染效果)

## 14.3 Blur:RenderEffect (模糊渲染效果)

### 构造函数

blur...

### 属性

属性名称	数据类型	默认值	说明
<blur>.blur_type	Integer	0	设置模糊类型： 0: Uniform (均匀型) 1: Directional (方向型) 2: Radial (径向型) 可动画
<blur>.bUnifPixRad	Float	10.0	指定模糊效果强度，适用于 Uniform 类型的模糊。 增大本设置使模糊效果更强烈
<blur>.bUnifAlpha	Boolean	True	如果设为 True，表示将 Uniform 类型的模糊效果施加到 Alpha 通道上。可动画

(续表)

属性名称	数据类型	默认值	说明
<blur>.bDirUPixRad	Float	10.0	指定像素水平方向的模糊效果强度,适用于Directional类型的模糊效果,增大本设置使水平方向的模糊效果更强烈
<blur>.bDirVPixRad	Float	10.0	指定像素垂直方向的模糊效果强度,适用于Directional类型的模糊效果,增大本设置使垂直方向的模糊效果更强烈
<blur>.bDirUTrail	Float	0.0	在U轴的两边加重模糊效果,用来模拟对象或摄像机在水平方向速度移动的效果
<blur>.bDirVTrail	Float	0.0	在V轴的两边加重模糊效果,用来模拟对象或摄像机在垂直方向速度移动的效果
<blur>.bDirRotation	Integer	0	旋转U轴和V轴像素的方向,利用本设置可以在任意方向产生模糊效果。如果本设置为0,U方向对应图像的X方向,而V方向对应图像的Y轴
<blur>.bDirAlpha	Boolean	True	如果设为True,将模糊效果指定到Alpha通道上。可动画
<blur>.bRadialPixRad	Float	20.0	同于指定Radial类型模型效果的强度,增大本设置会得到更强烈的模糊效果
<blur>.bRadialXOrig	Integer	320	设置模糊效果中心的X坐标
<blur>.bRadialYOrig	Integer	240	设置模糊效果中心的Y坐标
axis.<blur>.bRadialTrail	Float	0.0	设置由模糊中心向图像边缘增加方向性行迹。用来产生对象或摄像机朝某一方向快慢移动的效果
<blur>.bRadialAlpha	Boolean	True	当设置为True时,将Radial类型的模糊效果指定到Alpha通道上。可动画
<blur>.bRadialNode	Node	undefined	Radial类型模糊效果使用本对象集的中心作为模糊中心
<blur>.bRadialUseNode	Boolean	False	设置是否使用.bRadialNode属性指定的对象集的中心作为模糊中心
<blur>.selImageActive	Boolean	True	当设置为True时,模糊效果会影响整个渲染图像。如果模糊效果使渲染效果图像看起来非常暗淡,本设置非常有用。可动画
<blur>.selImageBrighten	Float	0.0	增加整个图像的亮度
<blur>.selImageBlend	Float	100.0	将模糊效果和整个图像的初始渲染图像进行混合,这样可以用来创建柔和聚焦的效果
<blur>.selIBackActive	Boolean	False	当设为True时,模糊效果不对场景的背景图像或背景动画进行处理
<blur>.selIBackBrighten	Float	0.0	增加图像的亮度,但不对背景图像或背景动画进行亮化处理
<blur>.selIBackBlend	Float	100.0	将渲染效果和初始渲染图像的非背景参数混合
<blur>.selIBackFRadius	Float	10.0	虚化模糊效果,但仅对场景里的非背景对象进行处理。如果将.selIBackActive属性设为True,用户将会看到场景对象与其模糊效果之间有一个明显的边界,这是因为对象被经过模糊处理,而背景却没有

(续表)

属性名称	数据类型	默认值	说明
<blur>.selLumActive	Boolean	False	模糊效果只对那些明度值在指定明度范围的像素进行处理
<blur>.selLumBrighten	Float	0.0	对明度值在指定范围内的像素进行亮化处理
<blur>.selLumBlend	Float	100.0	将指定明度范围内的模糊效果和图像的初始效果进行混合
<blur>.selLumMin	Float	90.0	指定明度范围的最小值。每个像素的明度必须大于这个值才会进行模糊效果处理
<blur>.selLumMax	Float	100.0	指定明度范围的最大值。每个像素的明度必须小于这个值才会进行模糊效果处理
<blur>.selLumFRadius	Float	10.0	设置羽化半径，对那些明度值在指定范围内的像素的模糊效果进行羽化处理。如果在 Blur Parameters 里勾选 Luminance 选项，模糊效果会有一条明显的边界，利用本设置可对边界进行柔化处理
<blur>.selMaskActive	Boolean	False	如果设为 True，将根据在 Material   Map Browser 对话框里设定的通道和 mask 的情况进行模糊处理
<blur>.selMaskMap	TextureMap	undefined	指定模糊效果的贴图
<blur>.selMaskChannel <blur>.select_mask_channel	Integer	4	指定模糊效果应用到哪一通道： 0: Red 1: Blue 2: Green 3: Alpha 4: Luminance
<blur>.selMaskBrighten	Float	0.0	对图像中被进行模糊处理的部分像素进行加亮处理
<blur>.selMaskBlend	Float	100.0	将 Map Mask 模糊效果与初始渲染图像进行混合
<blur>.selMaskMin	Float	90.0	指定像素的色彩 (RGB、Alpha、明度) 范围的最小值，如果小于该值，将不进行模糊处理
<blur>.selMaskMax	Float	100.0	指定像素的色彩 (RGB、Alpha、明度) 范围的最大值，如果大于该值，将不进行模糊处理
<blur>.selMaskFRadius	Float	10.0	设置羽化半径，对通道值在指定范围内的像素进行羽化处理。如果在 Blur Parameter 里勾选 Map Mask，会在像素边缘产生一条明显的边界，利用本参数可对边界进行柔化处理
<blur>.selObjIdsActive	Boolean	False	如果设为 True，会将模糊效果应用于对象或对象中具有特定对象 ID 的部分。可动画
<blur>.selObjIdsIds	Array	#()	包含所有要进行模糊处理的对象的对象 ID 号码的数组
<blur>.selObjIdsBrighthen	Float	0.0	加亮图像中应用模糊效果的部分。可动画
<blur>.selObjIdsBlend	Float	100.0	将对象 ID 模糊效果与原始的渲染图像混合。可动画

(续表)

属性名称	数据类型	默认值	说明
<blur>.selObjIdsFRadius	Float	10.0	设置羽化半径, 对明度值在指定范围内的像素进行羽化处理。可动画
<blur>.selObjIdsLumMin	Float	0.0	指定明度范围的最小值。每个像素的明度必须大于这个值才会进行模糊效果处理。可动画
<blur>.selObjIdsLumMax	Float	100.0	指定明度范围的最大值。每个像素的明度必须小于这个值才会进行模糊效果处理。可动画
<blur>.selMatIdsActive	Boolean	False	是否会将模糊效果应用于材质或材质中具有特定材质效果通道的部分。可动画
<blur>.selMatIdsIds	Array	#()	包含要进行模糊处理的材质 ID 号的数组
<blur>.selMatIdsBrighten	Float	0.0	对要进行模糊处理的像素进行加亮处理。可动画
<blur>.selMatIdsBlend	Float	100.0	将材质 ID 模糊效果和初始渲染图像进行混合。可动画
<blur>.selMatIdsFRadius	Float	10.0	设置羽化半径, 对明度值在指定范围内的像素进行羽化处理。可动画
<blur>.selMatIdsLumMin	Float	0.0	指定明度范围的最小值。每个像素的明度必须大于这个值才会进行模糊效果处理。可动画
<blur>.selMatIdsLumMax	Float	100.0	指定明度范围的最大值。每个像素的明度必须小于这个值才会进行模糊效果处理。可动画
<blur>.selGenBrightType	Integer	1	选择羽化衰减的控制曲线类型: 0: Additive; 1: Multiplicative 可动画
<blur>.select_falloff_curve	SubAnim	SubAnim:Select_Falloff_Curve	
<blur.select_falloff_curve>.curve_1	SubAnim	默认值: SubAnim:Curve_1 说明: 羽化衰减的亮化曲线	
<blur.select_falloff_curve>.curve_2	SubAnim	默认值: SubAnim:Curve_2 说明: 羽化衰减的混合曲线	

## 14.4 Brightness\_and\_Contrast: RenderEffect (亮度和对比度渲染效果)

### 构造函数

brightness\_and\_contrast...

### 属性

属性名称	数据类型	默认值	说明
<Brightness_and_Contrast>.Brightness	Float	0.5	指定压缩或扩展纯黑与纯白之间的亮度范围, 取值范围为 0~1.0。可动画
<Brightness_and_Contrast>.contrast	Float	0.5	设置对比度, 增加或减少输出图像的对比度, 取值范围为 0~1.0。可动画
<Brightness_and_Contrast>.ignoreBack <Brightness_and_Contrast>.ignore_background	Boolean	False	如果设为 True, 只对场景中对象进行亮度或对比度处理, 而不对背景图像或背景动画进行处理。可动画

## 14.5 Color\_Balance: RenderEffect (颜色平衡渲染效果)

### 构造函数

color\_balance...

### 属性

属性名称	数据类型	默认值	说明
<Color_Balance>.red	Integer	0	调整红色通道。可动画
<Color_Balance>.green	Integer	0	调整绿色通道。可动画
<Color_Balance>.blue	Integer	0	调整蓝色通道。可动画
<Color_Balance>.preserveLum <Color_Balance>.Preserve_Luminosity	Boolean	False	如果设为 True, 只改变图像的色相基调, 不改变图像的明度。可动画
<Color_Balance>.ignoreBack <Color_Balance>.Ignore_Background	Boolean	False	如果设为 True, 只对场景中对象进行亮度或对比度处理, 而不对背景图像或背景动画进行处理。可动画

## 14.6 Depth\_of\_Field: RenderEffect (景深渲染效果)

### 构造函数

depth\_of\_field...

### 属性

属性名称	数据类型	默认值	说明
<Depth_of_Field>.AffectAlpha <Depth_of_Field>.Affect_Alpha	Boolean	True	如果设为 True，景深渲染效果将作用于渲染输出图像的 Alpha 通道
<Depth_of_Field>.CamNodeIndex <Depth_of_Field>.Camera_Index	Integer	-1	指定景深渲染效果使用 Cameras 下拉列表中的哪一个摄像机。作为被选摄像机如果指定为 -1，表示未指定摄像机，摄像机序号从 0 开始
<Depth_of_Field>.focalNodeIndex <Depth_of_Field>.Focal_Index	Integer	-1	指定景深渲染效果使用 Focal Point 下拉列表里的哪一个摄像机被选择作为焦点对象。如果指定为 -1，表示未选择焦点对象，序号从 0 开始
<Depth_of_Field>.FocalPoint <Depth_of_Field>.Focal_Point	Integer	0	设置焦点类型： 0：使用焦点对象；1：使用摄像机
<Depth_of_Field>.FocalType <Depth_of_Field>.Focal_Type	Integer	0	0：Custom（自定义） 1：Use Camera（使用摄像机）
<Depth_of_Field>.HorizFocalLoss <Depth_of_Field>.Horiz_Focal_Loss	Float	10.0	水平焦点损失，指定沿图像水平轴向的虚化程度。可动画
<Depth_of_Field>.VertFocalLoss <Depth_of_Field>.Vert_Focal_Loss	Float	10.0	垂直焦点损失，指定沿图像垂直轴向的虚化程度。可动画
<Depth_of_Field>.FocalRange <Depth_of_Field>.Focal_Range	Float	100.0	焦点范围，指定场景 Z 轴向的一段距离，在焦点前后的指定距离范围内，保持清晰对焦的效果。可动画
<Depth_of_Field>.FocalLimit <Depth_of_Field>.Focal_Limit	Float	200.0	焦点限制，指定场景 Z 轴向的一段距离，在焦点前后的指定距离范围内，虚化达到最大的效果。可动画

## 方法

### 1. DOF.addCam <Depth\_of\_Field> <camera\_node\_name\_string>

将指定摄像机加入景深渲染效果的 Camera 下拉清单。参数<camera\_node\_name\_string>指定了摄像机的名称，其大小写必须与摄像机名称匹配。如果添加成功，返回 True；否则返回 False。通常操作失败的原因是指定摄像机在场景中不存在。

### 2. DOF.addFocalNode <Depth\_of\_Field> <node\_name\_string>

将指定对象添加到景深渲染效果的 Focal Point 下拉清单。参数<node\_name\_string>指定了添加对象的名称，其大小写必须与对象名称匹配。如果添加成功，返回 True；否则返回

`False`。通常操作失败的原因是指定对象在场景中不存在。

### 3. `DOF.deleteCam <Depth_of_Field> <index_integer>`

从景深渲染效果的 Camera 下拉清单中删去指定序号的摄像机。参数`<index_integer>`为摄像机在下拉清单中的序号（序号从 0 开始，以与属性 `camNodeIndex` 保持一致）。如果删除成功，返回 `True`；否则返回 `False`。一般操作失败的原因是指定序号的摄像机不在 Camera 下拉清单中。

### 4. `DOF.deleteCamByName <Depth_of_Field> <camera_node_name_string>`

从景深渲染效果的 Camera 下拉清单里删去指定名称的摄像机。参数`<camera_node_name_string>`指定了摄像机的名称，其大小写必须与摄像机名称匹配。如果删除成功，返回 `True`；否则返回 `False`。通常操作失败的原因是指定摄像机在场景中不存在。

### 5. `DOF.deleteFocalNode <Depth_of_Field> <index_integer>`

从景深渲染效果的 Focal Point 下拉清单里删除指定序号的焦点对象。参数`<index_integer>`为摄像机在下拉清单中的序号（序号从 0 开始，以与属性 `.camNodeIndex` 保持一致）。如果删除成功，返回 `True`；否则，返回 `False`。一般操作失败的原因是指定序号的摄像机不在 Camera 下拉清单中。

### 6. `DOF.deleteFocalNodeByName <Depth_of_Field> <node_name_string>`

从景深渲染效果的 Focal Point 下拉清单里删除指定名称的焦点对象。参数`<node_name_string>`指定了删除对象的名称，其大小写必须与对象名称匹配。如果删除成功，返回 `True`；否则返回 `False`。通常操作失败的原因是指定对象在下拉清单中不存在。

## 14.7 File\_Output: RenderEffect (文件输出渲染效果)

### 构造函数

`file_output...`

### 属性

属性名称	数据类型	默认值	说明
<code>&lt;File_Output&gt;.bitmap</code>	BitMap		打开一个对话框，用户可以将渲染的图像或动画保存到磁盘上
<code>&lt;File_Output&gt;.channelType</code> <code>&lt;File_Output&gt;.channel_type</code>	Integer	0	指定输出通道类型： 0: 整个图像 1: 明度 2: 景深 3: Alpha 通道
<code>&lt;File_Output&gt;.active</code>	Boolean	<code>True</code>	设置是否启用文件输出功能。可动画
<code>&lt;File_Output&gt;.affectSource</code> <code>&lt;File_Output&gt;.affect_source</code>	Boolean	<code>False</code>	当设置为 <code>On</code> 时，将接收以前应用了效果的图像，将其转换为所选的通道，再发送回堆栈，以便应用其他效果。渲染图像将保存在所选的通道中。如果属性 <code>ChannelType</code> 为 0，本设置不起作用。可动画

(续表)

属性名称	数据类型	默认值	说明
<File_Output>.cameraNode <File_Output>.camera_node	Node	undefined	指定场景中的摄像机
<File_Output>.nearZ <File_Output>.near_z_depth	Float	0.0	指定在 Z 轴方向距离摄像机最近的距离。可动画
<File_Output>.farZ <File_Output>.far_z_depth	Float	-500.0	指定在 Z 轴方向距离摄像机最远的距离。可动画
<File_Output>.fitScene <File_Output>.fit_entire_scene	Boolean	True	当设置为 On 时, 使所有其他深度参数均不可用, 并且将渲染深度通道图像文件中整个视口的场景几何体, 自动计算所需的近端 Z 轴方向距离和远端 Z 轴方向距离。可动画

## 14.8 Film\_Grain: RenderEffect (胶片颗粒渲染效果)

### 构造函数

film\_grain...

### 属性

1. <Film\_Grain>.Grain : Float, 默认值: 0.2, 可动画

指定加入到渲染输出图像中的胶片颗粒数量, 取值范围在 0~1.0 之间。

2. <Film\_Grain>.‘Mask Background’ Boolean, 默认值: False, 可动画

别名: Ignore\_Background。

如果设为 True, 只对场景中所有对象进行胶片颗粒处理, 不影响场景的背景图像或背景动画。

## 14.9 Lens\_Effects: RenderEffect (镜头渲染效果)

### 构造函数

lens\_effects...

### 属性

属性名称	数据类型	默认值	说明
<Lens_Effects>.size	Float	100.0	指定全部镜头效果的通用尺寸, 其单位为当前渲染尺寸的百分比。可动画
<Lens_Effects>.intensity	Float	100.0	指定全部镜头效果的通用亮度和不透明度, 该数值越大, 镜头效果越明亮, 也越不透明。可动画
<Lens_Effects>.seed	Integer	1357	为镜头效果的随机数指定一个开始点。可动画

(续表)

属性名称	数据类型	默认值	说明
<Lens_Effects>.angle	Float	0.0	指定镜头效果基于当前默认位置的旋转量。可动画
<Lens_Effects>.squeeze	Float	0.0	设置挤压镜头效果水平方向或垂直方向的尺寸。取值范围为-100~100, 正值表示拉伸镜头效果的水平轴向, 负值表示拉伸镜头效果的垂直轴向。可动画
<Lens_Effects>.affectAlpha <Lens_Effects>.Affect_Alpha	Boolean	True	设置镜头效果是否影响 32 位图像的 Alpha 通道
<Lens_Effects>.affectZBuffer <Lens_Effects>.Affect_Z_Buffer	Boolean	False	设置镜头效果是否影响图像的 Z 缓冲
<Lens_Effects>.distAffectsSize <Lens_Effects>.Distanc e_Affects_Size	Boolean	False	设置从摄像机到效果灯光之间的距离是否影响镜头效果的尺寸
<Lens_Effects>.distAffectsIntensity <Lens_Effects>.Distance_Affects_Intensity	Boolean	False	设置从摄像机到效果灯光之间的距离是否影响镜头效果的强度
<Lens_Effects>.centerAffectsSize <Lens_Effects>.Off_enter_Affects_Size	Boolean	False	指定从摄像机到效果灯光之间的中心偏移是否影响镜头效果的尺寸
<Lens_Effects>.centerAffectsIntensity <Lens_Effects>.Off_enter_Affects_Intensity	Boolean	False	指定从摄像机到效果灯光之间的中心偏移是否影响镜头效果的强度
<Lens_Effects>.innerOcclusionRadius <Lens_Effects>.Inner_Occlusion_Radius	Float	3.0	设置镜头效果内部阻光半径, 该范围内阻光对象完全被镜头效果遮挡。可动画
<Lens_Effects>.outerOcclusionRadius <Lens_Effects>.Outer_Occlusion_Radius	Float	33.0	设置镜头效果的外部阻光半径, 在内部阻光半径到外部阻光半径之间的范围, 镜头效果会逐渐淡出消失。可动画
<Lens_Effects>.occlusionAffectsSize <Lens_Effects>.Occlusion_Affects_Size	Boolean	True	如果设为 True, 阻光对象会移影响镜头效果的尺寸
<Lens_Effects>.occlusionAffectsIntensity <Lens_Effects>.Occlusion_Affects_Intensity	Boolean	True	如果设为 True 阻光对象会移影响镜头效果的强度
<Lens_Effects>.affectByAtmosphere <Lens_Effects>.Affect_By_Atmosphere	Boolean	False	指定场景中的大气效果是否会影响镜光效果

## 方法

下面所列方法可用装载/存储一个镜头效果配置文件，添加/删除灯光和添加/删除镜头效果元素。

1. le.addASec <Lens\_Effects>

为指定镜头效果添加一个自动二级光斑镜头效果，如果添加成功，返回 True；否则，返回 False。

2. le.addGlow <Lens\_Effects>

为指定镜头效果添加一个光晕效果，如果添加成功，返回 True；否则，返回 False。

3. le.addMSec <Lens\_Effects>

为指定镜头效果添加一个手动二级光斑镜头效果，如果添加成功，返回 True；否则，返回 False。

4. le.addRay <Lens\_Effects>

为指定镜头效果添加一个射线镜头效果，如果添加成功，返回 True；否则，返回 False。

5. le.addRing <Lens\_Effects>

为指定镜头效果添加一个光环镜头效果，如果添加成功，返回 True；否则，返回 False。

6. le.addStar <Lens\_Effects>

为指定镜头效果添加一个星形镜头效果，如果添加成功，返回 True；否则，返回 False。

7. le.addStreak <Lens\_Effects>

为指定镜头效果添加一个条纹镜头效果，如果添加成功，返回 True；否则，返回 False。

8. le.load <Lens\_Effects> <filename\_string>

导入指定镜头效果配置文件到当前场景中。

9. le.save <Lens\_Effects> <filename\_string>

将镜头效果的参数设置存储到指定文件中，属性的动画部分不会被存储。

10. le.numLights <Lens\_Effects>

返回 Light 下拉列表中的灯光数目。

11. le.addNode <Lens\_Effects> <light\_node>

将指定灯光对象加入到 Light 下拉列表中。

12. le.addLight <Lens\_Effects> <light\_node\_name\_string>

将指定名称的灯光加入到 Light 下拉列表中，参数<light\_node\_name\_string>为灯光对象的名称，其大小写必须与灯光对象名称一致。如果添加成功，返回 True；否则返回 False。一般而言，操作失败的原因是指定的灯光不在场景里。

13. le.deleteLight <Lens\_Effects> <index\_integer>

将指定序号的灯光从 Light 下拉列表中删除。如果添加成功，返回 True；否则返回 False。

一般而言，操作失败的原因是指定的灯光不在场景里。

14. le.deleteLightByName <light\_node\_name\_string>

将指定名称的灯光从 Light 下拉列表中删除，参数<light\_node\_name\_string>为灯光对象的名称，其大小写必须与灯光对象名称一致。如果添加成功，返回 True；否则返回 False。一般而言，操作失败的原因是指定的灯光不在 Light 下拉列表里。

15. le.lightIndex <Lens\_Effects><light\_node\_name\_string>

返回指定名称的灯光在 Light 下拉列表中的序号，如果指定灯光不在 Lights 下拉列表中，返回 False。

16. le.lightName <Lens\_Effects><index>

返回指定索引的灯光的名称，如果序号超出了 Lights 下拉列表的范围，返回一个空字符串。

17. le.deleteElement <Lens\_Effects><index\_integer>

删取指定序号的效果，如果删除成功，返回 True；否则，返回 False。

18. le.deleteElementByName <element\_name\_string>

删去指定名称的效果。变量<element\_name\_string>的大小写必须与效果名称一致。如果删除成功，返回 True；否则，返回 False。

#### 14.9.1 Lens\_Effects-Auto\_Secondary（自动二级光斑镜头效果）

##### 构造函数

le.addASec <Lens\_Effects>

##### 属性

属性名称	数据类型	默认值	说明
<auto_secondary>.elementName <auto_secondary>.name	String	“Auto Secondary”	指定自动二级光斑镜头效果的名称
<auto_secondary>.on	Boolean	True	设置是否将效果应用于渲染图像
<auto_secondary>.minSize <auto_secondary>.Min_Size	Float	25.0	设置最小光斑尺寸，单位为当前渲染尺寸的百分数。可动画
<auto_secondary>.maxSize <auto_secondary>.Max_Size	Float	100.0	设置最大光斑尺寸，单位为当前渲染尺寸的百分数。可动画
<auto_secondary>.axis	Float	3.0	定义自动二级光斑沿其进行分布的轴的总长度。可动画
<auto_secondary>.intensity	Float	30.0	设置效果的亮度和不透明度。数值越大，效果越明亮，越不透明。可动画
<auto_secondary>.quantity	Integer	12	设置光斑组中子光斑的数量。可动画
<auto_secondary>.colorSource <auto_secondary>.Source_Color	Float	20.0	指定自动光斑镜头效果与源灯光色彩或源对象色彩的混合程度，如果为 0，仅使用源色彩；如果为 100，仅使用效果色彩。可动画
<auto_secondary>.sides	Integer	0	指定光斑组中的形态： 0：圆形；1：三角形；2：四边形； 3：五边形；4：六边形；5：七边形； 6：八边形

(续表)

属性名称	数据类型	默认值	说明
<auto_secondary>.occlusion	Float	100.0	确定镜头效果场景阻光度参数对特定效果的影响程度。可动画
<auto_secondary>.presets	Integer	0	指定选择 Presets 下拉列表里选择哪一项作为预设置(如 Rainbow), 如果为 0, 表示无预设置
<auto_secondary>.squeeze	Boolean	False	指定挤压设置是否影响二级光斑镜头效果
<auto_secondary>.radialColor1 <auto_secondary>.Radial_Color_1	Color	(color 0 0 0)	设置第一个影响光斑效果的放射色彩。可动画
<auto_secondary>.radialColor2 <auto_secondary>.Radial_Color_2	Color	(color 57 34 89)	设置第二个影响光斑效果的放射色彩。可动画
<auto_secondary>.radialColor3 <auto_secondary>.Radial_Color_3	Color	(color 85 139 85)	设置第三个影响光斑效果的放射色彩。可动画
<auto_secondary>.radialColor4 <auto_secondary>.Radial_Color_4	Color	(color 190 99 10)	设置第四个影响光斑效果的放射色彩。可动画
<auto_secondary>.colorRadius1 <auto_secondary>.Radius_Color_1	Float	90.0	设置第一个放射色彩的影响半径。可动画
<auto_secondary>.colorRadius2 <auto_secondary>.Radius_Color_2	Float	92.0	设置第二个放射色彩的影响半径。可动画
<auto_secondary>.colorRadius3 <auto_secondary>.Radius_Color_3	Float	95.0	设置第三个放射色彩的影响半径。可动画
<auto_secondary>.colorRadius4 <auto_secondary>.Radius_Color_4	Float	98.0	设置第四个放射色彩的影响半径。可动画
<auto_secondary>.radialMtl <auto_secondary>.Radial_Material	TextureMap	undefined	指定径向颜色的贴图
<auto_secondary>.useRadialMtl <auto_secondary>.Apply_Radial_Color_Map	Boolean	False	设置是否使用贴图确定径向颜色

(续表)

属性名称	数据类型	默认值	说明
<auto_secondary>.radial_falloff	SubAnim		设置放射的径向衰减值
<auto_secondary>.radialMap <auto_secondary>.Radial_Falloff_Map	TextureMap	undefined	指定径向衰减贴图
<auto_secondary>.useRadialMap <auto_secondary>.Apply_Radial_Falloff_Map	Boolean	False	指定是否使用径向衰减贴图
<auto_secondary>.circularColorMix <auto_secondary>.Circular_Color_Mix	Float	0.0	混合在径向颜色和环绕颜色中设置的颜色。如果设置为 0，将只使用径向颜色中设置的值；如果设置为 100，将只使用环绕颜色中设置的值。0~100 之间的任何值将在两个值之间混合。可动画
<auto_secondary>.quadrant1Color <auto_secondary>.Quadrant_1_Color	Color	(color 255 0 0)	指定用于混合径向颜色和环绕颜色的第一个颜色值。可动画
<auto_secondary>.quadrant2Color <auto_secondary>.Quadrant_2_Color	Color	(color 255 0 0)	指定用于混合径向颜色和环绕颜色的第二个颜色值。可动画
<auto_secondary>.quadrant3Color <auto_secondary>.Quadrant_3_Color	Color	(color 255 0 0)	指定用于混合径向颜色和环绕颜色的第三个颜色值。可动画
<auto_secondary>.quadrant4Color <auto_secondary>.Quadrant_4_Color	Color	(color 255 0 0)	指定用于混合径向颜色和环绕颜色的第四个颜色值。可动画
<auto_secondary>.circularMtl <auto_secondary>.Circular_Material	TextureMap	undefined	指定定义环绕色彩效果的材质。可动画
<auto_secondary>.useCircularMtl <auto_secondary>.Apply_Circular_Color_Map	Boolean	False	指定是否使用材质来定义环绕色彩的效果
<auto_secondary>.circular_falloff	SubAnim		定义环绕色彩的衰减方式
<auto_secondary>.circularMap <auto_secondary>.Circular_Falloff_Map	TextureMap	undefined	指定定义环绕色彩效果的贴图

(续表)

属性名称	数据类型	默认值	说明
<auto_secondary>.useCircularMap<auto_secondary>.Apply_Circular_Falloff_Map	Boolean	False	指定是否使用贴图来定义环绕色彩的效果
<auto_secondary>.radial_size	SubAnim		指定自动二级光斑镜头效果的放射尺寸
<auto_secondary>.radialSizeMap<auto_secondary>.Radial_Size_Map	TextureMap	undefined	指定是否使用贴图来定义放射型色彩的效果
<auto_secondary>.useRadialSizeMap<auto_secondary>.Apply_Radial_Sze_Map	Boolean	False	指定是否使用贴图来定义放射型色彩的效果
<auto_secondary>.applyLights<auto_secondary>.Apply_to_Lights	Boolean	True	是否将效果指定到源灯光对象上
<auto_secondary>.applyImage<auto_secondary>.Apply_to_Image	Boolean	False	是否将效果指定到渲染输出的图像上
<auto_secondary>.applyImageCenters<auto_secondary>.Apply_to_Image_Centers	Boolean	False	确定将效果指定到对象的中心还是对象的一部分
<auto_secondary>.sourceObjectID<auto_secondary>.Source_Object_ID	Boolean	False	设置是否将镜头效果应用于场景中具有相应 G 缓冲区 (或对象) ID 的特定对象。可动画
<auto_secondary>.objectID<auto_secondary>.Object_ID	Integer	1	设置对象通道 ID。可动画
<auto_secondary>.sourceEffectsID<auto_secondary>.Source_Effects_ID	Boolean	False	设置是否将效果指定给具有同一材质效果 ID 的对象。可动画
<auto_secondary>.effectsID<auto_secondary>.Effects_ID	Integer	0	设置材质效果 ID。可动画
<auto_secondary>.SourceUnclampedColor<auto_secondary>.Source_Unclamped_Color	Boolean	False	设置是否使用超亮度颜色
<auto_secondary>.unclampedColor<auto_secondary>.Unclamped_Color	Float	1.0	指定最低像素颜色, 像素颜色必须大于该值, 才会进行自动二级光斑镜头效果处理。可动画

(续表)

属性名称	数据类型	默认值	说明
<auto_secondary>.UnclampedColorInvert <auto_secondary>.Source_Unclamped_Color_Invert	Boolean	False	设置是否反转最低像素颜色
<auto_secondary>.sourceSurfaceNormal <auto_secondary>.Source_Surface_Normal	Boolean	False	设置是否使用表面法线来控制像素是否被进行效果处理
<auto_secondary>.surfaceNormal <auto_secondary>.Surface_Normal	Float	0.0	对象表面法线与摄像机之间的角度值，大于指定值时，该表面才会被进行自动二级光斑镜头效果处理。可动画
<auto_secondary>.surfaceNormalInvert <auto_secondary>.Source_Surface_Normal_Invert	Boolean	False	设置是否反转表面法线角度值
<auto_secondary>.sourceWhole <auto_secondary>.Source_Whole	Boolean	False	设置是否将自动二级光斑镜头效果指定到全部场景
<auto_secondary>.sourceAlpha <auto_secondary>.Source_Alpha	Boolean	False	设置是否将自动二级光斑镜头效果指定到某一 Alpha 通道
<auto_secondary>.alpha	Integer	0	指定自动二级光斑镜头效果的 Alpha 通道，取值范围为 0~255
<auto_secondary>.sourceZBuffer <auto_secondary>.Source_Z_Buffer	Boolean	False	设置是否使用 Z 缓冲高值来控制自动二级光斑镜头效果
<auto_secondary>.zHi <auto_secondary>.Z_Hi	Float	1000.0	Z-Buffer 用来存储对象与摄像机之间的距离，zHi 来指定对象与摄像机之间的最大距离，如果大于该值，就不进行自动二级光斑镜头效果处理。可动画
<auto_secondary>.zLo <auto_secondary>.Z_Lo	Float	0.0	Z-Buffer 用来存储对象与摄像机之间的距离，zLo 来指定对象与摄像机之间的最小距离，如果小于该值，就不进行自动二级光斑镜头效果处理。可动画
<auto_secondary>.filterAll <auto_secondary>.Filter_All	Boolean	True	是否为场景中所有对象指定自动二级光斑镜头效果
<auto_secondary>.filterEdge <auto_secondary>.Filter_Edge	Boolean	False	是否为场景中所有对象的边界指定自动二级光斑镜头效果

(续表)

属性名称	数据类型	默认值	说明
<auto_secondary>.FilterPerimeterAlpha <auto_secondary>.Filter_Perimeter_Alpha	Boolean	False	是否仅为场景中对象 Alpha 通道的边缘指定自动二级光斑镜头效果
<auto_secondary>.filterPerimeter <auto_secondary>.Filter_Perimeter	Boolean	False	是否仅为场景中对象的边缘指定自动二级光斑镜头效果
<auto_secondary>.filterBrightness <auto_secondary>.Filter_Brightness	Boolean	False	是否基于场景对象的亮度指定自动二级光斑镜头效果
<auto_secondary>.brightness <auto_secondary>.Bright	Integer	0	设置基于亮度指定自动二级光斑镜头效果的亮度值，只有亮度值高于本值的源对象才会被进行自动二级光斑镜头效果处理。可动画
<auto_secondary>.brightnessInvert <auto_secondary>.Filter_Brightness_Invert	Boolean	False	是否将属性.brightness 的数值反转
<auto_secondary>.filterHue <auto_secondary>.Filter_Hue	Boolean	False	设置是否基于场景对象的色调指定自动二级光斑镜头效果
<auto_secondary>.hueRange <auto_secondary>.Hue_Range	Integer	0	设置基于色相指定自动二级光斑镜头效果的色调值，只有色相值高于本值的源对象才会被进行自动二级光斑镜头效果处理。可动画
<auto_secondary>.hue	Color	(color 255 0 0)	可动画
<auto_secondary>.applyNoise <auto_secondary>.Apply_Additional_Effects	Boolean	False	设置是否使用噪波效果
<auto_secondary>.noiseMap <auto_secondary>.Noise_Map	TextureMap	undefined	指定噪波贴图
<auto_secondary>.radial_density	SubAnim		设置放射线密度
<auto_secondary>.radialDensity <auto_secondary>.Radial_Density_Map	TextureMap	undefined	为放射线密度指定贴图
<auto_secondary>.useRadialDensity <auto_secondary>.Apply_Radial_Density_Map	Boolean	False	设置是否使用放射线密度贴图

(续表)

属性名称	数据类型	默认值	说明
<auto_secondary. radial_falloff>.curve_1	SubAnim		设置径向颜色中使用的颜色的权重。本属性包含环绕色彩衰减曲线，用户不能创建或存取曲线上的点（除非该点有设置动画。这种情况下，用户可以将点的位置作为属性.curve_1 的一个动画来存取。）
<auto_secondary. circular_falloff>.curve_1	SubAnim		设置环绕颜色中使用的颜色的权重。本属性包含环绕色彩衰减曲线，用户不能创建或存取曲线上的点（除非该点有设置动画。这种情况下，用户可以将点的位置作为属性.curve_1 的一个动画来存取。）
<auto_secondary. radial_size>.curve_1	SubAnim		确定围绕特定镜头效果的径向大小。本属性包含代表镜头效果的径向大小的曲线。用户不能创建或存取曲线上的点（除非该点有设置动画。这种情况下，用户可以将点的位置作为属性.curve_1 的一个动画来存取。）
<auto_secondary. radial_density>.curve_1	SubAnim		确定希望应用其他效果的位置和程度。本属性包含曲线代表应用于镜头效果的其他效果的密度。用户不能创建或存取曲线上的点（除非该点有设置动画。这种情况下，用户可以将点的位置作为属性.curve_1 的一个动画来存取。）

**注意** 方法 le.addASec <Lens\_Effects> 如果添加操作成功，返回 True；否则，返回 False。一旦效果被添加到镜头效果里，就可以作为镜头效果的一个属性来进行存取，其属性名称为.auto\_secondary，如果同一类型的自动二级光斑镜头效果被添加，第一个效果元素使用上面的属性名，剩下的效果元素必须作为镜头效果的子动画进行存取，子动画的名称与属性名相同。

#### 14.9.2 Lens\_Effects—Glow（光晕镜头效果）

##### 构造函数

```
le.addGlow <Lens_Effects>
```

##### 属性

属性名称	数据类型	默认值	说明
<glow>.elementName <glow>.name	String	“Glow”	光晕镜头效果的名称
<glow>.on	Boolean	True	是否在场景渲染输出时激活光晕镜头效果
<glow>.size	Float	30.0	发光效果的尺寸。可动画
<glow>.intensity	Float	110.0	发光效果的亮度和不透明度。可动画
<glow>.objectsHide <glow>.Object_Hide	Boolean	True	设置当效果处在场景对象背面时是否依然可以显示
<glow>.occlusion	Float	100.0	确定镜头效果场景阻光度参数对特定效果的影响程度。可动画
<glow>.squeeze	Boolean	False	指定挤压设置是否影响光晕镜头效果

(续表)

属性名称	数据类型	默认值	说明
<glow>.useSourceColor <glow>.Source_Color	Float	0.0	指定光晕镜头效果与源灯光色彩或源对象色彩的混合程度,如果为0,仅使用源色彩;如果为100,仅使用效果色彩。可动画
<glow>.centerColor <glow>.Center_Color	Color	(color 255 255 255)	指定光晕镜头效果中心颜色值。可动画
<glow>.edgeColor <glow>.Edge_Color	Color	(color 255 0 0)	指定光晕镜头效果边界颜色值。可动画
<glow>.radialMtl <glow>.Radial_Material	TextureMap	undefined	指定放射贴图
<glow>.useRadialMtl <glow>.Apply_Radial_Color_Map	Boolean	False	设置是否使用放射贴图
<glow>.radial_falloff	SubAnim		设置放射径向衰减值
<glow>.radialMap <glow>.Radial_Falloff_Map	TextureMap	undefined	指定放射径向衰减贴图
<glow>.useRadialMap <glow>.Apply_adial_Falloff_Map	Boolean	False	设置是否使用放射径向衰减贴图
<glow>.circularColorMix <glow>.Circular_Color_Mix	Float	0.0	混合在径向颜色和环绕颜色中设置的颜色。如果设置为0,将只使用径向颜色中设置的值;如果设置为100,将只使用环绕颜色中设置的值。0~100之间的任何值将在两个值之间混合。可动画
<glow>.quadrant1Color <glow>.Quadrant_1_Color	Color	(color 255 0 0)	指定用于混合径向颜色和环绕颜色的第一个颜色值。可动画
<glow>.quadrant2Color <glow>.Quadrant_2_Color	Color	(color 255 0 0)	指定用于混合径向颜色和环绕颜色的第二个颜色值。可动画
<glow>.quadrant3Color <glow>.Quadrant_3_Color	Color	(color 255 0 0)	指定用于混合径向颜色和环绕颜色的第三个颜色值。可动画
<glow>.quadrant4Color <glow>.Quadrant_4_Color	Color	(color 255 0 0)	指定用于混合径向颜色和环绕颜色的第四个颜色值。可动画
<glow>.circularMtl <glow>.Circular_Material	TextureMap	undefined	指定定义环绕色彩效果的材质。可动画
<glow>.useCircularMtl <glow>.Apply_Circular_Color_Map	Boolean	False	指定是否使用材质来定义环绕色彩的效果

(续表)

属性名称	数据类型	默认值	说明
<glow>.circular_falloff	SubAnim		定义环绕色彩的衰减方式
<glow>.circularMap <glow>.Circular_Falloff_Map	TextureMap	undefined	指定环绕色彩衰减的贴图
<glow>.useCircularMap <glow>.Apply_Circular_Falloff_Map	Boolean	False	指定是否使用环绕色彩衰减的贴图
<glow>.radial_size	SubAnim		指定镜头效果的放射尺寸
<glow>.radialSizeMap <glow>.Radial_Size_Map	TextureMap	undefined	指定镜头效果的放射尺寸的贴图
<glow>.useRadialSizeMap <glow>.Apply_Radial_Size_Map	Boolean	False	指定是否使用贴图来定义放射型色彩的效果
<glow>.applyLights <glow>.Apply_to_Lights	Boolean	True	是否将效果指定到源灯光对象上
<glow>.applyImage <glow>.Apply_to_Image	Boolean	True	是否将效果指定到渲染输出的图像上
<glow>.applyImageCenters <glow>.Apply_to_Image_Centers	Boolean	False	是否将效果指定到对象的中心还是对象的一部分
<glow>.sourceObjectID <glow>.Source_Object_ID	Boolean	False	设置是否将镜头效果应用于场景中具有相应 G 缓冲区（或对象）ID 的特定对象。可动画
<glow>.objectID <glow>.Object_ID	Integer	1	设置对象通道 ID。可动画
<glow>.sourceEffectsID <glow>.Source_Effects_ID	Boolean	False	设置是否将效果指定给具有同一材质效果 ID 的对象。可动画
<glow>.effectsID	Integer	1	设置材质效果 ID。可动画
<glow>.sourceUnclampedColor <glow>.Source_Unclamped_Color	Boolean	False	设置是否使用超亮度颜色
<glow>.unclampedColor <glow>.Unclamped_Color	Float	1.0	指定最低像素颜色，像素颜色必须大于该值，才会进行光晕镜头效果处理
<glow>.unclampedColorInvert <glow>.Source_Unclamped_Color_Invert	Boolean	False	设置是否反转最低像素颜色

(续表)

属性名称	数据类型	默认值	说明
<glow>.sourceSurfaceNormal <glow>.Source_Surface_Normal	Boolean	False	设置是否使用表面法线来控制像素是否被进行效果处理
<glow>.surfaceNormal <glow>.Surface_Normal	Float	0.0	对象表面法线与摄像机之间的角度值, 大于指定值时, 该表面才会被进行发光处理。可动画
<glow>.surfaceNormalInvert <glow>.Source_Surface_Normal_Invert	Boolean	False	设置是否反转表面法线角度值
<glow>.sourceWhole <glow>.Source_Whole	Boolean	False	是否将发光效果指定到全部场景
<glow>.sourceAlpha <glow>.Source_Alpha	Boolean	False	是否将发光效果指定到某一 Alpha 通道
<glow>.alpha	Integer	0	指定发光效果的 Alpha 通道, 取值范围为 0~255。可动画
<glow>.sourceZBuffer <glow>.Source_Z_Buffer	Boolean	False	设置是否使用 Z 缓冲值来控制发光效果
<glow>.zHi <glow>.Z_Hi	Float	1000.0	Z-Buffer 用来存储对象与摄像机之间的距离, zHi 来指定对象与摄像机之间的最大距离, 如果大于该值, 就不进行发光效果处理。可动画
<glow>.zLo <glow>.Z_Lo	Float	0.0	Z-Buffer 用来存储对象与摄像机之间的距离, zLo 来指定对象与摄像机之间的最小距离, 如果小于该值, 就不进行发光效果处理。可动画
<glow>.filterAll <glow>.Filter_All	Boolean	True	是否为场景中所有对象指定发光效果
<glow>.filterEdge <glow>.Filter_Edge	Boolean	False	是否为场景中所有对象的边界指定发光效果
<glow>.filterPerimeterAlpha <glow>.Filter_Perimeter_Alpha	Boolean	False	是否仅为场景中对象 Alpha 通道的边缘指定发光效果
<glow>.filterPerimeter <glow>.Filter_Perimeter	Boolean	False	是否仅为场景中对象的边缘指定发光效果
<glow>.filterBrightness <glow>.Filter_Brightness	Boolean	False	是否基于场景对象的亮度指定发光效果
<glow>.brightness <glow>.Bright	Integer	0	设置基于亮度指定发光效果的亮度值。高于本值的源对象才会被进行发光处理。可动画

(续表)

属性名称	数据类型	默认值	说明
<glow>.brightnessInvert<glow> Filter_Brightness_Invert	Boolean	False	是否将属性.brightness 的数值反转
<glow>.filterHue <glow>.Filter_Hue	Boolean	alse	是否基于场景对象的色调指定发光效果
<glow>.hueRange <glow>.Hue_Range	Integer	0	设置基于色相指定发光效果的色调值，只有色相值高于本值的源对象才会被进行发光处理。可动画
<glow>.hue	Color	(color 255 0 0)	可动画
<glow>.applyNoise <glow>.Apply_additional_Effects	Boolean	False	设置是否使用噪波效果
<glow>.noiseMap <glow>.Noise_Map	TextureMap	undefined	指定噪波贴图
<glow>.radial_density	SubAnim		设置放射线密度
glow>.radialDensity <glow>.Radial_Density_Map	TextureMap	undefined	为放射线密度指定贴图
<glow>.useRadialDensity <glow>.Apply_Radial_Density_Map	Boolean	False	设置是否使用放射线密度贴图
<glow>.radial_falloff>.curve_1	SubAnim		设置径向颜色中使用的颜色的权重。本属性包含环绕色彩衰减曲线，用户不能创建或存取曲线上的点（除非该点有设置动画。这种情况下，用户可以将点的位置作为属性.curve_1 的一个动画来存取。）
<glow>.circular_falloff>.curve_1	SubAnim		设置环绕颜色中使用的颜色的权重。本属性包含环绕色彩衰减曲线，用户不能创建或存取曲线上的点（除非该点有设置动画。这种情况下，用户可以将点的位置作为属性.curve_1 的一个动画来存取。）
<glow>.radial_size>.curve_1	SubAnim		确定围绕特定镜头效果的径向大小。本属性包含代表镜头效果的径向大小的曲线。用户不能创建或存取曲线上的点（除非该点有设置动画。这种情况下，用户可以将点的位置作为属性.curve_1 的一个动画来存取。）
<glow>.radial_density>.curve_1	SubAnim		确定希望应用其他效果的位置和程度。本属性包含曲线代表应用于镜头效果的其他效果的密度。用户不能创建或存取曲线上的点（除非该点有设置动画。这种情况下，用户可以将点的位置作为属性.curve_1 的一个动画来存取。）

**注意** 方法 le.addGlow <Lens\_Effects> 如果添加操作成功，返回 True；否则，返回 False。一旦效果被添加到镜头效果里，就可以作为镜头效果的一个属性来进行存取，其属性名称为.glow，如果同一类型的发光效果被添加，第一个效果元素使用上面属性名，剩下的效果元素必须作为镜头效果的子动画进行存取，子动画的名称与属性名相同。

### 14.9.3 Lens\_Effects-Manual\_Secondary（手动二级光斑镜头效果）

#### 构造函数

```
le.addMSec <Lens_Effects>
```

#### 属性

属性名称	数据类型	默认值	说明
<manual_secondary>.elementName <manual_secondary>.name	String	“Manual Secondary”	指定手动二级光斑镜头效果的名称
<manual_secondary>.on	Boolean	True	设置效果在场景渲染输出时是否被激活
<manual_secondary>.size	Float	50.0	可动画
<manual_secondary>.intensity	Float	30.0	设置效果的亮度和不透明度。数值越大，效果越明亮，越不透明。可动画
<manual_secondary>.plane	Float	150.0	可动画
<manual_secondary>.colorSource <manual_secondary>.Source_Color	Float	20.0	指定手动光斑效果与源灯光色彩或源对象色彩的混合程度，如果为0，仅使用源色彩；如果为100，仅使用效果色彩。可动画
<manual_secondary>.sides	Integer	0	指定光斑组中的形态： 0：圆形；1：三角形；2：四边形； 3：五边形；4：六边形；5：七边形；6：八边形
<manual_secondary>.occlusion	Float	100.0	确定镜头效果场景阻光度参数对特定效果的影响程度。可动画
<manual_secondary>.squeeze	Boolean	False	指定挤压设置是否影响镜头二级光斑镜头效果
<manual_secondary>.presets	Integer	0	指定选择 Presets 下拉列表里选择哪一项作为预设置（如 Rainbow） 如果为0，表示无预设置
<manual_secondary>.radialColor1 <manual_secondary>.Radial_Color_1	Color	(color 0 0 0)	设置第一个影响光斑效果的放射色彩。可动画
<manual_secondary>.radialColor2 <manual_secondary>.Radial_Color_2	Color	(color 57 34 89)	设置第二个影响光斑效果的放射色彩。可动画
<manual_secondary>.radialColor3 <manual_secondary>.Radial_Color_3	Color	(color 85 139 85)	设置第三个影响光斑效果的放射色彩。可动画
<manual_secondary>.radialColor4 <manual_secondary>.Radial_Color_4	Color	(color 190 99 10)	设置第四个影响光斑效果的放射色彩。可动画

(续表)

属性名称	数据类型	默认值	说明
<manual_secondary>.colorRadius1 <manual_secondary>.Radius_Color_1	Float	90.0	设置第一个放射色彩的影响半径。可动画
<manual_secondary>.colorRadius2 <manual_secondary>.Radius_Color_2	Float	92.0	设置第二个放射色彩的影响半径。可动画
<manual_secondary>.colorRadius3 <manual_secondary>.Radius_Color_3	Float	95.0	设置第三个放射色彩的影响半径。可动画
<manual_secondary>.colorRadius4 <manual_secondary>.Radius_Color_4	Float	98.0	设置第四个放射色彩的影响半径。可动画
<manual_secondary>.radialMtl <manual_secondary>.Radial_Material	TextureMap	undefined	指定径向颜色的贴图
<manual_secondary>.useRadialMtl <manual_secondary>.Apply_Radial_Color_Map	Boolean	False	设置是否使用贴图确定径向颜色
<manual_secondary>.radial_falloff	SubAnim		
<manual_secondary>.radialMap <manual_secondary>.Radial_Falloff_Map	TextureMap	undefined	指定径向衰减贴图
<manual_secondary>.useRadialMap <manual_secondary>.Apply_Radial_Falloff_Map	Boolean	False	指定是否使用径向衰减贴图
<manual_secondary>.circularColorMix <manual_secondary>.Circular_Color_Mix	Float	0.0	混合在径向颜色和环绕颜色中设置的颜色。如果设置为 0，将只使用径向颜色中设置的值；如果设置为 100，将只使用环绕颜色中设置的值。0~100 之间的任何值将在两个值之间混合。可动画
<manual_secondary>.quadrant1Color <manual_secondary>.Quadrant_1_Color	Color	(color 255 0 0)	指定用于混合径向颜色和环绕颜色的第一个颜色值。可动画
<manual_secondary>.quadrant2Color <manual_secondary>.Quadrant_2_Color	Color	(color 255 0 0)	指定用于混合径向颜色和环绕颜色的第二个颜色值。可动画
<manual_secondary>.quadrant3Color <manual_secondary>.Quadrant_3_Color	Color	(color 255 0 0)	指定用于混合径向颜色和环绕颜色的第三个颜色值。可动画

(续表)

属性名称	数据类型	默认值	说明
<manual_secondary>.quadrant4Color <manual_secondary>.Quadrant_4_Color	Color	(color 255 0 0)	指定用于混合径向颜色和环绕颜色的第四个颜色值。可动画
<manual_secondary>.circularMtl <manual_secondary>.Circular_Material	TextureMap	undefined	指定定义环绕色彩效果的材质。可动画
<manual_secondary>.useCircularMtl <manual_secondary>.Apply_Circular_Color_Map	Boolean	False	指定是否使用材质来定义环绕色彩的效果
<manual_secondary>.circular_falloff	SubAnim		定义环绕色彩的衰减方式
<manual_secondary>.circularMap <manual_secondary>.Circular_Falloff_Map	TextureMap	undefined	指定环绕色彩衰减的贴图
<manual_secondary>.useCircularMap <manual_secondary>.Apply_Circular_Falloff_Map	Boolean	False	指定是否使用贴图来定义环绕色彩的效果
<manual_secondary>.radial_size	SubAnim		指定手动镜头效果的放射尺寸
<manual_secondary>.radialSizeMap <manual_secondary>.Radial_Size_Map	TextureMap	undefined	为镜头效果的放射尺寸指定贴图
<manual_secondary>.useRadialSizeMap <manual_secondary>.Apply_Radial_Size_Map	Boolean	False	指定是否使用贴图来定义放射型色彩的效果
<manual_secondary>.applyLights <manual_secondary>.Apply_to_Lights	Boolean	True	是否将效果指定到源灯光对象上
<manual_secondary>.applyImage <manual_secondary>.Apply_to_Image	Boolean	False	是否将效果指定到渲染输出的图像上
<manual_secondary>.applyImageCenters <manual_secondary>.apply_to_Image_Centers	Boolean	False	是否将效果指定到对象的中心还是对象的一部分
<manual_secondary>.sourceObjectID <manual_secondary>.Source_Object_ID	Boolean	False	设置是否将镜头效果应用于场景中具有相应 G 缓冲区(或对象) ID 的特定对象。可动画

(续表)

属性名称	数据类型	默认值	说明
<manual_secondary>.objectID <manual_secondary>.Object_ID	Integer	1	设置对象通道 ID。可动画
<manual_secondary>.sourceEffectsID <manual_secondary>.Source_Effects_ID	Boolean	False	设置是否将效果指定给具有同一材质效果 ID 的对象。可动画
<manual_secondary>.effectsID <manual_secondary>.Effects_ID	Integer	0	设置材质效果 ID。可动画
<manual_secondary>.sourceUnclampedColor <manual_secondary>.Source_Unclamped_Color	Boolean	False	设置是否使用超亮度颜色
<manual_secondary>.unclampedColor <manual_secondary>.Unclamped_Color	Float	1.0	指定最低像素颜色，像素颜色必须大于该值，才会进行光晕镜头效果处理。可动画
<manual_secondary>.unclampedColorInvert <manual_secondary>.Source_Unclamped_Color_Invert	Boolean	False	设置是否反转最低像素颜色
<manual_secondary>.sourceSurfaceNormal <manual_secondary>.Source_Surface_Normal	Boolean	False	设置是否使用表面法线来控制像素是否被进行效果处理
<manual_secondary>.surfaceNormal <manual_secondary>.Surface_Normal	Float	0.0	对象表面法线与摄像机之间的角度值，大于指定值时，该表面才会被进行发光处理。可动画
<manual_secondary>.surfaceNormalInvert <manual_secondary>.Source_Surface_Normal_Invert	Boolean	False	设置是否反转表面法线角度值
<manual_secondary>.sourceWhole <manual_secondary>.Source_Whole	Boolean	False	是否将手动二级光斑镜头效果指定到全部场景
<manual_secondary>.sourceAlpha <manual_secondary>.Source_Alpha	Boolean	False	是将手动二级光斑镜头效果指定到某一 Alpha 通道
<manual_secondary>.alpha	Integer	0	指定手动二级光斑镜头效果的 Alpha 通道，取值范围为 0~255
<manual_secondary>.sourceZBuffer <manual_secondary>.Source_Z_Buffer	Boolean	False	设置是否使用 Z 缓冲高值来控制手动二级光斑镜头效果

(续表)

属性名称	数据类型	默认值	说明
<manual_secondary>.zHi <manual_secondary>.Z_Hi	Float	1000.0	Z-Buffer 用来存储对象与摄像机之间的距离, zHi 来指定对象与摄像机之间的最大距离, 如果大于该值, 就不进行手动二级光斑镜头效果处理。可动画
<manual_secondary>.zLo <manual_secondary>.Z_Lo	Float	0.0	Z-Buffer 用来存储对象与摄像机之间的距离, zLo 来指定对象与摄像机之间的最小距离, 如果小于该值, 就不进行手动二级光斑镜头效果处理。可动画
<manual_secondary>.filterAll <manual_secondary>.Filter_All	Boolean	True	是否为场景中所有对象指定手动二级光斑镜头效果
<manual_secondary>.filterEdge <manual_secondary>.Filter_Edge	Boolean	False	是否为场景中所有对象的边界指定手动二级光斑镜头效果
<manual_secondary>. filterPerimeterAlpha <manual_secondary>. Filter_Perimeter_Alpha	Boolean	False	是否仅为场景中对象 Alpha 通道的边缘指定手动二级光斑镜头效果
<manual_secondary>. filterPerimeter <manual_secondary>. Filter_Perimeter	Boolean	False	是否仅为场景中对象的边缘指定手动二级光斑镜头效果
<manual_secondary>. filterBrightness <manual_secondary>. Filter_Brightness	Boolean	False	是否基于场景对象的亮度指定手动二级光斑镜头效果
<manual_secondary>. Brightness <manual_secondary>. Bright	Integer	0	设置基于亮度指定手动光斑效果的亮度值, 只有亮度值高于本值的源对象才会被进行手动二级光斑处理。可动画
<manual_secondary>. brightnessInvert <manual_secondary>. Filter_Brightness_Invert	Boolean	False	是否将属性.brightness 的数值反转
<manual_secondary>.filterHue <manual_secondary>.Filter_Hue	Boolean	False	是否基于场景对象的色调指定手动二级光斑镜头效果
<manual_secondary>.hueRange <manual_secondary>.Hue_Range	Integer	0	设置基于色相指定发光效果的色调值, 只有色相值高于本值的源对象才会被进行手动二级光斑处理。可动画
<manual_secondary>.hue	Color	(color 255 0 0)	可动画
<manual_secondary>.applyNoise <manual_secondary>. Apply_Additional_Effects	Boolean	False	设置是否使用噪波效果

(续表)

属性名称	数据类型	默认值	说明
<manual_secondary>.noiseMap <manual_secondary>.Noise_Map	TextureMap	undefined	指定噪波贴图
<manual_secondary>.radial_density <manual_secondary>.Radial_Density_Map	SubAnim		设置放射线密度
<manual_secondary>.useRadialDensity <manual_secondary>.Apply_Radial_Density_Map	Boolean	False	设置是否使用放射线密度贴图
<manual_secondary>.radial_falloff>.curve_1	SubAnim		设置径向颜色中使用的颜色的权重。本属性包含环绕色彩衰减曲线，用户不能创建或存取曲线上的点（除非该点有设置动画。这种情况下，用户可以将点的位置作为属性.curve_1 的一个动画来存取。）
<manual_secondary>.circular_falloff>.curve_1	SubAnim		设置环绕颜色中使用的颜色的权重。本属性包含环绕色彩衰减曲线，用户不能创建或存取曲线上的点（除非该点有设置动画。这种情况下，用户可以将点的位置作为属性.curve_1 的一个动画来存取。）
<manual_secondary>.radial_size>.curve_1	SubAnim		确定围绕特定镜头效果的径向大小。本属性包含代表镜头效果的径向大小的曲线。用户不能创建或存取曲线上的点（除非该点有设置动画。这种情况下，用户可以将点的位置作为属性.curve_1 的一个动画来存取。）
<manual_secondary>.radial_density>.curve_1	SubAnim		确定希望应用其他效果的位置和程度。本属性包含曲线代表应用于镜头效果的其他效果的密度。用户不能创建或存取曲线上的点（除非该点有设置动画。这种情况下，用户可以将点的位置作为属性.curve_1 的一个动画来存取。）

**注意**

方法 le.ddMSec <Lens\_Effects> 如果添加操作成功，返回 True；否则，返回 False。一旦效果被添加到镜头效果里，就可以作为镜头效果的一个属性来进行存取，其属性名称为.manual\_secondary，如果同一类型的发光效果被添加，第一个效果元素使用上面的属性名，剩下的效果元素必须作为镜头效果的子动画进行存取，子动画的名称与属性名相同。

#### 14.9.4 Lens\_Effects-Ray（射线镜头效果）

##### 构造函数

```
le.addRay <Lens_Effects>
```

##### 属性

属性名称	数据类型	默认值	说明
<ray>.elementName <ray>.name	String	“Ray”	指定射线镜头效果的名称
<ray>.on	Boolean	True	设置效果在场景渲染输出时是否被激活
<ray>.size	Float	300.0	指定射线镜头效果的尺寸。可动画
<ray>.intensity	Float	10.0	设置效果的亮度和不透明度。数值越大，效果越明亮，越不透明。可动画
<ray>.number <ray>.Quantity	Integer	100	指定出现在射线镜头效果中射线的数量。可动画
<ray>.angle	Float	0.0	指定射线的角度，可以为正值或负值。可动画
<ray>.sharpness	Float	8.0	指定射线的锐利程度，较高的数值创建锐利、边沿清晰的射线镜头效果，较低的数值创建发光、边沿虚化的射线镜头效果，取值范围为0~10。可动画
<ray>.objectsHide	Boolean	False	设置当效果处在场景对象背面时是否依然可以显示
<ray>.occlusion	Float	100.0	确定镜头效果场景阻光度参数对特定效果的影响程度。可动画
<ray>.squeeze	Boolean	False	指定挤压设置是否影响射线镜头效果
<ray>.colorSource <ray>.Source_Color	Float	100.0	指定射线镜头效果与源灯光色彩或源对象色彩的混合程度。如果为0，仅使用源色彩；如果为100，仅使用效果色彩。可动画
<ray>.centerColor <ray>.Center_Color	Color	默认值：(color 255 255 255) 指定镜头效果的中心颜色值。可动画	
<ray>.edgeColor <ray>.Edge_Color	Color		
<ray>.radialMtl <ray>.Radial_Material	TextureMap	undefined	指定径向颜色的贴图
<ray>.useRadialMtl <ray>.Apply_Radial_Color_Map	Boolean	False	设置是否使用贴图确定径向颜色
<ray>.radial_falloff	SubAnim		指定衰减方式
<ray>.radialMap <ray>.Radial_Falloff_Map	TextureMap	undefined	指定径向衰减贴图
<ray>.useRadialMap <ray>.Apply_Radial_Falloff_Map	Boolean	False	指定是否使用径向衰减贴图
<ray>.circularColorMix <ray>.Circular_Color_Mix	Float	0.0	混合在径向颜色和环绕颜色中设置的颜色。如果设置为0，将只使用径向颜色中设置的值；如果设置为100，将只使用环绕颜色中设置的值。0~100之间的任何值将在两个值之间混合。可动画

(续表)

属性名称	数据类型	默认值	说明
<ray>.quadrant1Color <ray>.Quadrant_1_Color	Color	默认值: (color 255 0 0)	
<ray>.quadrant2Color <ray>.Quadrant_2_Color		指定用于混合径向颜色和环绕颜色的第一个颜色值。可动画	
<ray>.quadrant3Color <ray>.Quadrant_3_Color	Color	默认值: (color 255 0 0)	
<ray>.quadrant4Color <ray>.Quadrant_4_Color		指定用于混合径向颜色和环绕颜色的第三个颜色值。可动画	
<ray>.circularMtl <ray>.Circular_Material	TextureMap	undefined	指定定义环绕色彩效果的材质
<ray>.useCircularMtl <ray>.Apply_Circular _Color_Map	Boolean	False	指定是否使用材质来定义环绕色彩的效果
<ray>.circular_falloff	SubAnim		定义环绕色彩的衰减方式
<ray>.circularMap <ray>. Circular_Falloff_Map	TextureMap	undefined	指定定义环绕色彩效果的贴图
<ray>.useCircularMap <ray>.Apply_Circular _Falloff_Map	Boolean	False	指定是否使用贴图来定义环绕色彩的效果
<ray>.radial_size	SubAnim		指定射线的放射尺寸
<ray>.radialSizeMap <ray>.Radial_Size_Map	TextureMap	undefined	为射线的放射尺寸指定贴图
<ray>.useRadialSizeMap <ray>. Apply_Radial_Size_Map	Boolean	False	设置是否使用射线的放射尺寸贴图
<ray>.applyLights <ray>.Apply_to_Lights	Boolean	True	是否将效果指定到源灯光对象上
<ray>.applyImage <ray>.Apply_to_Image	Boolean	True	是否将效果指定到渲染输出的图像上
<ray>.applyImageCenters <ray>. Apply_to_Image_Centers	Boolean	False	是否将效果指定到对象的中心还是对象的一部分
<ray>.sourceObjectID <ray>.Source_Object_ID	Boolean	False	设置是否将镜头效果应用于场景中具有相应 G 缓冲区(或对象) ID 的特定对象。可动画
<ray>.objectID <ray>.Object_ID	Integer	1	设置对象通道 ID。可动画
<ray>.sourceEffectsID <ray>.Source_Effects_ID	Boolean	False	设置是否将效果指定给具有同一材质效果 ID 的对象。可动画
<ray>.effectsID <ray>.Effects_ID	Integer	1	设置材质效果 ID。可动画

(续表)

属性名称	数据类型	默认值	说明
<ray>.sourceUnclampedColor <ray>.Source_Unclamped_Color	Boolean	False	设置是否使用超亮度颜色
<ray>.unclampedColor <ray>.Unclamped_Color	Float	1.0	指定最低像素颜色, 像素颜色必须大于该值, 才会进行射线镜头效果处理。可动画
<ray>.unclampedColorInvert <ray>.Source_Unclamped_Color_Invert	Boolean	False	设置是否反转最低像素颜色
<ray>.sourceSurfaceNormal <ray>.Source_Surface_Normal	Boolean	False	设置是否使用表面法线来控制像素是否被进行效果处理
<ray>.surfaceNormal <ray>.Surface_Normal	Float	0.0	对象表面法线与摄像机之间的角度值, 大于指定值时, 该表面才会被进行射线处理。可动画
<ray>.surfaceNormalInvert <ray>.Source_Surface_Normal_Invert	Boolean	False	设置是否反转表面法线角度值
<ray>.sourceWhole <ray>.Source_Whole	Boolean	False	是否将射线镜头效果指定到全部场景
<ray>.sourceAlpha <ray>.Source_Alpha	Boolean	False	是将射线镜头效果指定到某一 Alpha 通道
<ray>.alpha	Integer	0	指定射线镜头效果的 Alpha 通道, 取值范围为 0~255。可动画
<ray>.sourceZBuffer <ray>.Source_Z_Buffer	Boolean	False	设置是否使用 Z 缓冲高值来控制射线镜头效果。
<ray>.zHi <ray>.Z_Hi	Float	1000.0	Z-Buffer 用来存储对象与摄像机之间的距离, zHi 来指定对象与摄像机之间的最大距离, 如果大于该值, 就不进行射线镜头效果处理。可动画
<ray>.zLo <ray>.Z_Lo	Float	0.0	Z-Buffer 用来存储对象与摄像机之间的距离, zLo 来指定对象与摄像机之间的最小距离, 如果小于该值, 就不进行射线镜头效果处理。可动画
<ray>.filterAll <ray>.Filter_All	Boolean	True	是否为场景中所有对象指定射线镜头效果
<ray>.filterEdge <ray>.Filter_Edge	Boolean	False	是否为场景中所有对象的边界指定射线镜头效果
<ray>.filterPerimeterAlpha <ray>.Filter_Perimeter_Alpha	Boolean	False	是否仅为场景中对象 Alpha 通道的边缘指定射线镜头效果
<ray>.filterPerimeter <ray>.Filter_Perimeter	Boolean	False	是否仅为场景中对象的边缘指定射线镜头效果

(续表)

属性名称	数据类型	默认值	说明
<ray>.filterBrightness <ray>.Filter_Brightness	Boolean	False	是否基于场景对象的亮度指定射线镜头效果
<ray>.brightness <ray>.Bright	Integer	0	设置基于亮度指定手动光斑效果的亮度值，只有亮度值高于本值的源对象才会被进行射线处理。可动画
<ray>.brightnessInvert <ray>. Filter_Brightness_Invert	Boolean	False	是否将属性.brightness 的数值反转
<ray>.filterHue <ray>.Filter_Hue	Boolean	False	是否基于场景对象的色调指定射线镜头效果
<ray>.hueRange <ray>.Hue_Ra nge	Integer	0	设置基于色相指定射线镜头效果的色调值，只有色相值高于本值的源对象才会被进行射线处理。可动画
<ray>.hue	Color	(color 255 0 0)	设置色相值。可动画
<ray>.applyNoise <ray>.Apply_ Additional_Effects	Boolean	False	设置是否使用噪波效果
<ray>.noiseMap <ray>.Noise_Map	TextureMap	undefined	指定噪波贴图
<ray>.radial_density	SubAnim		设置放射线密度
<ray>.radialDensity <ray>. Radial_Density_Map	TextureMap	undefined	为放射线密度指定贴图
<ray>.useRadialDensity <ray>.Apply_ Radial_Density_Map	Boolean	False	设置是否使用放射线密度贴图
<ray>.radial_falloff> curve_1	SubAnim		设置径向颜色中使用的颜色的权重。本属性包含环绕色彩衰减曲线，用户不能创建或存取曲线上的点（除非该点有设置动画。这种情况下，用户可以将点的位置作为属性.curve_1 的一个动画来存取。）
<ray>.circular_falloff> curve_1	SubAnim		设置环绕颜色中使用的颜色的权重。本属性包含环绕色彩衰减曲线，用户不能创建或存取曲线上的点（除非该点有设置动画。这种情况下，用户可以将点的位置作为属性.curve_1 的一个动画来存取。）
<ray>.radial_size> curve_1	SubAnim		确定围绕特定镜头效果的径向大小。本属性包含代表镜头效果的径向大小的曲线。用户不能创建或存取曲线上的点（除非该点有设置动画。这种情况下，用户可以将点的位置作为属性.curve_1 的一个动画来存取。）
<ray>.radial_density> curve_1	SubAnim		确定希望应用其他效果的位置和程度。本属性包含曲线代表应用于镜头效果的其他效果的密度。用户不能创建或存取曲线上的点（除非该点有设置动画。这种情况下，用户可以将点的位置作为属性.curve_1 的一个动画来存取。）

**注意**

方法 le.addRay <Lens\_Effects> 如果添加操作成功，返回 True；否则，返回 False。一旦效果被添加到镜头效果里，就可以作为镜头效果的一个属性来进行存取，其属性名称为 .ray，如果同一类型的射线镜头效果被添加，第一个效果元素使用上面的属性名，剩下的效果元素必须作为镜头效果的子动画进行存取，子动画的名称与属性名相同。

### 14.9.5 Lens\_Effects—Ring（光环镜头效果）

#### 构造函数

le.addRing <Lens\_Effects>

#### 属性

属性名称	数据类型	默认值	说明
<ring>.elementName <ring>.name	String	“Ring”	指定光环镜头效果的名称
<ring>.on	Boolean	True	设置效果在场景渲染输出时是否被激活
<ring>.size	Float	75.0	指定光环镜头效果的尺寸。可动画
<ring>.intensity	Float	75.0	单独设置光环镜头效果的亮度和不透明度。数值越大，效果越明亮，越不透明。可动画
<ring>.plane	Float	0.0	指定光环镜头效果的平面位置。可动画
<ring>.thickness	Float	10.0	指定光环镜头效果的粗细，单位为像素。可动画
<ring>.objectsHide <ring>.Object_Hide	Boolean	False	设置当效果处在场景对象背面时是否依然可以显示
<ring>.occlusion	Float	100.0	确定镜头效果场景阻光度参数对特定效果的影响程度。可动画
<ring>.squeeze	Boolean	False	指定挤压设置是否影响光环镜头效果。
<ring>.colorSource <ring>.Source_Color	Float	0.0	指定手动光环镜头效果与源灯光色彩或源对象色彩的混合程度，如果为 0，仅使用源色彩，如果为 100，仅使用效果色彩。可动画
<ring>.centerColor <ring>.Center_Color	Color	默认值：(color 255 255 255) 可动画	
<ring>.edgeColor <ring>.Edge_Color	Color	默认值：(color 255 0 0) 可动画	
<ring>.radialMtl <ring>.Radial_Material	TextureMap	undefined	指定径向颜色的贴图
<ring>.useRadialMtl <ring>.Apply_Radial_Color_Map	Boolean	False	设置是否使用贴图确定径向颜色
<ring>.radial_falloff	SubAnim		指定放射的径向衰减值

(续表)

属性名称	数据类型	默认值	说明
<ring>.radialMap <ring>.Radial_Falloff_Map	TextureMap	undefined	指定放射的径向衰减贴图
<ring>.useRadialMap <ring>.Apply_Radial_Falloff_Map	Boolean	False	指定是否使用径向衰减贴图
<ring>.circularColorMix <ring>.Circular_Color_Mix	Float	0.0	混合在径向颜色和环绕颜色中设置的颜色。如果设置为0，将只使用径向颜色中设置的值；如果设置为100，将只使用环绕颜色中设置的值。0~100之间的任何值将在两个值之间混合。可动画
<ring>.quadrant1Color <ring>.Quadrant_1_Color	Color	(color 255 0 0)	指定用于混合径向颜色和环绕颜色的第一个颜色值。可动画
<ring>.quadrant2Color <ring>.Quadrant_2_Color	Color	(color 255 0 0)	指定用于混合径向颜色和环绕颜色的第二个颜色值。可动画
<ring>.quadrant3Color <ring>.Quadrant_3_Color	Color	(color 255 0 0)	指定用于混合径向颜色和环绕颜色的第三个颜色值。可动画
<ring>.quadrant4Color <ring>.Quadrant_4_Color	Color	(color 255 0 0)	指定用于混合径向颜色和环绕颜色的第四个颜色值。可动画
<ring>.circularMtl <ring>.Circular_Material	TextureMap	undefined	指定环绕颜色的贴图
<ring>.useCircularMtl <ring>.Apply_Circular_Color_Map	Boolean	False	设置是否使用贴图确定环绕颜色
<ring>.circular_falloff	SubAnim		指定环绕衰减方式
<ring>.circularMap <ring>.Circular_Falloff_Map	TextureMap	undefined	指定环绕衰减贴图
<ring>.useCircularMap <ring>.Apply_Circular_Falloff_Map	Boolean	False	指定是否使用环绕衰减贴图
<ring>.radial_size	SubAnim		指定光环镜头效果的放射尺寸
<ring>.radialSizeMap <ring>.Radial_Size_Map	TextureMap	undefined	为光环镜头效果的放射尺寸指定贴图
<ring>.useRadialSizeMap <ring>.Apply_Radial_Size_Map	Boolean	False	设置环镜头效果是否使用光放射尺寸贴图
<ring>.applyLights <ring>.Apply_to_Lights	Boolean	True	是否将效果指定到源灯光对象上
<ring>.applyImage <ring>.Apply_to_Image	Boolean	True	是否将效果指定到渲染输出的图像上
<ring>.applyImageCenters <ring>.Apply_to_Image_Centers	Boolean	False	是否将效果指定到对象的中心还是对象的一部分

(续表)

属性名称	数据类型	默认值	说明
<ring>.sourceObjectID <ring>.Source_Object_ID	Boolean	False	设置是否将镜头效果应用于场景中具有相应 G 缓冲区(或对象) ID 的特定对象。可动画
<ring>.objectID <ring>.Object_ID	Integer	1	设置对象通道 ID。可动画
<ring>.sourceEffectsID <ring>.Source_Effects_ID	Boolean	False	设置是否将效果指定给具有同一材质效果 ID 的对象。可动画
<ring>.effectsID <ring>.Effects_ID	Integer	1	设置材质效果 ID。可动画
<ring>.sourceUnclampedColor <ring>.Source_Unclamped_Color	Boolean	False	设置是否使用超亮度颜色
<ring>.unclampedColor <ring>.Unclamped_Color	Float	1.0	指定最低像素颜色, 像素颜色必须大于该值, 才会进行光环镜头效果处理。可动画
<ring>.unclampedColorInvert <ring>.Source_Unclamped_Color_Invert	Boolean	False	设置是否反转最低像素颜色
<ring>.sourceSurfaceNormal <ring>.Source_Surface_Normal	Boolean	False	设置是否使用表面法线来控制像素是否被进行效果处理。可动画
<ring>.surfaceNormal <ring>.Surface_Normal	Float	0.0	对象表面法线与摄像机之间的角度值, 大于指定值时, 该表面才会被进行光环处理
<ring>.surfaceNormalInvert <ring>.Source_Surface_Normal_Invert	Boolean	False	设置是否反转表面法线角度值
<ring>.sourceWhole <ring>.Source_Whole	Boolean	False	是否将光环镜头效果指定到全部场景
<ring>.sourceAlpha <ring>.Source_Alpha	Boolean	False	是将光环镜头效果指定到某一 Alpha 通道
<ring>.alpha	Integer	0	指定光环镜头效果的 Alpha 通道, 取值范围为 0~255。可动画
<ring>.sourceZBuffer <ring>.Source_Z_Buffer	Boolean	False	设置是否使用 Z 缓冲高值来控制光环镜头效果
<ring>.zHi <ring>.Z_Hi	Float	1000.0	Z-Buffer 用来存储对象与摄像机之间的距离, zHi 来指定对象与摄像机之间的最大距离, 如果大于该值, 就不进行光环镜头效果处理。可动画

(续表)

属性名称	数据类型	默认值	说明
ring>.zLo <ring>.Z_Lo	Float	0.0	Z-Buffer 用来存储对象与摄像机之间的距离, zLo 来指定对象与摄像机之间的最小距离, 如果小于该值, 就不进行光环镜头效果处理。可动画
<ring>.filterAll <ring>.Filter_All	Boolean	True	是否为场景中所有对象指定光环镜头效果
<ring>.filterEdge <ring>.Filter_Edge	Boolean	False	是否为场景中所有对象的边界指定光环镜头效果
<ring>.filterPerimeterAlpha <ring>.Filter_Perimeter_Alpha	Boolean	False	是否仅为场景中对象 Alpha 通道的边缘指定光环镜头效果
<ring>.filterPerimeter <ring>.Filter_Perimeter	Boolean	False	是否仅为场景中对象的边缘指定光环镜头效果
<ring>.filterBrightness <ring>.Filter_Brightness	Boolean	False	是否基于场景对象的亮度指定光环镜头效果
<ring>.brightness <ring>.Bright	Integer	0	设置基于亮度指定手动光斑效果的亮度值, 只有亮度值高于本值的源对象才会被进行光环处理。可动画
<ring>.brightnessInvert <ring>.Filter_Brightness_Invert	Boolean	False	是否将属性.brightness 的数值反转
<ring>.filterHue <ring>.Filter_Hue	Boolean	False	是否基于场景对象的色调指定光环镜头效果
<ring>.hueRange <ring>.Hue_Range	Integer	0	设置基于色相指定光环镜头效果的色调值, 只有色相值高于本值的源对象才会被进行光环处理。可动画
<ring>.hue	Color	(color 255 0 0)	设置光环镜头效果的色相值。可动画
<ring>.applyNoise <ring>.Apply_Additional_Effects	Boolean	False	设置是否使用噪波效果
<ring>.noiseMap <ring>.Noise_Map	TextureMap	undefined	指定噪波贴图
<ring>.radial_density	SubAnim		设置放射线密度
<ring>.radialDensity <ring>.Radial_Density_Map	TextureMap	undefined	为放射线密度指定贴图
<ring>.useRadialDensity <ring>.Apply_Radial_Density_Map	Boolean	False	设置是否使用放射线密度贴图
<ring.radial_falloff>.curve_1	SubAnim		设置径向颜色中使用的颜色的权重。本属性包含环绕色彩衰减曲线, 用户不能创建或存取曲线上的点(除非该点有设置动画。这种情况下, 用户可以将点的位置作为属性.curve_1 的一个动画来存取。)

(续表)

属性名称	数据类型	默认值	说明
<ring.circular_falloff>.curve_1	SubAnim		设置环绕颜色中使用的颜色的权重。本属性包含环绕色彩衰减曲线，用户不能创建或存取曲线上的点（除非该点有设置动画。这种情况下，用户可以将点的位置作为属性.curve_1 的一个动画来存取。）
<ring.radial_size>.curve_1	SubAnim		确定围绕特定镜头效果的径向大小。本属性包含代表镜头效果的径向大小的曲线。用户不能创建或存取曲线上的点（除非该点有设置动画。这种情况下，用户可以将点的位置作为属性.curve_1 的一个动画来存取。）
<ring.radial_density>.curve_1	SubAnim		确定希望应用其他效果的位置和程度。本属性包含曲线代表应用于镜头效果的其他效果的密度。用户不能创建或存取曲线上的点（除非该点有设置动画。这种情况下，用户可以将点的位置作为属性.curve_1 的一个动画来存取。）

**注意**

方法 le.addRing <Lens\_Effects> 如果添加操作成功，返回 True；否则，返回 False。一旦效果被添加到镜头效果里，就可以作为镜头效果的一个属性来进行存取，其属性名称为.ring，如果同一类型的射线镜头效果被添加，第一个使用上面属性名，剩下的效果元素必须作为镜头效果的子动画进行存取，子动画的名称与属性名相同。

#### 14.9.6 Lens\_Effects—Star (星形镜头效果)

##### 构造函数

```
le.addStar <Lens_Effects>
```

##### 属性

属性名称	数据类型	默认值	说明
<star>.elementName <star>.name	String	“Star”	指定星形镜头效果的名称
<star>.on	Boolean	True	设置效果在场景渲染输出时是否被激活。可动画
<star>.size	Float	200.0	指定星形镜头效果的大小。可动画
<star>.intensity	Float	20.0	设置效果的亮度和不透明度。数值越大，效果越明亮，越不透明。可动画
<star>.width	Float	8.0	指定星形镜头效果的宽度。可动画
<star>.angle	Float	0.0	指定星形镜头效果的角度。其值可以为正值也可以为负值。可动画
<star>.taper	Float	0.5	指定星形镜头效果每个星角的导角量，较小的数值使星角顶端更为尖锐，较大的数值使星角顶端更为发散。可动画
<star>.sharp <star>.Sharpness	Float	9.5	指定星形镜头效果每个星角的锐化系数，取值范围为 0~10。较小的数值可以创建清晰、锐利、明亮的星角，较大的数值可以创建模糊、发光的星角。可动画

(续表)

属性名称	数据类型	默认值	说明
<star>.quantity	Integer	6	指定出现在星形镜头效果中所有的星形的星角数量，这些星角之间的间距是相等的。可动画
<star>.objectsHide <star>.Object_ide	Boolean	False	设置当效果处在场景对象背面时是否依然可以显示
<star>.occlusion	Float	50.0	确定镜头效果场景阻光度参数对特定效果的影响程度。可动画
<star>.squeeze	Boolean	False	指定挤压设置是否影响星形镜头效果
<star>.colorSource <star>.Source_Color	Float	100.0	指定手动星形镜头效果与源灯光色彩或源对象色彩的混合程度。如果为 0，仅使用源色彩；如果为 100，仅使用效果色彩。可动画
<star>.centerColor <star>.Center_Color	Color	(color 0 0 255)	可动画
<star>.edgeColor <star>.Edge_Color	Color	(color 255 255 255)	可动画
<star>.radialMtl <star>.Radial_Material	TextureMap	undefined	指定径向颜色的贴图
<star>.useRadialMtl <star>.Apply_Radial_Color_Map	Boolean	False	设置是否使用贴图确定径向颜色
<star>.radial_falloff	SubAnim		
<star>.radialMap <star>.Radial_Falloff_Map	TextureMap	undefined	指定径向衰减贴图
<star>.useRadialMap <star>.Apply_Radial_Falloff_Map	Boolean	False	指定是否使用径向衰减贴图
<star>.circularColorMix <star>.Circular_Color_Mix	Float	0.0	混合在径向颜色和环绕颜色中设置的颜色。如果设置为 0，将只使用径向颜色中设置的值；如果设置为 100，将只使用环绕颜色中设置的值。0~100 之间的任何值将在两个值之间混合。可动画
<star>.leftSectionColor <star>.Left_Section_Color	Color	(color 0 0 255)	指定第一个截面颜色值。可动画
<star>.centerSectionColor <star>.Center_Section_Color	Color	(color 255 255 255)	指定第二个截面颜色值。可动画
<star>.rightSectionColor <star>.Right_Section_Color	Color	(color 0 0 255)	指定第三个截面颜色值。可动画

(续表)

属性名称	数据类型	默认值	说明
<star>.circularMtl <star>.Circular_Material	TextureMap	undefined	使用材质定义环绕色彩的效果
<star>.useCircularMtl <star>.Apply_Circular_Color_Map	Boolean	False	指定是否使用材质来定义环绕色彩的效果
<star>.circular_falloff	SubAnim		定义环绕色彩的衰减方式
<star>.circularMap <star>.Circular_Falloff_Map	TextureMap	undefined	为环绕色彩的衰减指定贴图
<star>.useCircularMap <star>.Apply_Circular_Falloff_Map	Boolean	False	设置是否为环绕色彩的衰减指定贴图
<star>.radial_size	SubAnim		指定星形镜头效果的放射尺寸
<star>.radialSizeMap <star>.Radial_Size_Map	TextureMap	undefined	为星形镜头效果的放射尺寸指定贴图
<star>.useRadialSizeMap <star>.Apply_Radial_Size_Map	Boolean	False	设置是否为星形镜头效果的放射尺寸指定贴图
<star>.applyLights <star>.Apply_to_Lights	Boolean	True	是否将效果指定到源灯光对象上
<star>.applyImage <star>.Apply_to_Image	Boolean	True	是否将效果指定到渲染输出的图像上
<star>.applyImageCenters <star>.Apply_to_Image_Centers	Boolean	False	是否将效果指定到对象的中心还是对象的一部分
<star>.sourceObjectID <star>.Source_Object_ID	Boolean	False	设置是否将镜头效果应用于场景中具有相应 G 缓冲区(或对象) ID 的特定对象。可动画
<star>.objectID <star>.Object_ID	Integer	1	设置对象通道 ID。可动画
<star>.sourceEffectsID <star>.Source_Effects_ID	Boolean	False	设置是否将效果指定给具有同一材质效果 ID 的对象。可动画
<star>.effectsID <star>.Effects_ID	Integer	1	设置材质效果 ID。可动画
<star>.sourceUnclampedColor <star>.Source_Unclamped_Color	Boolean	False	设置是否使用超亮度颜色
<star>.unclampedColor <star>.Unclamped_Color	Float	1.0	指定最低像素颜色, 像素颜色必须大于该值, 才会进行星形镜头效果处理

(续表)

属性名称	数据类型	默认值	说明
<star>.unclampedColorInvert <star>.Source_Unclampede_Color_Invert	Boolean	False	设置是否反转最低像素颜色
<star>.sourceSurfaceNormal <star>.Source_Surface_Normal	Boolean	False	设置是否使用表面法线来控制像素是否被进行效果处理
<star>.surfaceNormal <star>.Surface_Normal	Float	0.0	对象表面法线与摄像机之间的角度值，大于指定值时，该表面才会被进行星形镜头效果处理。可动画
<star>.surfaceNormalInvert <star>.Source_Surface_Normal_Invert	Boolean	False	设置是否反转表面法线角度值
<star>.sourceWhole <star>.Source_Whole	Boolean	False	是否将星形镜头效果指定到全部场景
<star>.sourceAlpha <star>.Source_Alpha	Boolean	False	是将星形镜头效果指定到某一 Alpha 通道
<star>.alpha	Integer	0	指定星形镜头效果的 Alpha 通道，取值范围为 0~255。可动画
<star>.sourceZBuffer <star>.Source_Z_Buffer	Boolean	False	设置是否使用 Z 缓冲高值来控制星形镜头效果
<star>.zHi <star>.Z_Hi	Float	1000.0	Z-Buffer 用来存储对象与摄像机之间的距离，zHi 来指定对象与摄像机之间的最大距离，如果大于该值，就不进行星形镜头效果处理。可动画
<star>.zLo <star>.Z_Lo	Float	0.0	Z-Buffer 用来存储对象与摄像机之间的距离，zLo 来指定对象与摄像机之间的最小距离，如果小于该值，就不进行星形镜头效果处理。可动画
<star>.filterAll <star>.Filter_All	Boolean	True	是否为场景中所有对象指定星形镜头效果
<star>.filterEdge <star>.Filter_Edge	Boolean	False	是否为场景中所有对象的边界指定星形镜头效果
<star>.filterPerimeterAlpha <star>.Filter_Perimeter_Alpha	Boolean	False	是否仅为场景中对象 Alpha 通道的边缘指定星形镜头效果
<star>.filterPerimeter <star>.Filter_Perimeter	Boolean	False	是否仅为场景中对象的边缘指定星形镜头效果
<star>.filterBrightness <star>.Filter_Brightness	Boolean	False	是否基于场景对象的亮度指定星形镜头效果

(续表)

属性名称	数据类型	默认值	说明
<star>.brightness <star>.Bright	Integer	0	设置基于亮度指定星形镜头效果的亮度值，只有亮度值高于本值的源对象才会被进行星形镜头效果处理。可动画
<star>.brightnessInvert <star>.Filter_Brightness_Invert	Boolean	False	是否将属性.brightness 的数值反转
<star>.filterHue <star>.Filter_Hue	Boolean	False	是否基于场景对象的色调指定星形镜头效果
<star>.hueRange <star>.Hue_Range	Integer	0	设置基于色相指定星形镜头效果的色调值，只有色相值高于本值的源对象才会被进行星形镜头效果处理。可动画
<star>.hue	Color	(color 255 0 0)	可动画
<star>.applyNoise <star>.Apply_Additional_Effects	Boolean	False	设置是否使用噪波效果
<star>.noiseMap <star>.Noise_Map	TextureMap	undefined	指定噪波贴图
<star>.radial_density	SubAnim		设置放射线密度
<star>.radialDensity <star>.Radial_Density_Map	TextureMap	undefined	为放射线密度指定贴图
<star>.useRadialDensity <star>.Apply_Radial_Density_Map	Boolean	False	设置是否使用放射线密度贴图
<star>.radial_falloff>.curve_1	SubAnim		设置径向颜色中使用的颜色的权重。本属性包含环绕色彩衰减曲线，用户不能创建或存取曲线上的点(除非该点有设置动画。这种情况下，用户可以将点的位置作为属性.curve_1 的一个动画来存取。)
<star>.circular_falloff>.curve_1	SubAnim		设置环绕颜色中使用的颜色的权重。本属性包含环绕色彩衰减曲线，用户不能创建或存取曲线上的点(除非该点有设置动画。这种情况下，用户可以将点的位置作为属性.curve_1 的一个动画来存取。)
<star>.radial_size>.curve_1	SubAnim		确定围绕特定镜头效果的径向大小。本属性包含代表镜头效果的径向大小的曲线。用户不能创建或存取曲线上的点(除非该点有设置动画。这种情况下，用户可以将点的位置作为属性.curve_1 的一个动画来存取。)
<star>.radial_density>.curve_1	SubAnim		确定希望应用其他效果的位置和程度。本属性包含曲线代表应用于镜头效果的其他效果的密度。用户不能创建或存取曲线上的点(除非该点有设置动画。这种情况下，用户可以将点的位置作为属性.curve_1 的一个动画来存取。)

**注意**

方法 le.addStar <Lens\_Effects> 如果添加操作成功，返回 True；否则，返回 False。一旦效果被添加到镜头效果里，就可以作为镜头效果的一个属性来进行存取，其属性名称为.star，如果同一类型的射线镜头效果被添加，第一个效果元素使用上面的属性名，剩下的效果元素必须作为镜头效果的子动画进行存取，子动画的名称与属性名相同。

### 14.9.7 Lens\_Effects-Streak (条纹镜头效果)

#### 构造函数

```
le.addStreak <Lens_Effects>
```

#### 属性

属性名称	数据类型	默认值	说明
<streak>.elementName <streak>.name	String	“Streak”	指定条纹镜头效果的名称
<streak>.on	Boolean	True	设置效果在场景渲染输出时是否被激活
<streak>.size	Float	50.0	指定镜头条纹的尺寸。可动画
<streak>.intensity	Float	30.0	设置效果的亮度和不透明度。数值越大，效果越明亮，越不透明。可动画
<streak>.width	Float	2.0	指定镜头条纹的宽度。可动画
<streak>.angle	Float	0.0	指定镜头条纹的角度。可动画
<streak>.taper	Float	0.5	指定镜头条纹的倒角量。可动画
<streak>.sharp <streak>.Sharpness	Float	9.8	指定镜头条纹的锐化系数。可动画
<streak>.objectsHide <streak>.Object_Hide	Boolean	False	设置当效果处在场景对象背面时是否依然可以显示
<streak>.occlusion	Float	100.0	确定镜头效果场景阻光度参数对特定效果的影响程度。可动画
<streak>.squeeze	Boolean	False	指定挤压设置是否影响条纹镜头效果。
<streak>.colorSource <streak>.Source_Color	Float	100.0	指定条纹镜头效果与源灯光色彩或源对象色彩的混合程度，如果为0，仅使用源色彩，如果为100，仅使用效果色彩。可动画
<streak>.centerColor <streak>.Center_Color	Color	(color 0 0 255)	可动画
<streak>.edgeColor <streak>.Edge_Color	Color	(color 255 255 255)	可动画
<streak>.radialMtl <streak>.Radial_Material	TextureMap	undefined	指定径向颜色的贴图
<streak>.useRadialMtl <streak>.Apply_Radial_Color_Map	Boolean	False	设置是否使用贴图确定径向颜色
<streak>.radial_falloff	SubAnim		指定径向衰减方式
<streak>.radialMap <streak>.Radial_Falloff_Map	TextureMap	undefined	指定径向衰减贴图

(续表)

属性名称	数据类型	默认值	说明
<streak>.useRadialMap <streak>.Apply_Radial_Falloff_Map	Boolean	False	指定是否使用径向衰减贴图
<streak>.circularColorMix <streak>.Circular_Color_Mix	Float	50.0	指定放射色彩与环绕色彩混合的程度,如果为0,仅使用放射色彩,如果为100,仅使用环绕色彩。可动画
<streak>.quadrant1Color <streak>.Left_Section_Color	Color	(color 0 0 0)	指定用于混合径向颜色和环绕颜色的第一个颜色值。可动画
<streak>.quadrant2Color <streak>.Center_Section_Color	Color	(color 255 255 255)	指定用于混合径向颜色和环绕颜色的第二个颜色值。可动画
<streak>.quadrant3Color <streak>.Right_Section_Color	Color	(color 0 0 0)	指定用于混合径向颜色和环绕颜色的第三个颜色值。可动画
<streak>.circularMtl <streak>.Circular_Material	TextureMap	undefined	指定用于混合径向颜色和环绕颜色的第四个颜色值。可动画
<streak>.useCircularMtl <streak>.Apply_Circular_Color_Map	Boolean	False	指定是否使用材质来定义环绕色彩的效果
<streak>.circular_falloff	SubAnim		定义环绕色彩的衰减方式
<streak>.circularMap <streak>.Circular_Falloff_Map	TextureMap	undefined	指定定义环绕色彩效果的贴图
<streak>.useCircularMap <streak>.Apply_Circular_Falloff_Map	Boolean	False	指定是否使用贴图来定义环绕色彩的效果
<streak>.radial_size	SubAnim		指定条纹的放射尺寸
<streak>.radialSizeMap <streak>.Radial_Size_Map	TextureMap	undefined	为条纹的放射尺寸指定贴图
<streak>.useRadialSizeMap <streak>.Apply_Radial_Size_Map	Boolean	False	设置条纹是否使用放射尺寸贴图
<streak>.applyLights <streak>.Apply_to_Lights	Boolean	True	是否将效果指定到源灯光对象上
<streak>.applyImage <streak>.Apply_to_Image	Boolean	True	是否将效果指定到渲染输出的图像上
<streak>.applyImageCenters <streak>.Apply_to_Image_Centers	Boolean	False	是否将效果指定到对象的中心还是对象的一部分

(续表)

属性名称	数据类型	默认值	说明
<streak>.sourceObjectID <streak>.Source_Object_ID	Boolean	False	设置是否将镜头效果应用于场景中具有相应 G 缓冲区(或对象)ID 的特定对象。可动画
<streak>.objectID <streak>.Object_ID	Integer	1	设置对象通道 ID。可动画
<streak>.sourceEffectsID <streak>.Source_Effects_ID	Boolean	False	设置是否将效果指定给具有同一材质效果 ID 的对象。可动画
<streak>.effectsID <streak>.Effects_ID	Integer	1	设置材质效果 ID。可动画
<streak>.sourceUnclampedColor <streak>.Source_Unclamped_Color	Boolean	False	设置是否使用超亮度颜色
<streak>.unclampedColor <streak>.Unclamped_Color	Float	1.0	指定最低像素颜色, 像素颜色必须大于该值, 才会进行条纹镜头效果处理。可动画
<streak>.unclampedColorInvert <streak>.Source_Unclamped_Color_Invert	Boolean	False	设置是否反转最低像素颜色
<streak>.sourceSurfaceNormal <streak>.Source_Surface_Normal	Boolean	False	设置是否使用表面法线来控制像素是否被进行效果处理
<streak>.surfaceNormal <streak>.Surface_Normal	Float	0.0	对象表面法线与摄像机之间的角度值, 大于指定值时, 该表面才会被进行条纹处理。可动画
<streak>.surfaceNormalInvert <streak>.Source_Surface_Normal_Invert	Boolean	False	设置是否反转表面法线角度值
<streak>.sourceWhole <streak>.Source_Whole	Boolean	False	是否将条纹镜头效果指定到全部场景
<streak>.sourceAlpha <streak>.Source_Alpha	Boolean	False	是将条纹镜头效果指定到某一 Alpha 通道
<streak>.alpha	Integer	0	指定条纹镜头效果的 Alpha 通道, 取值范围为 0~255。可动画
<streak>.sourceZBuffer <streak>.Source_Z_Buffer	Boolean	False	设置是否使用 Z 缓冲高值来控制条纹镜头效果
<streak>.zHi <streak>.Z_Hi	Float	1000.0	Z-Buffer 用来存储对象与摄像机之间的距离, zHi 来指定对象与摄像机之间的最大距离, 如果大于该值, 就不进行条纹镜头效果处理。可动画

(续表)

属性名称	数据类型	默认值	说明
<streak>.zLo <streak>.Z_Lo	Float	0.0	Z-Buffer 用来存储对象与摄像机之间的距离, zLo 来指定对象与摄像机之间的最小距离, 如果小于该值, 就不进行条纹镜头效果处理。可动画
<streak>.filterAll <streak>.Filter_All	Boolean	True	是否为场景中所有对象指定条纹镜头效果
<streak>.filterEdge <streak>.Filter_Edge	Boolean	False	是否为场景中所有对象的边界指定条纹镜头效果。
<streak>.filterPerimeterAlpha <streak>.Filter_Perimeter_Alpha	Boolean	False	是否仅为场景中对象 Alpha 通道的边缘指定条纹镜头效果
<streak>.filterPerimeter <streak>.Filter_Perimeter	Boolean	False	是否仅为场景中对象的边缘指定条纹镜头效果
<streak>.filterBrightness	Boolean	False	是否基于场景对象的亮度指定条纹镜头效果
<streak>.brightness <streak>.Bright	Integer	0	设置基于亮度指定条纹镜头效果的亮度值, 只有亮度值高于本值的源对象才会被进行条纹处理。可动画
<streak>.brightnessInvert <streak>.Filter_Brightness_Invert	Boolean	False	是否将.brightness 属性的数值反转
<streak>.filterHue <streak>.Filter_Hue	Boolean	False	是否基于场景对象的色调指定条纹镜头效果
<streak>.hueRange <streak>.Hue_Range	Integer	0	设置基于色相指定条纹镜头效果的色调值, 只有色相值高于本值的源对象才会被进行条纹处理。可动画
<streak>.hue	Color	(color 255 0 0)	可动画
<streak>.applyNoise <streak>.Apply_Additional_Effects	Boolean	False	设置是否使用噪波效果
<streak>.noiseMap <streak>.Noise_Map	TextureMap	undefined	指定噪波贴图
<streak>.radial_density	SubAnim		设置放射线密度
<streak>.radialDensity <streak>.Radial_Density_Map	TextureMap	undefined	为放射线密度指定贴图
<streak>.useRadialDensity <streak>.Apply_Radial_Density_Map	Boolean	False	设置是否使用放射线密度贴图

(续表)

属性名称	数据类型	默认值	说明
<streak.radial_falloff>.curve_1	SubAnim		设置径向颜色中使用的颜色的权重。本属性包含环绕色彩衰减曲线，用户不能创建或存取曲线上的点（除非该点有设置动画。这种情况下，用户可以将点的位置作为属性.curve_1 的一个动画来存取。）
<streak.circular_falloff>.curve_1	SubAnim		设置环绕颜色中使用的颜色的权重。本属性包含环绕色彩衰减曲线，用户不能创建或存取曲线上的点（除非该点有设置动画。这种情况下，用户可以将点的位置作为属性.curve_1 的一个动画来存取。）
<streak.radial_size>.curve_1	SubAnim		确定围绕特定镜头效果的径向大小。本属性包含代表镜头效果的径向大小的曲线。用户不能创建或存取曲线上的点（除非该点有设置动画。这种情况下，用户可以将点的位置作为属性.curve_1 的一个动画来存取。）
<streak.radial_density>.curve_1	SubAnim		确定希望应用其他效果的位置和程度。本属性包含曲线代表应用于镜头效果的其他效果的密度。用户不能创建或存取曲线上的点（除非该点有设置动画。这种情况下，用户可以将点的位置作为属性.curve_1 的一个动画来存取。）

**注意**

方法 le.addStreak <Lens\_Effects> 如果添加操作成功，返回 True；否则，返回 False。一旦效果被添加到镜头效果里，就可以作为镜头效果的一个属性来进行存取，其属性名称为.streak，如果同一类型的射线镜头效果被添加，第一个效果元素使用上面的属性名，剩下的效果元素必须作为镜头效果的子动画进行存取，子动画的名称与属性名相同。

## 14.10 Motion Blur: RenderEffect (运动模糊渲染效果)

### 构造函数

```
Motion_Blr ...
imageMotionBlur ...
```

### 属性

属性名称	数据类型	默认值	说明
<Motion_Blr>.duration	Float	1.0	指定虚拟快门打开的时间。设置为 1.0 时，虚拟快门在一帧和下一帧之间的整个持续时间保持打开。值越大，运动模糊效果越明显。可动画
<Motion_Blr>.transparency	Boolean	True	当设置为 On 时，运动模糊效果会应用于透明对象后面的对象。否则，透明对象后面的对象不会应用运动模糊效果。禁用此开关可以加快渲染速度

# 第15章 创建脚本工具程序 Utility

## 15.1 关于定制脚本工具程序 Utility

在 MAXScript 里有一系列的类和函数可以让用户来定制自己的 Rollout（卷展栏），并将它们插入到 3ds max 用户界面里去。这些定制 Rollout 主要为用户编写的脚本程序提供一个鼠标点击的界面，这样编程者以外的人使用这些脚本会更方便、更直观。

在 3ds max 用户界面里，有两种方法用来插入用户自己定制的 Rollout：

方法一：在 Utilities 命令面板下。用户先按语法要求编制好自己的 Rollout，然后将脚本加载到系统，如果脚本程序加载成功，定制 Rollout 的标题会显示在 MAXScript 卷展栏底部的 Utility 下拉列表里。如果用户在该下拉列表里选中一个 Utility，与之对应的定制 Rollout 会展开在 MAXScript 卷展栏的下面，其使用方法与 3ds max 系统里的 Plug-in Utility 一样。

方法二：还可以用函数 newRolloutFloater() 来创建无模式的浮动窗口。这种窗口显示在 Windows 桌面上，类似于 3ds max 系统的 Material Editor 和 Selection 浮动窗口。用户可以用鼠标在这些窗口和 3ds max 主界面之间随意切换。

一般来说，建议读者尽量使用第一种方法，这样做有两个好处：少占用屏幕空间，且和 3ds max 用户界面的使用方法一致。但如果用户在使用脚本工具时需要在不同命令面板之间来回切换，使用第二种方法会比较方便。在某些情况下，建议编程者考虑在第一种方法构造的卷展栏末尾加上一个 Float Me 的按钮（读者可参考 Color Clipboard Utility 窗口），这样用户可以选择定制卷展栏的显示方式：在 Utilities 命令面板显示还是采用浮动窗口来显示。

## 15.2 定义脚本工具程序 Utility

在 MAXScript 中用 Utility 子句来定义脚本工具程序 Utility，其语法为：

```
Utility <var_name> <description_String> [ rolledUp:<Boolean> ] \ 
    [ silentErrors:<Boolean> ] (<Utility_body>)
```

参数说明：

- ◆ <var\_name> 为脚本工具程序的名字，实际上是一个代表定制 Utility 的全局变量；
- ◆ <description\_String> 是出现在 MAXScript 卷展栏 Utilities 列表里的描述字符串；
- ◆ rolledUp: 为可选参数，指定 Utility Rollout 是否在初始化时被展开，默认值为 False；

- ◆ silentErrors: 为可选参数, 指定在执行 Utility 时, 若发生运行错误, 是否显示错误信息。默认值为 False, 表示当发生运行错误时会在 Listener 窗口或弹出的错误对话框中显示错误信息, 同时程序会中断运行; 如果设为 True, 发生运行错误时, 错误信息不被显示, 程序继续运行。
- ◆ <Utility\_body> 为括在一对括号里的子句序列, 用来定义在该 Utility 里出现的界面项, 以及函数和事件处理程序。

下面是一个简单的例子, 图 15-1 为该脚本创建的 Utility, 其中 Close 按钮是 MAXScript 自动为定制脚本工具 Utility 创建用来关闭 Rollout 的。

```
-- 定义 Utility 名称和描述字符串
Utility spread "Spread objects"
(local last_amt = 0      -- 定义并初始化局部变量
--
    Checkbox x "Spread in x"           -- 创建 3 个 Checkbox 控件
    Checkbox y "Spread in y"
    Checkbox z "Spread in z"
-- 创建 1 个 Spinner 控件
    Spinner spread "Spread amount: " range:[-1000,1000,0]
--
    on spread changed amt do          -- 定义 Spinner 控件的事件处理程序
(delta = amt - last_amt
    for obj in selection do
(
    p = obj.pos + normalize(obj.pos - selection.center)* delta
    -- 如果 Checkbox 控件 x 被选择, 处理对象 x 坐标
if x.checked then obj.pos.x = p.x
-- 如果 Checkbox 控件 y 被选择, 处理对象 y 坐标
if y.checked then obj.pos.y = p.y
-- 如果 Checkbox 控件 z 被选择, 处理对象 z 坐标
if z.checked then obj.pos.z = p.z
)
last_amt = amt
)-- Spinner 控件的事件处理程序结束
)-- Utility 定义结束
```

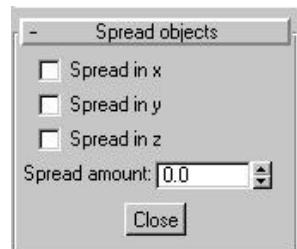


图 15-1 脚本创建的 Utility (一)

**注意** 如果存在相同的 Utility 名称和描述字符串, 旧的 Utility 定义的 Rollout 会被新的覆盖, 即使旧的 Rollout 当前正在 Utility 里被打开。

### 15.3 Utility 子句

脚本工具程序定义中的 <Utility\_body> 由一系列 Utility 子句组成, 其语法定义如下:

<Utility\_body> ::= { <Utility\_clause> }+

其中：`<Utility_clause> ::= <Rollout_clause> | <nested_Rollout>`

一个`<Utility_clause>`可包含多个`<Rollout_clause>`或`<nested_Rollout>`，用来在脚本工具程序里构造多个 Rollout 或 Rollout 浮动窗口，其中`<Rollout_clause>`称为 Rollout 子句，`<nested_Rollout>`称为嵌套 Rollout 子句，可以包含多个`<Rollout_clause>`。`<Rollout_clause>`的语法定义为：

```
Rollout <var_name> <description_String> \
    [ rolledUp:<Boolean> ] [ silentErrors:<Boolean> ] \
    (<Rollout_body> )
```

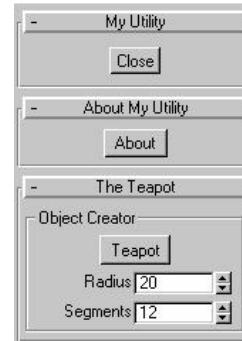
其中：`<Rollout_body> ::= {<Utility_clause>}+`

由上可以看出 Rollout 的定义与 Utility 的定义是一样的，但 Rollout 子句的`<description_String>`不会显示在 MAXScript 卷展栏的 Utility 下拉列表中。实际上，Utility 可以看作是 Rollout 子句的一个特例。

请读者阅读下面的例子。本例中，第 1 行为 Utility 的开始，`<Utility_body>`为第 2 行到脚本结束，其中包含了 3 个 Rollout\_clause 和 2 个 nested\_Rollout。

```
Utility MyUtil "My Utility"
(
    local pot --局部变量声明，第 1 个 Rollout_clause
    --
    Rollout bout "About My Utility"
    --第 1 个 nested_Rollout 开始
    (button aboutMU "About" width:45 height:20
        on aboutMU pressed do
            messagebox "My First Utility\nby ME\nVersion .1" \
                title: "About My Utility"
        )--第 1 个 nested_Rollout 定义完成
    --
    Rollout creator "The Teapot" --第 2 个 nested_Rollout 开始
    (group "Object Creator"
        (button tea "Teapot"
            Spinner rad "Radius" range:[10,50,20] type:#Integer
            Spinner seg "Segments" range:[4,32,12] type:#Integer scale:1
        )
        --
        on tea pressed do
            (pot=teapot radius:rad.value
                pot.name="TestPot"
                pot.segs=seg.value
            )--on tea pressed 事件定义完成
        --
        on rad changed value do
            pot.radius=value
        --
        on seg changed value do
            pot.segs=seg.value
        --
    )
```

```
-- 第 2 个 nested_Rollout 定义完成
--
on MyUtil open do -- 第 2 个 Rollout_clause 开始
(addRollout bout
    addRollout creator
)-- 第 2 个 Rollout_clause 定义完成
--
on MyUtil close do -- 第 3 个 Rollout_clause
开始
(removeRollout bout
    removeRollout creator
)-- 第 3 个 Rollout_clause 定义完成
--
)--Utility MyUtil 定义完成
```



由上面的脚本程序创建的 Utility 如图 15-2 所示。 图 15-2 由脚本创建的 Utility (二)

当定义 Utility 或 Rollout 的脚本被执行后，系统会为它们创建一个值。一般而言，Utility 值的生存期一直到用户退出 3ds max 或者到一个同名的 Utility 定义被执行。

## 15.4 在一个脚本工具 Utility 里定义多个卷展栏

一个 Utility 里可包含几个卷展栏，这一点在开发较大的脚本工具程序时是很有用的。由 Utility 定义本身创建一个主 Rollout，其他包含在 Utility 定义体之内的 Rollout 子句称之为嵌套 Rollout。这些嵌套 Rollout 可以随时被加入主 Rollout，也可以随时被移走。下面是一个定义嵌套 Rollout 的例子：

```
Utility foo "name"
(
    local ...
    Spinner ...
    ...
    Rollout baz "name"
    (
        local ...
        Checkbox ..
        on ...
    )
    ...
)
```

一个嵌套 Rollout 可以包含任何 Utility 子句定义可以包含的内容，但一个嵌套 Rollout 不能包含更深层次的嵌套 Rollout。嵌套 Rollout 不会随着主 Rollout 的打开和关闭而自动打开、关闭，用户必须用 open 和 close 事件处理器或其他处理器、函数来显式地打开或关闭它们。有两个函数分别用来打开、关闭嵌套 Rollout：

`addRollout <Rollout> [ rolledUp:<Boolean> ]`

```
removeRollout <Rollout>
```

可选参数 rolledUp:指定当嵌套 Rollout 被加入主 Rollout 时是否被展开，默认值为 False。嵌套 Rollout 按 addRollout()函数调用的顺序排列，这一点用户应注意以确保它们按要求的顺序排列。

当主 Rollout 被关闭时，嵌套 Rollout 应确保被同时关闭。通常我们可以在主 Rollout 的 close 事件处理程序里调用 removeRollout()函数。同理我们可以在主 Rollout 的 open 事件处理程序里调用 addRollout()函数，如下例：

```
on foo open do
(
    ...
    addRollout panel_1
    addRollout panel_2 rolledUp:True
    ...
)
on foo close do
(
    ...
    removeRollout panel_1
    removeRollout panel_2
    ...
)
```

如果一个<Rollout>已被关闭，再对它调用 removeRollout()函数会返回 OK，但系统什么也不用做。关闭整个 Utility 可以关闭任何未关闭的脚本 Rollout。因为嵌套 Rollout 是包含在 Utility 定义里面的，所有 Utility 里的局部变量、函数、界面控件对嵌套 Rollout 都是可见的，其作用域遵循 MAXScript 里的标准嵌套作用域，也即用户可以在嵌套 Rollout 里存取 Utility 的局部变量、用户控件和函数。相反，嵌套 Rollout 里的用户控件和局部变量可以作为该 Rollout 的属性，这样在主 Rollout 或其他嵌套 Rollout 里只能通过该 Rollout 来存取它们。如下例中 Spinner 控件 bar 的 on change 事件处理程序可以调用嵌套 Rollout baz 里 Checkbox 类控件 baz\_enable 的状态属性：

```
Utility foo "name"
(
    local ...
    Spinner bar ...
    ...
    Rollout baz "name"
    (
        local ...
        Checkbox baz_enable ...
        on ...
    )
    ...
    on bar changed val do
        if baz.baz_enable.checked == True then ...
    )
)
```

## 15.5 Rollout 子句

Rollout 子句为 15.2 节中<Utility\_body>和 15.3 节中<Rollout\_body>的基本组成部分，它定义了一个卷展栏的构成。一个 Rollout 子句可以包含下面四种基本内容：

- ◆ 局部变量、全局变量声明和函数、结构定义。它们的作用域为 Rollout 子句内部，其生存期与由 Rollout 子句定义的 Rollout 类值相同（Rollout 类值将一直存在，直到用户将变量名赋新值为止）。局部变量在存储拾取对象和对象数组的数据方面特别有用。
- ◆ 鼠标工具。用来响应鼠标在 3ds max 视窗里的动作，如 click 等。具体说明详见第 18 章“脚本鼠标工具”。
- ◆ 用户界面控件。用来显示在 Rollout 上的元件，如 Button、Spinner 和 Listbox。
- ◆ 控件事件处理程序。与某一特定控件相关的特殊函数，用来指定当用户与控件进行交互（如按下一个按钮或调整一个 Spinner 控件的数值）时，编程者希望作出的反应。

一般而言，<Rollout\_clause>的语法定义如下：

```
<rollout_clause> ::= <local_variable_decl> |
    <global_variable_decl> | \
    <local_function_decl> | \
    <local_struct_decl> | \
    <mousetool> | \
    <user_interface_item> | \
    <item_group> | \
    <event_handler>
```

下面各节详细说明以上各部分。

### 15.5.1 局部变量、全局变量声明和函数、结构定义

<local\_variable\_decl>（局部变量声明）

<global\_variable\_decl>（全局变量声明）

<local\_function\_decl>（局部函数定义）

<local\_struct\_decl>（结构定义）

上述四者在 MAXScript 里的语法分别为：

- ◆ 局部变量声明

```
<local_variable_decl> ::= local <decl> { , <decl> }
```

<decl> ::= <name> [ = <expr> ] （可选的初始值）

◆ 全局变量声明

```
<global_variable_decl> ::= global <decl> { , <decl> }
<decl> ::= <name> [ = <expr> ] (可选的初始值)
```

◆ 局部函数定义

```
<local_function_decl> ::= [ mapped ](function | fn) <name> { <argument> } = <expr>
```

◆ 结构定义

```
<local_struct_decl> ::= struct <name> ( <member> { , <member> })
<member> ::= ( <name> [ = <expr> ] | <local_function_decl> )
```

例如：

```
local numSelected
global foo
local numSelected = 0
fn onlyOneSelected = (selection.count == 1)
struct parents (mother="", father="")
```

在这里再次提醒读者，在编程中应对用到的局部变量和全局变量进行显式地声明，隐式的变量声明为用户在 Listener 窗口里工作或短小的脚本程序提供了一些方便，但在一些较大的脚本程序里，显式的变量声明可以减少程序的错误，并增加程序的可读性。

### 15.5.2 用户界面控件<user\_interface\_item>

<user\_interface\_item>在语法定义中定义一个单独显示在 Rollout 上的用户界面控件，如：Button、Checkbox、Spinner 等，详细介绍参见 15.10 节。

### 15.5.3 用户界面控件组<item\_group>

一个<item\_group>用来在 Rollout 里把一系列的用户界面控件组织到一个带标签的框里面，这样用户可以把 Rollout 里的控件按某一归类原则分成几组，其语法为：

```
group <group_label_String>({ <user_interface_item> })
```

下面的例子说明了 group 的用法：

```
Utility Infinity "Game Utilities"
(
    group "Lighting"
        label label1d "Number of Day lights:" across:2 offset:[10,0]
        label label2d "0" offset:[10,0]
        label label1n "Number of Night lights:" across:2 offset:[13,0]
        label label2n "0" offset:[10,0]
        label label3
        Radiobuttons WhichOn "Active Lights:" labels:#( "Day", "Night" )
    )
    group "Scene Data Dump"
    ( Button scenedump "Dump Scene Data" )
```

```

group "Exclusions/Inclusions"
( Button DispExcl "Unhide Exclusions&Inclusions" )
group "Camera Mattes"
(radiobuttons CamMatte labels:#( "None", "1", "2", "3", "4")columns:3)
    Button resetb "Reset"
)--Utility Infinity 定义结束

```

由上面的例子生成的 Utility 面板如图 15-3 所示。

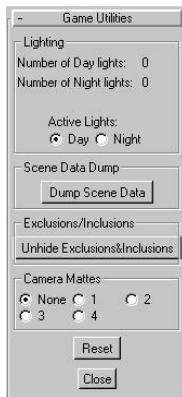


图 15-3 由脚本创建的 Utility (三)

#### 15.5.4 控件事件处理程序<event\_handler>

语法中<event\_handler>是一个特殊的函数定义，定义了当一个控件执行了某一动作后编程者希望作出的反应，仅在 Utility 或 Rollout 子句内部可用。如用户按下某一按钮、调整了一个 Spinner 的值、打开或关闭 Utility 或 Rollout、调整或移动了 Rollout 浮动窗口，这时相应的事件处理程序就会被调用。事件处理程序的语法为：

```
on <item_name> <event_name> [ <argument> ] do <expr>
```

参数说明：

- ◆ <item\_name> 指定与处理器相连的控件名称；
- ◆ <event\_name> 指定了要处理的事件类型；
- ◆ <argument> 为可选参数，用来传递变量值给处理器。

用户可指定的事件取决于控件类型，下面是 MAXScript 里所有的事件名称：

- ◆ pressed
- ◆ changed
- ◆ picked
- ◆ entered
- ◆ selected
- ◆ resized
- ◆ moved
- ◆ open

### ◆ close

哪些控件适用于哪些类型的事件详见 15.10 节。

例如：在一个 Rollout 里有一个 Checkbox 类型的控件，名称为 foo，并且定义了一个 on foo changed 的事件处理程序函数，可以用下面方法调用处理器：

```
foo.changed True --调用控件 foo 的 changed 事件处理程序，并将值 True 传递
```

如果在一个 Rollout 里有一个名为 apply 的 Button 类控件，并定义了 on apply pressed 事件处理程序，可以用下面方法调用事件处理程序：

```
apply.pressed() --调用 pressed 函数，没有变元
```

用户还可以将事件处理程序函数作为控件的一个子属性来进行存取，如：

```
ApplyPressed = apply.pressed
```

将事件处理程序函数 on apply pressed 的一个副本存储到变量 ApplyPressed 里。

## 15.6 Utility 和 Rollout 的属性、方法和事件处理程序

### 属性

下面的几个属性用来控制 Utility 和 Rollout 状态：

1. <Rollout\_or\_Utility>.open Boolean

返回 Utility 或 Rollout 当前是否展开或卷起。用户也可以给该属性赋给值 True 或 False，来展开或卷起指定 Rollout。

2. <Rollout\_or\_Utility>.scrollPos Integer

本属性严格来说，是一个 Rollout 浮动窗口或实用工具 Utility 里所有 Rollout 的一个属性，用来为那些没有办法全部显示在界面里的 Rollout 指定滚动条的位置。

用户可以编制一个 Utility 和 Rollout 状态的存储和恢复程序，这样在关闭 Rollout 时记录下每一 Rollout 的状态，在重新打开时再恢复它们的状态。

### 方法

下面的方法用来打开/关闭、添加/移走 Utility 里的 Rollout：

1. openUtility <Utility>

打开指定 Utility 里的主 Rollout。本方法相当于在 3ds max 的 MAXScript | Utilities 下拉列表里选择指定 Utility。

2. closeUtility <Utility>

关闭指定 Utility 里的主 Rollout。本方法相当于按下指定 Utility 的 Close 按钮。

3. addRollout <Rollout> [ rolledUp:<Boolean> ]

将指定嵌套 Rollout 添加到主 Rollout 里，参数 rolledUp: 用来指定是否在加入主 Rollout 后展开它，默认值为 False。

## 4. removeRollout &lt;Rollout&gt;

从主 Rollout 里移走指定的嵌套 Rollout。

### 事件处理程序

在为单个用户界面控件定义事件处理程序之外，用户还可以为第一次打开整个 Rollout 或关闭整个 Rollout 定义事件处理程序。这两个事件处理程序主要用来进行初始化设置或关闭 Rollout 时进行清理工作。

当移动一个浮动窗口或改变它的大小时，一个事件处理程序（分别为 moved 或 resized）会被浮动窗口的第一个 Rollout 调用。

事件处理程序的语法为：

## 1. on &lt;Rollout\_or\_Utility&gt; open do &lt;expr&gt;

当 Rollout 或 Utility 在打开时被调用。

## 2. on &lt;Rollout\_or\_Utility&gt; close do &lt;expr&gt;

当 Rollout 或 Utility 在关闭时被调用。

## 3. on &lt;Rollout\_or\_Utility&gt; oktclose do &lt;expr&gt;

当用户按下 Rollout 的 Close 按钮时被调用，这样编程者可以根据情况决定是否要关闭 Rollout。如果表达式的求值结果为 True，表示允许关闭 Rollout；否则，用户关闭 Rollout 的请求被忽略，Rollout 仍然保持打开。

## 4. on &lt;Rollout&gt; resized &lt;arg&gt; do &lt;expr&gt;

当一个浮动窗口被用户用鼠标或在脚本里改变大小时，本方法被浮动窗口的第一个 Rollout 调用。在 Resized 事件处理程序里面改变浮动窗口的大小不会导致再次调用 Resized 事件处理程序。变量<arg>为浮动窗口改变大小后的宽度和高度。如：

```
global my_rof_size
...
on my_Rollout resized size do
(my_rof_size = size --存储新窗口的大小
)
```

## 5. on &lt;Rollout&gt; moved &lt;arg&gt; do &lt;expr&gt;

当一个浮动窗口被用户用鼠标或在脚本里移动时，本方法被浮动窗口的第一个 Rollout 调用。变量<arg>为浮动窗口改变大小后的位置。

下面的例子为其操作建立了一个日志文件，在 Rollout 的 Open 和 Close 事件处理程序里会打开、关闭该日志文件。

```
Utility frab "Optimal Frabulator"
(
    local log
    Spinner frab_x ...
    button do_it "Enfrabulate now"
    ...
    on do_it pressed do
    (
        frabulate selection
```

```

        format "selection frabbed at %\n" localTime to:log
    )
on frab open do log = createFile "frabulator.log"
on frab close do close log
)

```

在下面的例子中，只有 Checkbox 类控件 OKToClose 被选择时，Utility 才能被关闭：

```

Utility ui_oktoclose "OKToClose Test"
(
    checkbutton      a2 "OK To Close"
    on ui_oktoclose oktoclose do a2.state
)

```

在下面的例子中，当浮动窗口被改变大小时，事件处理程序 on a resized 被调用，因为只有浮动窗口里的第一个 Rollout 的 Resize 事件处理程序被调用，所以事件处理程序 on b resized 永远都不会被调用。

```

Rollout a "Rollout A"
(button a1 "a1"
    on a resized val do(format "A: %\n" val)
)
--
Rollout b "Rollout B"
(button b1 "b1"
    on b resized val do(format "B: %\n" val)
)
--
rof=newRolloutFloater "test" 200 200
addRollout a rof
addRollout b rof
rof.size=[300,300]

```

## 15.7 Rollout 浮动窗口

MAXScript 允许用户创建无模式对话框，用户可以在浮动窗口里定义 Rollout 或 Utility，系统会将该窗口连同其包含的 Utility 和 Rollout 一起弹出到桌面，并可以通过鼠标按住窗口的底边垂直拖拉来改变窗口的大小。

有两个函数和一个特别的类用于无模式浮动窗口，说明如下：

1. newRolloutFloater <title\_String> <width\_Integer> \
 <height\_Integer> [<top\_Integer> <left\_Integer>]

功能：以指定标题、宽度和高度在指定位置创建并打开一个新的 Rollout 浮动窗口。如果调用该函数时未指定 top 和 left 坐标，窗口将位于屏幕正中。

返回值：返回一个 Rollout Floater 类值。

2. closeRolloutFloater <RolloutFloater>

功能：关闭指定 Rollout 浮动窗口。也可以点击窗口的 close 按钮来关闭窗口。窗口一旦关闭，该窗口不再可用。

用于向 Utility 面板添加和移去 Rollout 的函数同样可用于浮动窗口，其语法为：

```
addRollout <Rollout> [ <RolloutFloater> ] [ rolledUp:<Boolean> ]
```

```
removeRollout <Rollout> [ <RolloutFloater> ]
```

在上面的函数中，如果未指定可选参数<Rollout Floater>，将从 Utilities 面板加入或移走指定 Rollout。

Rollout 浮动窗口有如下几个属性：

1. <RolloutFloater>.size Point2

以像素为单位设置浮动窗口的尺寸。Point2 的 x 分量为窗口宽度，y 分量为高度，该属性为可读可写的。

2. <RolloutFloater>.pos Point2

设置浮动窗口当前在屏幕上的位置，以像素为单位。该属性也是可读可写的。

当一个 Rollout 浮动窗口被用户或脚本程序改变大小时，系统会为浮动窗口的第一个 Rollout 产生一个 resized 事件。同样，当一个 Rollout 浮动窗口被移动时，系统也会为该窗口的第一个 Rollout 产生一个 moved 事件。有关 resized、moved 事件处理程序的描述请读者参见 15.6 节。

例如：

```
Rollout grin "Grin Control"
(Slider happy "Happy" orient:#vertical across:5
    Slider sad "Sad" orient:#vertical
    Slider oo "OO" orient #vertical
    Slider ee "EE" orient #vertical
    Slider oh "OH" orient:#vertical
    on happy changed val do object.xform1.center = ...
    on sad changed val do object.xform2.gizmo.rotation = ...
    ...
)
gw = newRolloutFloater "Grinning" 300 220
addRollout grin gw
```

## 15.8 局部变量、函数、结构和用户界面控件的定义顺序

在 Utility 里，用户界面控件、函数和 Rollout 子句的定义顺序是非常重要的。只有那些预先已经被定义好的控件、Rollout 或函数才能被别的事件处理程序或函数引用。下面的排列顺序可以确保这一点：

- ◆ 局部变量声明
- ◆ 结构定义
- ◆ 用户界面控件

- ◆ 嵌套 Rollout
- ◆ 函数
- ◆ 事件处理程序

在某些情况下，我们可能需要在两个控件、Rollout 或函数之间进行交叉引用，这样就没有办法确定其定义顺序。MAXScript 提供了一个解决的办法：用户可以预先声明局部 Rollout 和函数，这样如果在定义之前有对其进行引用，可确保系统能找到正确的对象。在下面的例子中，两个 Rollout（ro1 和 ro2）可以互相交叉引用：

```
Utility foo "name"
(
    local ro1, ro2, ... --声明 Rollout 的局部变量
    ...
    Rollout ro1 "name"
    (local ...
        Checkbox enable ...
        on ... do
            ro2.enable.checked = False
    )
    ...
    Rollout ro2 "name1"
    (local ...
        Checkbox enable ...
        on ... do
            ro1.enable.checked = True
    )
    ...
)
```

将所有的局部变量在其作用域的最外围显式地声明为局部变量是一个良好的编程习惯，如果万一在另一个脚本里同名的变量被声明为全局变量，这样可以避免引起错误。尽管函数、结构定义和 Rollout 作用域总是限于其定义的 Utility 里，将其声明为 local 可确保当定义顺序被改变时，交叉引用可以找到正确的对象。下面是一个包含这些声明的例子：

```
Utility foo "Object Frabulator"
(
    --局部变量声明
    local target_obj
    --局部函数声明
    local prop_name
    --局部 Rollout 声明
    local setup
    fn prop_name obj name =
        for c in obj.children do c.name = name + "_" + c.name
    checkbutton setup_btn "Setup"
    edittext name_box "New name:"
    --使用一个 Checkbutton 控件来判断操作类型
    on setup_btn changed state do
        if state then addRollout setup else removeRollout setup
```

```

on foo close do
    removeRollout setup
Rollout setup "Setup fraber"           --嵌套 Rollout
(
    label hello
    pickbutton pick_tgt "Pick object"
    on pick_tgt picked obj do
        (target_obj = obj                  --存取 Utility 局部变量
         name_box.text = obj.name        --存取 Utility 控件
         prop_name obj obj.name        --调用 Utility 局部函数
        )--事件处理程序结束
    )--嵌套 Rollout 定义结束
)--)Utility 定义结束

```

## 15.9 从外部代码里存取 Utility 内部局部变量和控件

在 Utility 或 Rollout 里定义的局部变量可以作为 Utility 或 Rollout 的属性在外部脚本里来进行存取，例如：

```

Utility foo "Object Frabulator"
(local var1, var2, ...
...
Checkbox enable "Enable frabber"
...
Rollout setup "Setup frabber"
(local var3
    button doit "Execute"
...
    on doit pressed do ...
)
function frab a b = ...
...
on enable changed state do ...
...
)

```

上例定义了一个 Utility 类对象，一个名称为 foo 的全局变量指向这个对象，此后用户可以在 Listener 窗口或别的程序里对 foo 的成员以 Utility 对象属性的形式进行存取运算，而用户界面的所有事件处理程序也可以作为控件的子属性进行存取。如：

```

print foo.var1           --获取 foo 的局部变量 var1
if foo.enable then ...   --获取 foo 的 Checkbox 控件的属性.enable
foo.enable = False       --并给它赋值
foo.enable.changed False --调用事件处理程序
foo.frab $box01 $box02   --调用 foo 的局部函数 frab
foo.setup.var3=10         --为 foo 嵌套 Rollout setup 的局部变量 var3 赋值
--调用嵌套 Rollout setup 里 doit 按钮的事件处理程序
foo.setup.doit.pressed()

```

Utility 和 Rollout 里的局部变量、函数和结构在 Utility 和 Rollout 定义之后马上会被初始化，而不用等到 Utility 或 Rollout 首次显示到屏幕上时再进行初始化。这样当 Utility 和 Rollout 一旦被定义好，就可以在别的程序里调用它们里面的函数并对它们里面的局部变量进行赋值。

## 15.10 Rollout 用户界面控件

在 MAXScript 里，共有 26 种用户界面控件供用户选择来构造 Rollout，这些控件有下面相同的语法：

```
<item_type> <name> [ <label_String> ] [ <parameters> ]
```

其中：

- ◆ <item\_type>为控件类型，详见后面说明；
- ◆ <name>为控件名，实际上为一个作用域限于包含控件的 Rollout 内的局部变量；
- ◆ <label\_String>是可选参数，用来作为控件的标题、标签或文本内容（取决于控件类型）；
- ◆ <parameters>是可选参数，为一系列关键字型的可选参数，用来定义控件的布局。详见后面关于各控件的介绍。

默认情况下 MAXScript 按照定义顺序将控件放在 Rollout 里，从上向下排列，用户也可以用一些特别的布局参数取代这样的默认布局方式。对控件的布局进行调整，详见 15.10.2 节。

### 15.10.1 控件通用属性

所有定义好的用户界面控件都有一个作用域限于 Rollout 的局部变量，并将一个指向该控件的值赋给该变量。这个值包含了所有与控件有关的状态信息，如：Checkbox 类型控件的 checked 状态或 Spinner 类型控件的当前值等。这些信息作为控件值的属性供用户存取。我们可以用标准的 MAXScript 属性存取办法来读取、设置这些信息，如：

```
frab_x.enabled = True
foo.text = "Don't do it"
first_item = baz.items[1]
```

除 caption 以外的所有用户控件的通用属性都可以在构造控件的同时作为参数来进行指定。如：

```
button foo "Press Me!" enabled:False
```

下面是所有界面控件的通用属性：

1. <ui\_item>.caption String

该属性的意义随不同控件类型而不同。如果控件有一个.caption 属性，这个属性包含了标题字符串。对那些 Button 类控件而言，.caption 属性就是按钮上面的文字。.caption 属性

的默认值为控件定义时指定的<label\_String>参数值。

### 2. <ui\_item>.text String

除 Edittext 和 Combobox 类型控件之外，其他控件的.text 属性是.caption 属性的别名。

对 Edittext 和 Combobox 类型的控件，.text 属性为编辑框里的所有文本，Edittext 控件的.text 属性的默认值为一个空字符串（“”），而 Combobox 控件.text 属性的默认值为该控件中被选择的文本。

在 3ds max 8 里，Rollout 控件的 .text 属性真正等同于.caption 属性，所以不用单独定义.text 属性。给 Label 类控件的.text 属性赋值同时也会给.caption 属性赋值。而在以前的版本里，给 Label 类控件的.text 属性赋值不会引起 UI 界面刷新，而仅刷新控件内部的.caption 属性值。

### 3. <ui\_item>.enabled Boolean 默认值: True

设置当 Rollout 被第一次打开时，控件是否可用来与用户交互。不可用的控件在界面上呈灰色。所有控件的.enabled 属性默认值都为 True，我们一般通过将控件的.enabled 属性设为 False 来在 Rollout 初始化时禁用一些控件。

## 15.10.2 控件通用布局参数

默认情况下在 Utility 或 Rollout 里定义的用户界面控件的布局由 MAXScript 自动处理：按定义顺序自上而下排列，水平、垂直方向对齐。在大多情况下，这种处理能满足要求，但有时我们希望作一些调整，而不仅仅按默认的布局方式，这时我们可以使用几个定义行布局参数。

控件构造语法中的<parameters>的内容随控件类型不同而不同，但也有几个通用的布局参数。除 pos 参数外，这些参数都不是控件的属性，所以不能由脚本程序存取或改变。下面是这些通用的布局参数：

### 1. align:#left

指定控件与 Rollout 边框的对齐方式，默认值随控件类型不同而不同。

### 2. offset: <Point2> 默认值: [0,0]

指定相对于系统自动排列方式的[x,y]偏移，以像素为单位。这种偏移是在别的布局参数之后进行的。

### 3. width: <number> 默认值: 与控件有关

指定控件的宽度，以像素为单位。对 Button 类型的控件非常有用。

### 4. height:<number> 默认值: 与控件有关

指定控件的高度，通常以像素为单位。对 Combobox、Dropdownlist 和 Listbox 类型控件，高度为文本行数。为了使 Combobox 和 Dropdownlist 显示 N 项内容，应将 height 设为 N+2；而如果要让 Listbox 显示 N 行内容，应将 height 设为 N..height 属性对 Spinner 和 Slider 控件没有影响。

### 5. across:<number> 默认值: 1

指定接下来的<number>-1 个控件按水平方向排列，然后又转为正常的垂直排列，这个参数把 Utility 面板等分成<number>列，按正常的对齐方式把每个控件放到每列里。其他的布局参数可以和 across:同时使用来控制控件在每列里的布局。

#### 6. pos:<Point2>      默认值：与控件有关

指定控件在 Rollout 里的位置，以像素为单位。[0,0]原点坐标为 Rollout 的左上角。

下面例子显示了各种布局参数的用法：

```
button foo "foo" align:#right --默认对齐方式为中心对齐
button baz "baz" align:#right offset:[0,6] --右对齐
radiobuttons btn1 "Size: " labels:#("Big", "Bigger", ...) columns:3
checkboxbutton b1 "yes" across:3
checkboxbutton b2 "no"
checkboxbutton b3 "maybe"
--放置 b1、b2、b3 在同一行里的布局
Spinner s1 "Length: " align:#left
--默认对齐方式为左对齐
```

### 15.10.3 控件类型

下面是在 MAXScript 的所有控件类型：

- ◆ Angle (角填充)
- ◆ Bitmap (图像框)
- ◆ Button (按钮)
- ◆ Checkbox (复选框)
- ◆ Checkbutton (复选按钮)
- ◆ Colorpicker (颜色拾取)
- ◆ Combobox (组合框)
- ◆ Curvecontrol (曲线控件)
- ◆ Dropdownlist (下拉列表)
- ◆ Edittext (编辑框)
- ◆ GroupBox (组合框)
- ◆ HyperLink (超级链接)
- ◆ ImgTag (图像)
- ◆ Label (标签)
- ◆ Listbox (列表框)
- ◆ Mapbutton (贴图按钮)
- ◆ Materialbutton (材质按钮)
- ◆ MultiListbox (多选列表框)
- ◆ Pickbutton (对象拾取按钮)
- ◆ PopUpMenu (鼠标右键弹出菜单)
- ◆ Progressbar (进度栏)

- ◆ Radiobuttons (频道按钮)
- ◆ Slider (滑标)
- ◆ Spinner (数值微调器)
- ◆ SubRollout (子 Rollout)
- ◆ Timer (记时器)

例子：

```
utility ui_items "ui items"
(
bitmap a1 bitmap:(bitmap 50 50)
button a2 "button"
Checkbox a3 "Checkbox"
checkbutton a4 "checkbutton"
colorpicker a5 "colorpicker:"
Combobox a6 "Combobox: " items:#("1/2", \
"1/4", "1/8")height:5
Dropdownlist a7 "Dropdownlist:" \
items:#("1/2", "1/4", "1/8")
edittext a8 "edittext:"
label a9 "label"
Listbox a10 "Listbox: " items:#("1/2", "1/4",
"1/8" )height:3
mapbutton a11 "mapButton"
materialbutton a12 "materialbutton"
pickbutton a13 "pickbutton"
progressbar a14
radiobuttons a15 "radiobuttons: "
labels:#("lbl 1", "lbl 2", "lbl 3")
Spinner a16 "Spinner:"
Slider a17 "Slider:"
timer a18
)
```

由上例生成的 Rollout 如图 15-4 所示。

#### 15.10.4 Angle (角填充)

##### 控件语法

```
angle <name> [caption] [color:<color>] \
[degrees:<Float>] \
[bitmap:<bitmap>] \
[diameter:<Float>] \
[StartDegrees:<Integer>] \
[StartRadians:<Integer>] \
[dir:<#cw|#ccw>] \
```

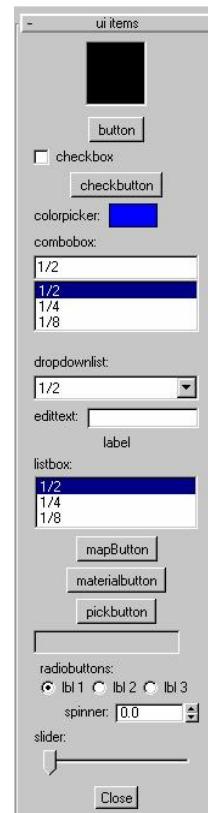


图 15-4 Utilities 面板里的一些控件类型

[range:<Point3>] \

### 属性

属性名称	数据类型	默认值	说明
<angle>.color	Color	(color 0 0 255)	填充角颜色
<angle>.bitmap	Bitmap	undefined	指定填充的位图
<angle>.diamete	Integer	64	直径，与.width 和.height 同义
<angle>.degrees	Float	0.0	用度数定义当前角度
<angle>.radians	Float	0.0	用弧度定义当前角度
<angle>.StartDegrees			定义 0 度的位置
<angle>.StartRadians			定义 0 度的位置
<angle>.dir	Name	#ccw	定义角度正值的方向，可以为#cw（顺时针）或#ccw（逆时针）
<angle>.range	Point3		一个 Point3 值，三个元素分别为角度最小、最大、当前值，单位为度。如果最小、最大值的正负符号相同，按下鼠标移动角度值会得到正确的结果；如果最小、最大值的符号相反，按下 Shift 或 Ctrl 键和鼠标移动角度会得到负的角度值，单独按下鼠标移动角度会得到正的角度值

### 事件

on <angle> changed <arg> do <expr>

当 Angle 控件的角度值被改变时被调用，变量<arg>为控件当前的角度值。

一个 Angle 控件包括下面的 UI（用户界面）要素：

- ◆ 显示标题字符串
- ◆ 处理对齐的布局参数
- ◆ .startDegree 和.startRadians 属性
- ◆ .dir 属性
- ◆ .range 属性

例如：

```
rollout test "Angle Test"
(
    angle ang1 "Angle 1" diameter:50 align:#left \
    range:[-180,180,45] startdegrees:0 dir:#cw color:red across:3
    angle ang2 "Angle 2" diameter:50 align:#center \
    range:[0,270,90] startdegrees:90 dir:#ccw color:green
    angle ang3 "Angle 3" diameter:50 align:#right \
    range:[0,360,120] startdegrees:270 dir:#cw color:blue
    Spinner val1 fieldwidth:40 align:#left across:3 range:[-180,180,45]
    Spinner val2 fieldwidth:40 align:#center range:[0,270,90]
    Spinner val3 fieldwidth:40 align:#right range:[0,360,120]
    on ang1 changed val do val1.value = val
```

```

on ang2 changed val do val2.value = val
on ang3 changed val do val3.value = val
on val1 changed val do ang1.degrees = val
on val2 changed val do ang2.degrees= val
on val3 changed val do ang3.degrees= val
)
createdialog test 200 110

```

由上例生成的 Rollout 如图 15-5 所示。

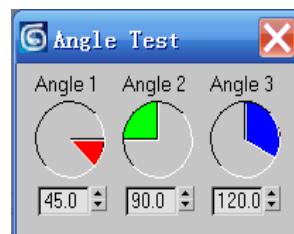


图 15-5 Angle 控件

### 15.10.5 Bitmap (图像框)

Bitmap 控件用来在 Rollout 里放置一个图像框, Bitmap 控件默认值对齐方式为#center, 没有标题和文本与 Bitmap 控件一起显示。其语法为:

```

bitmap <name> [ <caption> ] \
    [ fileName:<filename_String> | bitmap:<bitmap> ]

```

例如:

```
bitmap the_bmp fileName: "my_pic.bmp"
```

参数说明:

- ◆ **Filename:** 表示要加载的位图文件名。位图控件图像在 Rollout 里的大小就是位图文件包含图像的大小。系统会按下面路径顺序搜索指定的位图文件: 当前 MAXScript 路径 → MAXScript startup 路径 → MAXScript 路径 → 3ds max Bitmap 路径 → 3ds max image 路径。
- ◆ **Bitmap:** 表示用户可以在需要参数 **fileName:** 的地方直接指定一个 **bitmap:** 创建参数, 这种方法允许直接用 **fileName:** 参数指定位图文件外, 用户也可以使用那些已存在的 Bitmap 类值, 如调用 **render()** 函数生成的位图。还可以用一个 Bitmap 构造函数将 **bitmap:** 参数设置为一个指定尺寸的“空”图像, 如下例:

```
bitmap BitmapImage bitmap:(bitmap 50 50)
```

#### 属性

##### 1. <bitmap>.fileName      String

Bitmap 控件的图形文件名, 用户可以通过给该属性赋一新值来改变图像的内容, 但显示区的大小不会改变, 将一直保持控件初始化时的大小。

##### 2. <bitmap>.bitMap      textureMap

一个只能写入的属性, 用来设置 bitmap 图像显示的内容。同样用户可以通过给该属性赋一新值来改变图像的内容, 但显示区的大小不会改变, 将一直保持控件初始化时的大小。

#### 事件

无。

### 15.10.6 Button (按钮)

Button 控件用来在 Rollout 上放置一个按钮，用户可以用鼠标点击它来执行某些任务。

Button 控件默认对齐方式为#center。其语法为：

```
button <name> [ <caption> ] [ images:<image_spec_array> ] [ toolTip:<String> ]
```

参数说明：

- ◆ images: 为 Button 控件指定一个元素数据类型为 Bitmap 的数组。如果指定了 images: 参数，系统将忽略<label>参数。images:参数的格式为：

```
images:#(<image>, <maskImage>, <count_Integer>,
          <enabled_out_image_index>, <enabled_in_image_index>,
          <disabled_out_image_index>, <disabled_in_image_index>)
```

其中<image>和<maskImage>可以是一个位图文件名字符串或一个 Bitmap 类值。<counter\_Integer>指定位图里的子图像数目。<image\_index>为四种按钮状态指定子图像索引号。例：

```
bml = render camera:$cam01 outputSize:[80,60].
...
--使用渲染图像作为按钮图像
button foo images:#(bml, undefined, 1, 1, 1, 1)
--使用已有的图像作为按钮图像
button decay images:#( "dcybt�ns.bmp", "dcymask.bmp", 6, 1, 4, 1, 4)
```

- ◆ toolTip: 指定按钮的提示字符，当鼠标停留在按钮上时，在鼠标的右下角显示提示符，如果没有指定 toolTip:参数，则没有提示符。

#### 属性

<button>.images      Array

见 images 属性，该属性为只读属性。

#### 事件

on <button> pressed do <expr>

当用户点击按钮事件发生时调用。

### 15.10.7 Checkbox (复选框)

Checkbox 控件用于在 Rollout 上放置一个复选框，用户可以连续点击它，来切换其状态，默认对齐方式为：#left。其语法为：

```
Checkbox <name> [ <caption> ] [ checked:<Boolean> ]
```

参数说明：

checked: 为 Checkbox 控件的初始状态，默认值为 False。

例如：

```

Checkbox frab_enable "Enable frabing"
on frab_enable changed state do ...
...
if frab_enable.checked then frabulate master_obj
...

```

### 属性

1. <Checkbox>.checked Boolean

设置 Checkbox 控件的状态, On (True) 或 Off (False)。

2. <Checkbox>.state Boolean

与属性.checked 相同。

### 事件

on <Checkbox> changed <arg> do <expr>

当用户点击 Checkbox 控件改变其状态时调用, <arg>变量为 Checkbox 控件新的状态。

## 15.10.8 Checkbutton (复选按钮)

Checkbutton 控件用来在 Rollout 上放置一个有两种状态(On 和 Off)的按钮, 与 Checkbox 类控件类似。默认对齐方式为: #center。其语法为:

```

checkbutton <name> [ <caption> ] [ highlightColor:<color> ] \
[ toolTip:<String> ] [ checked:<Boolean> ] \
[ images:<image_spec_array> ] \

```

参数说明:

- ◆ checked: 为 Checkbutton 控件的初始状态, 默认值为 Off;
- ◆ highlightColor: 为当按钮被按下时的背景色, 默认值为浅灰色, 以与 3ds max 用户界面保持一致;
- ◆ toolTip: 同 Button 控件;
- ◆ images: 同 Button 控件。

例如:

```

checkbutton setup "Setup" checked:True tooltip: "Opens setup panels"
on setup changed state do
    if state == on
        then openRollout setup_pan
        else closeRollout setup_pan

```

### 属性

1. <checkbutton>.checked

同 Checkbox 控件。

2. <checkbutton>.state

同 Checkbox 控件。

## 3. &lt;checkbutton&gt;.images

同 Button 控件。

## 事件

```
on <checkbutton> changed <arg> do <expr>
```

同 Checkbox 控件。

## 15.10.9 Colorpicker (颜色拾取器)

Colorpicker 控件用于在 Rollout 里放置一个 3ds max 颜色拾取样板。用户可以点击该样板来打开一个 Color Selector (颜色选取) 对话框。默认对齐方式为: #left。

其语法为:

```
colorpicker <name> [ <caption> ] [ color:<color> ]      \
[ fieldWidth:<number> ] [ height:<number> ] \
[ title:<String> ]
```

参数说明:

- ◆ color:指定颜色拾取样板的初始颜色, 默认值为 blue。
- ◆ title:指定 Color Selector 对话框的标题, 默认值为 Choose a color。
- ◆ fieldWidth:指定颜色拾取样板的宽度, 以像素为单位, 默认值为 40。
- ◆ height:指定颜色拾取样板的高度, 以像素为单位, 默认值为 16。

例如:

```
colorpicker foo "Wire color:" color:[0,0,255]
on foo changed new_col do selection.wirecolor = new_col
```

## 属性

<colorPicker>.color: Color

当前颜色拾取样板的颜色值。

## 事件

```
on <colorpicker> changed <arg> do <expr>
```

当用户在 Color Selector 对话框里选取了一种新的颜色时调用, <arg>变元为新的颜色。

## 15.10.10 Combobox (组合框)

Combobox 控件用来在 Rollout 上放置组合框。这种控件是另外一种控件类型 Dropdownlist 的变种, 但这种控件的各项目都显示在 Rollout 上, 且在项目清单的顶部有一个附加的 edittextbox 控件来显示当前被选中的项目, 并可被编辑。用户可以上下滚动项目清单并点击清单中的项目。默认对齐方式为: #left。其语法为:

```
Combobox <name> [ <caption> ] [ items:<array_of_Strings> ] \
[ selection:<number> ] [ height:<number> ]
```

参数说明：

- ◆ **text:** 控件通用属性，为显示在控件顶部文本框中的文字；
- ◆ **items:** 为存储控件清单列表的字符串数组；
- ◆ **selection:** 为当前被选中项的号码，最开始一项为 1，默认值为 1；
- ◆ **height:** 为 Combobox 控件的总高度，以项目行为单位，默认值为 10。如果要让控件显示 N 个项目，必须将 height: 设为 N+2。

例如：

```
Combobox scale_cb "Scale" items:#("1/2", "1/4", "1/8", "1/16")
on scale_cb selected i do
    format "You selected %\n!" scale_cb.items[i]
scale_cb.selected = "new item text"
```

### 属性

#### 1. <Combobox>.caption      String

语法定义中的<caption>可选项。

#### 2. <Combobox>.text      String

控件上方附加 Editbox 控件里的文本。

#### 3. <Combobox>.items      Array

元素数据类型为 String 的数组，表示列表中所有项目。

#### 4. <Combobox>.selection      Integer

当前被选中项目的序号，其起始项目序号为 1。如果<Combobox>.items 数组为空，.selection 值为 0。

#### 5. <Combobox>.selected      String

当前被选中项的文本内容。可用来单独改变某项的内容而不用给整个 items 数组重新赋值。如果 items 为一个空数组，.selected 返回值 undefined。

### 事件

#### 1. on <Combobox> selected <arg> do <expr>

当用户在 Combobox 控件列表中选中某项时被调用。<arg>变元包含了当前被选中项的序号。

#### 2. on <Combobox> doubleClicked <arg> do <expr>

当用户双击 Combobox 控件的某项时被调用，注意在用户双击某项时，在第一次点击时会同时调用 on selected 事件处理程序。变元<arg>包含了被双击项的号码。

#### 3. on <Combobox> entered <arg> do <expr>

如果用户在 Combobox 控件附加 Editbox 里改变了文本内容，然后当用户从附加 Editbox 控件转移焦点时被调用。读者应注意：如果用户在附加 Editbox 里按下 Enter 键，并不会调用 on entered 事件处理程序。<arg>变元包含了 Editbox 里的新文本。

#### 4. on <Combobox> changed <arg> do <expr>

用户在 Combobox 控件附加 Editbox 里每改变一个字符，都会被调用。但用户按 Enter

或从附加 Editbox 控件转移焦点时不会调用 on\_changed 事件。变元<arg>包含附加 Editbox 控件里的新文本。

### 15.10.11 CurveControl (曲线控件)

#### 语法

```
editboxCurveControl <name> [ <caption> ] \
    [ visible: <Boolean> ] [ numCurves: <Integer> ] \
    [ x_range: <Point2> ] [ y_range: <Point2> ] \
    [ x_value: <Float> ] \
    [ zoomValues: <Point2> ] [ scrollValues: <Point2> ] \
    [ displayModes: <bitarray> ] \
    [ commandMode: <#move_xy | #move_x | #move_y \
        | #scale | #corner | bezier> ] \
    [ uiFlags: <array_of_ui_flags> ] \
    [ rcmFlags: <array_of_rcm_flags> ] [ asPopup:<Boolean> ]
```

我们可以用下面的方法创建一个曲线点：

```
ccPoint <pt_Point2> <in_point> <out_Point2> \
    [bezier:<False>][corner:<False>][lock_x:<False>] \
    [lock_y:<False>][select:<False>][end:<False>] \
    [noXConstraint:<False>]
```

参数说明：

◆ uiFlags:为下表中所列标记的任意组合：

标记	标记说明
#drawBG	指定在图形窗口上绘制白色背景
#drawGrid	指定在图形窗口上绘制栅格线和坐标
#upperToolbar	指定在控件上面绘制上部工具栏
#showReset	指定在部工具栏里显示 Reset 按钮
#lowerToolbar	指定在控件下方绘制下部工具栏
#scrollBars	指定为控件绘制水平、垂直滚动条
#autoScroll	指定进行自动滚动
#ruler	指定绘制一个小的可移动的标尺窗口（用于测量水平坐标）
#constrainY	指定不能将点（或句柄）移动到超出 SetYRange()设置的范围以外
#hideDisabled	指定不显示不活动的曲线
#all	打开除#singleSelect、#noFilterButtons 和#xvalue 以外的所有标记
#xvalue	指定在图形上绘制一个垂直条显示当前的 X 值
#singleSelect	允许用户选择单独的点。一般来说，如果许多点集中在一个区域，鼠标点击该区域会选择所有的点，设置本标记后会仅选择第一个点

#noFilterButtons	本标记关闭曲线的 visible/editable 触发器（在菜单栏的顶部）
------------------	--

◆ rcmFlags: 为下面标记的任意组合:

#move\_xy、#move\_x、#move\_y、#scale、#corner、#bezier、#delete、#all

### 属性

1. <curvecontrol>.visible: Boolean

设置 CurveControl 控件是否可见。

2. <curvecontrol>.x\_range: Point2

设置第一个和最后一个 CurvePoint 点的 x 坐标值。

3. <curvecontrol>.y\_range: Point2

设置 CurvePoint 点的 y 坐标最大和最小值。本属性仅在指定标记#constrainY 时有效。

4. <curvecontrol>.x\_value: Float

本属性用来在图形窗口上显示一条垂直线（如果标记#xvalue 被指定）。

5. <curvecontrol>.displayModes: BitArray

BitArray 值指定显示哪些曲线，如果其值为空{}，没有曲线被显示，如果设置为{1}，显示第一条曲线；如果设置为{2}，显示第二条曲线；如果设置打开两位或更多，同时显示多条曲线，比如{1,3}同时显示曲线 1 和 3。

6. <curvecontrol>.commandMode: Name

存取当前命令模式。可能的值有：#move\_xy、#move\_x、#move\_y、#scale、#corner、#bezier。

7. <curvecontrol>.zoomValues: Point2

存取水平和垂直方向上的缩放值。

8. <curvecontrol>.scrollValues: Point2

存取水平和垂直方向上的滚动值。

9. <curvecontrol>.curves: Array, 只读

返回一个数据类型为 MAXCurve 的数组。

下面这些属性中<ccCurve>代表 Curvecontrol 控件里的单独一条曲线。

1. <ccCurve>.animatable: Boolean

如果设为 True，该曲线的点可以被设置动画。

2. <ccCurve>.color: Color

存取当曲线活动时的颜色。

3. <ccCurve>.disabledColor: Color

存取当曲线不活动时的颜色。

4. <ccCurve>.width: Integer

存取当曲线活动时的宽度。

5. <ccCurve>.disabledWidth: Integer

存取当曲线不活动时的宽度。

6. <ccCurve>.disabledStyle: Name

<ccCurve>.style: Name

分别设置曲线活动和不活动时的绘制类型。可能的值有：

#solid、#dash、#dot、#dashDot、#dashDotDot、#null、#insideFrame。

7. <ccCurve>.lookupTableSize: Integer

本方法设置 Curve Control 的 lookup 数组的长度。lookup 数组可以加速对 Curve Control 的存取速度。默认值为 1000。

8. <ccCurve>.numPoints: Integer

存取曲线上的点数。

9. <ccCurve>.points: Array, 只读

返回一个 CurvePointsArray 数组。

下面的<ccpoint>代表一个曲线点。

1. <ccpoint>.value: Point2

返回指定点的坐标值。

2. <ccpoint>.inTangent: Point2

返回指定点的 inTangent 值。

3. <ccpoint>.outTangent: Point2

返回指定点的 outTangent 值。

4. <ccpoint>.bezier: Boolean

如果设为 True, 点将表现为一个 Bezier 点。

5. <ccpoint>.corner: Boolean

如果设为 True, 点将表现为一个 Corner 点。但如果.bezier 属性为 True, 该点为一个 Bezier-Corner 点。

6. <ccpoint>.lock\_x: Boolean

如果设为 True, 点的 X 坐标被锁定。

7. <ccpoint>.lock\_y: Boolean

如果设为 True, 点的 Y 坐标被锁定。

8. <ccpoint>.selected: Boolean

如果设为 True, 点被选择。

9. <ccpoint>.end: Boolean

如果设为 True, 没有终点。

10. <ccpoint>.noXConstraint: Boolean

如果设为 False, 点的 X 坐标被约束。

## 方法

zoom <curvecontrol> #all

执行一个 zoom extents all 命令。

下面的<ccCurve>代表 Curvecontrol 控件里的单独一条曲线。

1. getValue <ccCurve> <time\_value> <Float\_x> [lookup:<False>]

返回一个 Point2 值，代表在指定时间、指定 X 坐标对应的曲线上点坐标。

2. isAnimated <ccCurve> <index>

如果指定序号点被设置动画，返回 True，否则返回 False。

3. getSelectedPts ccCurve

返回一个 BitArray 值，表示被选择的点。

4. setSelectedPts <ccCurve> <bitarray> [&#select] [#deSelect] [#clear]

选择<bitarray>里指定的点。

5. setPoint <ccCurve> <index> <ccpoint> [checkConstraints:<True>]

将曲线的指定点设为指定值，如果参数 checkConstraints 为 True，该点会被约束。

6. getPoint <ccCurve> <index>

返回一个 Point2 值，代表曲线上指定序号的点。

7. insertPoint <ccCurve> <index> <ccpoint>

在指定位置用指定值插入一个新点。

8. deletePoint <ccCurve> <index>

从曲线上删除指定点。

## 事件

在下面的事件里，变量<ci>代表活动曲线的序号。

1. on <curvecontrol> selChanged <ci> <val> do <expr>

当一个点被选择或被解除选择时被执行。变量<val>为被选择的点编号。

2. on <curvecontrol> ptChanged <ci> <val> do <expr>

当一个点被修改时被执行。变量<val>为被修改的点编号。

3. on <curvecontrol> tangentChanged <ci> <val> type do <expr>

当一个点的 in 或 out 切线被修改时被执行。变量<val>为被修改的点编号。

变量 type 可能为#inTangent 或#outTangent，取决于之前的修改。

4. on <curvecontrol> deleted <ci> <val> do <expr>

当点删除时被执行。变量<val>为被删除的点编号。

5. on <curvecontrol> reset <ci> do <expr>

当曲线重置时被执行。

下面是一个 Curvecontrol 控件的例子，图 15-6 为运行结果。

```
rollout uTestCurveControl "Curve Control"
(
    --Curvecontrol 控件属性
    local ccProps = #(
        #visible,
        #numCurves,
        #x_range,
        #y_range,
        #displayModes,
        #zoomValues,
        #scrollValues,
```

```
#commandMode)

--Curve 属性
local curveProps = #(
    #name,
    #animatable,
    #color,
    #disabledColor,
    #width,
    #disabledWidth,
    #style,
    #disabledStyle,
    #numPoints,
    #lookupTableSize)

--Curve Point 属性
local cpProps = #(
    #value,
    #inTangent,
    #outTangent,
    #bezier,
    #corner,
    #lock_x,
    #lock_y,
    #selected,
    #end,
    #noXConstraint)

button btn_props "Print Properties"
Checkbox chk_visible "Visible" checked:True
Checkbox chk_enable "Enable" checked:True

CurveControl cc_test "Curve Control: " \
height:200 \
width:400 \
align:#center \
numCurves:2 \
visible:True \
x_range:[-100,100] \
y_range:[-100,100] \
scrollValues:[-100,100] \
commandMode:#move_xy \
--下面参数如果没有被指定，默认值为所有标记:
--uiFlags:#( #drawBG, #drawGrid, #upperToolbar, #showReset,
--#scrollBars, #constrainY, #xvalue)
rcmFlags:#(#delete) \
asPopup:False

on chk_visible changed state do cc_test.visible = state
on chk_enable changed state do cc_test.enabled = state
```

```

on uTestCurveControl open do
(
zoom cc_test #all
local colors = #(red, green, blue)
local styles = #(#solid, #dash, #dot, #dashDot, #dashDotDot,\n
#null, #insideFrame)
local num = cc_test.numCurves

--初始化 curve 属性
for i=2 to num do
(
local crv = cc_test.curves[i]
local ci =((mod(i-1)colors.count)+1)as Integer
local si =((mod(i-1)styles.count)+1)as Integer

crv.name = "Curve" + i as String
crv.color = colors[ci]
crv.disabledColor = colors[ci]*0.5
crv.style = styles[si]
--crv.width = crv.disabledWidth = i
crv.numPoints = i*2

local del = (cc_test.x_range.y - cc_test.x_range.x)/(crv.numPoints-1)
--format "del:%\n" del
--等间距布点
for j=1 to crv.numPoints do
(
    local cp = crv.points[j]
    format "% end: % -> " j cp.end
    --cp.corner = True
    cp.value = [cc_test.x_range.x+(j-1)*del, cp.value.y]
    cp.inTangent = [0.2,0.2]
    cp.outTangent = [0.2,-0.2]
    crv.points[j] = cp
    format "%\n" crv.points[j].end
    format "value: %\n" crv.points[j].value
) --for j 结束
) --for i 结束
) -- on uTestCurveControl 结束

on btn_props pressed do
(
local tab = "\t"
--打印 CC 属性
format "Curve Control Properties\n"
for p in ccProps do
format(tab + "%: %\n")(p as String)(getProperty cc_test p)
format(tab + "Curves:\n")
tab += "\t"
for i=1 to cc_test.numCurves do

```

```

(
local crv = cc_test.curves[i]
--打印每一 Curve 的属性
format(tab + "Curve%\n")i
for p in curveProps do
format(tab + "\t%: %\n")(p as String)(getProperty crv p)
format(tab + "\tPoints:\n")
for j=1 to crv.numPoints do
(
local cp = crv.points[j]
format(tab + "\t\tPoint%:\n")j
--打印每一 Curve point 的属性
for pp in cpProps do
format(tab + "\t\t\t%: %\n")(pp as String)(getProperty cp pp)
) --for j 结束
) --for i 结束
) -- on btn_props 结束
--Curve control 事件处理程序
on cc_test selChanged ci val do
format "curve % numSelected: %\n" ci val
on cc_test ptChanged ci val do format "curve % ptChanged: %\n" ci val
on cc_test tangentChanged ci val type do
format "curve % tangentChanged: % %\n" ci val(type as String)
on cc_test deleted ci pi do format "curve % deleted: %\n" ci pi
on cc_test reset ci do format "curve % reseted\n" ci
)
curveFloater = newRolloutFloater "Curve Control Test" 500 400
addRollout uTestCurveControl curveFloater

```

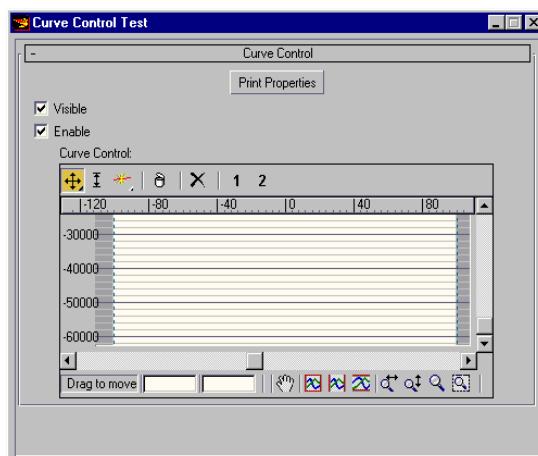


图 15-6 Curve Control 控件运行结果

### 15.10.12 Dropdownlist (下拉列表)

Dropdownlist 控件用来在 Rollout 里加入一个下拉列表。用户可用鼠标点击来打开列表并且按住鼠标上下移动来上下翻滚列表项，然后再次点击来选择某项。默认对齐方式为：

#left。其语法为：

```
Dropdownlist <name> [ <caption> ] [ items:<array_of_Strings> ] \
[ selection:<number> ] [ height:<number> ]
```

参数说明：

- ◆ items:为包含下拉列表中所有项的字符串数组；
- ◆ Selection:同 Combobox；
- ◆ height:同 Combobox。

例如：

```
Dropdownlist scale_dd "Scale" items:#("1/2", "1/4", "1/8", "1/16")
on scale_dd selected i do
    format "You selected %\n! " scale_dd.items[i]
```

#### 属性

<Dropdownlist>.items	Array
----------------------	-------

<Dropdownlist>.selection	Integer
--------------------------	---------

<Dropdownlist>.selected	String
-------------------------	--------

均同 Combobox。

#### 事件

on <Dropdownlist> selected <arg> do <expr>

当用户在下拉列表中选中某项时被调用，变元<arg>包含了当前被选中项的号码。

例如：

```
Rollout test "t"
(Dropdownlist dd "dd" items:#("1", "2", "3", "4", "5", "6", "7", "8", "9") \
height:6 label l "L"
)
rof=newRolloutFloater "A" 200 200
addRollout test rof
```

### 15.10.13 Edittext (编辑框)

Edittext 控件用来向 Rollout 添加一个可编辑的文本框，用户可以输入和编辑文本。默认对齐方式为：#left。其语法为：

```
edittext <name> [ <caption> ] [ text:<String> ] \
[ fieldWidth:<Integer> ] [ bold:<Boolean> ]
```

参数说明：

- ◆ text:为编辑框里的文本；
- ◆ fieldWidth:为文本框的宽度，以像素为单位。默认的宽度为紧接标题文字到 Rollout 的右边界；
- ◆ bold:如果设为 True，文本以 bold 格式显示；如果设为 False，文本以正常格式显示，

默认值为 False。

例如：

```
edittext prefix_txt "Name prefix:" fieldWidth:70
on prefix_txt entered text do ...
...
new_obj = copy master pos:[x,y,z] prefix:prefix_txt.text
...
```

#### 属性

1. <edittext>.text      String

存取文本框里的文本。

2. <edittext>.caption      String

存取控件标题。

3. <edittext>.bold      Boolean

.bold 如果设为 True，文本以 bold 格式显示；如果设为 False，文本以正常格式显示，  
默认值为 False。

#### 事件

1. on <edittext> changed <arg> do <expr>

用户每次改变文本框中的一个字符，都会调用变元<arg>包含文本框中的新文本。

2. on <edittext> entered <arg> do <expr>

用户在文本框中输入文本然后按 Enter 或 TAB 键将光标移出文本框时被调用。变元<arg>包含了文本框中的新文本。

如果用户在 Editbox 里输入字符，然后按下 Enter 键，每输入一个字符，就调用一次 on changed 事件处理程序，但仅当按下 Enter 键时调用一次 on entered 事件处理程序。

### 15.10.14 GroupBox（组合框）

GroupBox 控件可用来在界面指定位置处绘制一个指定大小的矩形，控件可以有标题，也可以没有。和 Rollout 子句里的 Group 不一样，GroupBox 控件并不用来将界面控件形成一个组，而是可以在 Rollout 里独立布置的一种控件。

#### 语法

```
groupBox <name> [caption] [pos:<Point2>] \
[width:<Integer>] [height:<Integer>]
```

#### 属性

1. <groupBox>.caption

存取 groupBox 控件标题。

2. <groupBox>.pos

存取 groupBox 控件左上角位置。

3. <groupBox>.width

存取 groupBox 控件宽度。

4. <groupBox>.height

存取 groupBox 控件高度。

**注意** 在 Visual MAXScript 里GroupBox 控件仅用来在 Rollout 里定义一个组。

例如：

```
rollout test "GroupBox Test"
(
    Spinner spin "Spin1:" align:#left across:2 offset:[0,20]
    Spinner unspin "Spin2: " \
        align:#right offset:[0,20]
    groupBox group1 "Group 1" pos:[5,5] \
        width:95 height:50
    groupBox group2 "Group 2" pos:[105,5] \
        width:90 height:50
)
createDialog test 200 60
```

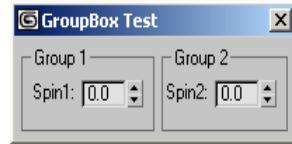


图 15-7 GroupBox 控件

运行结果如图 15-7 所示。

### 15.10.15 HyperLink (超级链接)

语法

```
hyperLink <name> [caption] [color:<color>] \
    [hoverColor:<color>] [visitedColor:<color>] [address:<String>]
```

属性

1. <hyperLink>.color: Color 默认值：(color 0 0 255)

存取文本基色。

2. <hyperLink>.hoverColor: Color 默认值：(color 0 255 255)

当鼠标移动至文本上面时的文本颜色。

3. <hyperLink>.visitedColor: Color 默认值：(color 0 0 192)

当使用过指定链接后的文本颜色。

4. <hyperLink>.address: String 默认值：“http://www.discreet.com”

存取发送到 ShellExecute 的字符串。

### 15.10.16 ImgTag (图像)

ImgTag 控件可用来在界面上显示指定位图。

语法

```
imgTag <name> [caption] [tooltip:<String>] [style:<key>] \
    [bitmap:<bitmap>] [opacity:<Float>] [transparent:<color>]
```

### 属性

1. <imgTag>.tooltip: String 默认值: “”  
当鼠标移动至 ImgTag 控件上面时所显示的提示文本。
2. <imgTag>.bitmap: Bitmap 默认值: undefined  
当前显示的位图。
3. <imgTag>.style: Key 默认值: #bmp\_stretch  
设置位图在控件里的显示方式，可以为下面的 Name 值：  
#bmp\_stretch、#bmp\_tile、#bmp\_center。
4. <imgTag>.opacity: Float 默认值: 0.0  
设置图像的总体不透明度，取值范围为[0.0 1.0]。
5. <imgTag>.transparent: Color 默认值: (color 0 0 0)  
将指定颜色设为透明色。

**注意** 属性.opacity 和.transparent 仅在 Windows 2000 里有效而在 NT 里无效。

### 事件

1. on <imgTag> mousedown do <expr>  
当鼠标按下左键时被调用。
2. on <imgTag> mouseup do <expr>
3. on <imgTag> click do <expr>  
当松开鼠标左键时被调用。
4. on <imgTag> dblclick do <expr>  
当双击鼠标左键时被调用。
5. on <imgTag> mouseover do <expr>  
当鼠标移进图像区域时被调用。
6. on <imgTag> mouseout do <expr>  
当鼠标移出图像区域时被调用。

## 15.10.17 Label (标签)

Label 控件用来在 Rollout 里加入一个静态的文本，它可能是另一个控件的标签，也可能是一个提示信息。默认对齐方式为：#center。其语法为：

label <name> [<String>]

例如：

```
label lab1 "Please choose an object:"
```

### 属性

Label 控件没有额外的属性。

### 事件

无

#### 15.10.18 Listbox (列表框)

Listbox 控件是 Dropdownlist 控件的另一个变种，它和 Combobox 不同的是，它的顶部没有附加的 Editbox 控件，而只有一个可以上下翻滚的列表。默认对齐方式为：#left。其语法为：

```
Listbox <name> [ <caption> ] [ items:<array_of_Strings> ] \
[ selection:<number> ] [ height:<number> ]
```

参数说明：

- ◆ items: 同 Combobox 控件；
- ◆ selection: 同 Combobox 控件；
- ◆ height: 为列表框的总高度，以行为单位。默认值为 10 行，如果想在 Listbox 列表框中显示 N 行，将.height 设为 N。

例如：

```
Listbox selns items:obj_name_array
on selns selected i do
    copy obj_array[i] pos:[rand_x, rand_y, rand_z]
selns.selection = 2
```

### 属性

```
<list box>.items: Array
<list box>.selection
<list box>.selected
```

均同 Combobox 控件。

### 事件

```
on <Listbox> selected <arg> do <expr>
on <Listbox> doubleClicked <arg> do <expr>
```

同 Combobox 控件。

#### 15.10.19 Mapbutton (贴图按钮)

Mapbutton 控件用来在 Rollout 上加入一个特殊的按钮，当按下这个按钮时，会打开 3ds max 的 Material/Map Browser 对话框。在打开的对话框里仅显示贴图。其语法为：

```
mapbutton <name> [ <caption> ] [ map:<texturemap> ] \
[ images:<image_spec_array> ] [ toolTip:<String> ]
```

参数说明：

- ◆ map:为用户尚未使用按钮时的初始 textureMap 值, 默认值为 undefined;
- ◆ images:同 Button 控件;
- ◆ toolTip:同 Button 控件。

例如:

```
label    sbm_lbl "Background Map:"
mapbutton choosemap "<<none>>" tooltip: "Select Background Map"
on choosemap picked texmap do
(environmentmap = texmap
choosemap.text=classof texmap as String
)
```

#### 属性

##### 1. <mapbutton>.map: textureMap

存取 Mapbutton 控件的当前 textureMap 值, 如果用户尚未使用按钮, 返回由参数 map: 指定的值。

##### 2. <mapbutton>.images

同 Button 类控件。

#### 事件

##### on <mapbutton> picked <arg> do <expr>

当用户从 Material/Map Browser 对话框里选择了一个 TextureMap 值时被调用, 变元 <arg> 包含了被选中的 textureMap 值。如果用户以按 Cancel 方式退出对话框, 不会调用 on picked 事件处理程序。

### 15.10.20 Materialbutton (材质按钮)

Materialbutton 控件用来在 Rollout 里加入另一种特殊的按钮, 当按下按钮时, 也会打开 3ds max 的 Material/Map Browser 对话框, 但只有 material 才会显示在对话框中。默认的对齐方式为: #center。其语法为:

```
materialbutton <name> [ <caption> ] [ material:<material> ] \
[ images:<image_spec_array> ] [ toolTip:<String> ]
```

参数说明:

- ◆ material:为用户尚未使用按钮时的初始 Material 值, 默认值为 undefined;
- ◆ images:同 Button 控件;
- ◆ toolTip:同 Button 控件。

例如:

```
label    smtl_lbl "Set selected object's material to: "
materialbutton choosemtl "Pick Material"
on choosemtl picked mtl do
(print mtl
```

```
if $ != undefined do $.material=mtl
)
```

### 属性

#### 1. <materialbutton>.material Material

Materialbutton 控件的当前 Material 值, 如果用户尚未使用按钮, 返回由参数 Material: 指定的值。

#### 2. <materialbutton>.images

同 Button 控件。

### 事件

#### on <Materialbutton> picked <arg> do <expr>

当用户从 Material/Map Browser 对话框里选择了一个 Material 值时被调用, 变元<arg> 包含了被选中的 Material 值。如果用户以按 Cancel 方式退出对话框, 不会调用 on picked 事件处理程序。

## 15.10.21 MultiListbox (多选列表框)

MultiListbox 类控件用来在 Rollout 中加入一个列表框, 这种控件是 Listbox 类控件的一个变种: 这种控件一次可以选择列表中的多个项目。默认的对齐方式为: #left。其语法为:

```
MultiListbox <name> [ <caption> ] [ items:<array_of_Strings> ] \
[ selection:{<bitarray> | <number_array> | <number>} ] \
[ height:<number> ]
```

参数说明:

- ◆ items: 为元素数据类型为字符串的数组, 表示列表中所有项目。
- ◆ selection: 为一个 BitArray 值, 表示当前被选择的列表中的项目, 默认值为#{}。
- ◆ height: 为 MultiListbox 控件的总高度, 以项目行为单位, 默认值为 10。如果要让控件显示 N 个项目, 必须将 height 设为 N。

例如:

```
Rollout test "test"
(MultiListbox mlb "MultiListbox" \
items:#("A", "B", "C") selection:#(1,3)
on mlb selected val do format "selected:
% - %\n" val mlb.selection[val]
on mlb doubleclicked val do format "doubleclicked: % - %\n" \
val mlb.selection[val]
on mlb selectionEnd do format "selectionEnd: %\n" mlb.selection
)
rof=newRolloutFloater "tester" 200 300
addRollout test rof
test.mlb.items
test.mlb.selection=1
```

```
test.mlb.selection=#(1,3)
test.mlb.selection=#{}
```

### 属性

#### 1. <Listbox>.items: Array

元素为字符串的数组，表示列表中所有项目。

#### 2. <Listbox>.selection: BitArray

元素为当前被选中项目号码的 BitArray 值。如果通过脚本给本属性赋值，值类型可以为 BitArray、整数数组或单个整数。项目编号从 1 开始。如果<Listbox>.items 数组为空，.selection 值为 0。

### 事件

#### 1. on <Listbox> selected <arg> do <expr>

当用户在 MultiListbox 控件列表中选中某项或解除某项的选择时被调用。<arg>变元包含了当前被选中项或被解除选择项的序号。因为在 MultiListbox 控件里一次可以选择或解除选择多项，本事件处理程序会对每一个选择状态改变的项调用一次，从列表顶部开始。

#### 2. on <Listbox> doubleClicked <arg> do <expr>

当用户双击 MultiListbox 控件的某项时被调用，注意在用户双击某项时，在第一次点击时会同时调用 on selected 事件处理程序。变元<arg>包含了被双击项的序号。

#### 3. on <Listbox> selectionEnd do <expr>

当用户进行选择或解除选择操作时，在系统执行完全部的 on selected 事件后再调用本事件。

**注意** 没有办法通过点击列表的某处来解除对所有项的选择，如果要做到这一点，用户可以在界面上加一个“清除选择”的按钮，在按钮的 on picked 事件处理程序里设置：  
`<Listbox>.selection=#{}`

## 15.10.22 Pickbutton (对象拾取按钮)

Pickbutton 控件用来在 Rollout 上加一个场景对象拾取按钮。它的功能和 3ds max 用户界面里的对象拾取按钮是一样的，当被按下时，按钮颜色变成浅绿色，系统进入对象拾取模式，此时用户可以通过鼠标点击场景对象来选取它，然后系统退出拾取模式，按钮的颜色恢复原状。用户可以右击鼠标来取消拾取模式。默认对齐方式为：#center。语法为：

```
pickbutton <name> [ <caption> ] [ message:<String> ] \
[ filter:<function> ] [ toolTip:<String> ]
```

参数说明：

- ◆ **message:**为可选参数，当处于拾取模式时显示在 3ds max 状态栏的提示文本。默认值为没有提示信息。
- ◆ **filter:**为可选的过滤函数，来检验鼠标指向的场景对象是否符合设定的标准。它必须是一个只有一个变元的函数，而且这个变元就是要检测的场景对象。如果对象符合

条件，返回 True，否则返回 False。默认条件下没有过滤函数。  
下面的例子中，过滤函数让用户仅能选择那些.name 属性以 foo 开头的对象。

```
fn foo_filt obj = findString obj.name "foo" == 1
```

- ◆ toolTip: 同 Button 控件。

例如：

```
fn foo_filt obj = findString obj.name "foo" == 1
pickbutton chooseit "Select master object" filter:foo_filt
on chooseit picked obj do
(master = obj
chooseit.text = obj.name
)
```

### 属性

<pickbutton>.object: Node

用 Pickbutton 控件拾取的最后一个对象，如果没有对象被拾取，返回 undefined，本属性为只读。

### 事件

on <pickbutton> picked <arg> do <expr>

在拾取模式下当用户选择了一个场景对象时被调用。变元<arg>包含了被选择的对象。

## 15.10.23 PopupMenu (右键弹出菜单)

PopupMenu 控件用来建立一个鼠标右键弹出菜单。

### 语法

popupMenu <RCMenu> [pos:<Point2>] [rollout:<Rollout>] [align:<key>]

本方法可在屏幕的任何位置弹出一个菜单。

参数说明如下：

- ◆ <RCMenu> 为任何已定义的 RCMenu，其不一定要被注册；
- ◆ pos:<Point2> 为可选参数，显示弹出菜单的点位置，坐标值可以相对于 Rollout，也可以相对于屏幕。默认值 popup 表示弹出菜单显示在当前鼠标位置；
- ◆ rollout:<Rollout> 为可选参数，指定弹出菜单的原点位置为指定 Rollout 的坐标原点。默认值 undefined 指定弹出菜单的原点位置为屏幕左上角。
- ◆ [align:<key>]: 指定对齐方式，可以为下面值：
  - #align\_topleft (左上角对齐)
  - #align\_topcenter (上中对齐)
  - #align\_topright (上右对齐)
  - #align\_bottomleft (下左对齐)

```
#align_bottomcenter (下中对齐)
#align_bottomright (上右对齐)
#align_vcenterleft (视窗左中对齐)
#align_vcentercenter (视窗中心对齐)
#align_vcenterright (视窗右中对齐)
```

### 15.10.24 ProgressBar (进度栏)

ProgressBar 控件用来在 Rollout 上加入一个进度栏，默认对齐方式为：#left。其语法为：

```
progressBar <name> [ value:<number> ] [ color:<color> ] [ orient:<name> ]
```

参数说明：

- ◆ value: 为整数初始进度百分值 (0~100)，默认为 0。
- ◆ color: 为进度栏颜色，默认值为 [30,10,190]。
- ◆ orient: 指定进度栏放置方式：#horizontal (水平放置)、#vertical (垂直放置)。

例如：

```
button doit "Process Scene"
progressbar doit_prog color:red
on doit pressed do
(for i = 1 to objArray.count do
(doit_prog.value = 100.*i/objArray.count
...
)
doit_prog.value = 0
)
```

#### 属性

1. <progressbar>.value: Integer

存取进度条完成百分值 (0~100)。

2. <progressbar>.color

参见参数 color:。

3. <progressbar>.orient

参见参数 orient:。

#### 事件

on <progressbar> Clicked <arg> do <expr>

当用户点击进度条时调用，变元<arg>包含点击时的百分值。

### 15.10.25 Radiobuttons (单选按钮)

Radiobutton 控件用来向 Rollout 加入单选按钮，一次只能有一个按钮被选中。默认对齐方式为：#center。其语法为：

```
radiobuttons <name> [ <caption> ] labels:<array_of_Strings> \
[ default:<number> ] [ columns:<number> ]
```

参数说明：

- ◆ **labels:**为字符串数组。数组长度为单选按钮的个数，每个字符串元素指定了一个按钮的标签。默认值为 1，表示单选按钮初始化时被选中的按钮；
- ◆ **columns:**指定每行排列几个按钮。默认方式下，MAXScript 会尽量把按钮按水平排列，如果水平方向排列不下，就垂直排列，通过指定 columns:参数，可以强制按钮排成几列。

例如：

```
radiobuttons copy_type labels:#("copy", "instance", "reference")
radiobuttons which_obj labels:obj_name_array
on copy_type changed state do ...
...
copyfn = case copy_type.state of
(1: copy
 2: instance
 3: reference
)
...
```

#### 属性

<radiobuttons>.state: Integer

当前被选中按钮的序号，序号从 1 开始。

#### 事件

on <radiobuttons> changed <arg> do <expr>

当用户点击 Radiobuttons 的按钮组时被调用，变元<arg>包含了控件的新的 state 属性。

### 15.10.26 Slider (滑标)

Slider 控件用来向 Rollout 里加入一个滑标，它是一种 Spinner 控件的替代品，用户可以左右拖动滑标。默认对齐方式为：#center。其语法为：

```
Slider <name> [ <caption> ] [ range:[min,max,val] ] \
[ type:#Float ] [ ticks:10 ] [ orient:#horizontal ]
```

参数说明：

- ◆ **range:**为 Point3 值，x、y、z 分别代表最小值、最大值和初始值。默认值为[0,100,0]。
- ◆ **type:**滑标类型有#Float 和#Integer 两种，默认值为#Float。
- ◆ **orient:**表示布局方向，有#vertical(垂直)和#horizontal(水平)两种，默认为#horizontal。
- ◆ **ticks:**表示沿滑杆放置多少刻度。如果指定 ticks:0，没有刻度。默认值为 10。

例如：

```
Slider tilt "Head tilt" orient:#vertical ticks:0 range:[-30,30,0]
on tilt changed val do $head.bend.angle = val
```

### 属性

1. <Slider>.range: Point3

2. <Slider>.value

存取当前滑标值。

3. <Slider>.ticks: Integer

### 事件

1. on <Slider> changed <arg> do <expr>

当用户移动滑杆值时被调用。变元<arg>包含滑杆的新值。

2. on <Slider> buttondown do <expr>

当用户首次点击滑杆时被调用。

3. on <Slider> buttonup do <expr>

当用户松开滑杆时被调用。

## 15.10.27 Spinner (数值微调器)

Spinner 控件用来向 Rollout 里加入一个数值微调器，用户可以点击或拖拉微调箭头或直接在编辑区里输入数值，默认对齐方式为：#right。其语法为：

```
Spinner <name> [ <caption> ] [ range:[min,max,val] ] \
[ type:<name> ] [ scale:<Float> ] \
[ fieldWidth:<Integer> ] [ controller:<controller> ]
```

例如：

```
Spinner frab_amt "Frab %:" range:[0,100,10] type:#Integer
Spinner ball_radius "Ball radius" controller:$ball.radius.controller
```

参数说明：

- ◆ range: 为一个 Point3 类值，其 x、y、z 值分别为 Spinner 控件的最大、最小、初始值，  
默认值为[0,0,0]
- ◆ type: 为 Spinner 控件的数据类型，有#Float、#Integer 和#Worldunits 三种，  
默认值为#Float。如果类型为#Worldunits，数值以当前 3ds max 设置的显示单位来显示。
- ◆ scale: 为 Spinner 控件每按一次上、下箭头所增减的数据量。浮点型 (#Float) Spinner  
控件的默认值为 0.1，整型 (#Integer) Spinner 控件的默认值为 1。
- ◆ fieldwidth: 为 Spinner 控件文本编辑框的宽度，以像素为单位。默认方式下，Spinner  
控件文本编辑框的左边在 Rollout 的中线上，右边正好在 Rollout 的右边界上。
- ◆ controller: 指定 Spinner 控件和特定的 Controller (控制器) 相连，这样可以在 Spinner  
控件和 Controller 之间建立直接的链接。用户对 Spinner 控件数值的改变可以自动更  
新 Controller。反过来对 Controller 的改变也会自动更新 Spinner 控件的数值。假如

你建立了一个球体名为 ball，并为它的半径值赋了一个 Float\_controller，下面的程序设置了一个 Spinner 控件，自动影响并能记录场景对象 ball 的半径变化。

```
Utility foo "foo"
(
    Spinner ball_radius "Ball radius" range:[0,1000,1] \
    controller:$ball.radius.controller
)
```

Spinner 控件的任何变化会导致 ball 对象的半径改变，而用户对 ball 半径的改变（如在 Modify 面板或动画）也会随时刷新 Spinner 控件的值。注意被指定与 Spinner 相连的 Controller 必须在创建 Spinner 控件之前已存在，该 Controller 可以是已经被赋予用户想要链接的参数，也可以在脚本里先创建一个 Controller，然后再将它赋给指定的参数。如下例：

```
Utility foo "foo"
(
    local myController=bezier_Float()
    Spinner ball_radius "Ball radius" range:[0,1000,1] \
    controller:myController
    button apply "Apply Radius Controller"
    on apply pressed do
        (animate off at time 0 $ball.radius=myController.value
        $ball.radius.controller=myController
    )
)
```

### 属性

<Spinner>.range: Point3，参见参数 range 说明

<Spinner>.value: Float，当前 Spinner 控件的数值

### 事件

1. on <Spinner> changed <arg> do <expr>

当用户调整 Spinner 控件数值或直接在编辑框里输入数值后按 Enter 或 Tab 键后被调用。如果用户设置了 on changed 事件，那么在调用 on entered 之前，会先调用 on changed 事件，且 Spinner 控件数值已被刷新。

2. on <Spinner> buttondown do <expr>

当用户第一次点击 Spinner 时被调用。

3. on <Spinner> buttonup do <expr>

当用户松开 Spinner 时被调用。

## 15.10.28 SubRollout

SubRollout 控件为 RolloutControl 的子类。所有 Rollout 界面控件通用属性和布局参数同样适用于 SubRollout 类。

SubRollout 控件可以放置在 Rollout 里，它没有标题显示，如果它里面包含的界面控件的高度大于它的高度，会在控件的右侧显示一个滚动条。

## 属性

.rollouts: 通过方法 AddSubRollout 安装 SubRollout 控件的 rollout，只读属性。

关于界面通用属性说明如下：

- ◆ .caption: 不显示，可读写
- ◆ .text: 不显示，可读写
- ◆ .enabled: 改变本设置无效，总是为 True，可读写
- ◆ .pos: 控件的位置，可读写
- ◆ .align: 控件对齐方式，仅在创建时使用
- ◆ .offset: 控件偏移距离，仅在创建时使用
- ◆ .width: 控件宽度
- ◆ .height: 控件高度
- ◆ .across: 控件水平方向排列个数，仅在创建时使用

## 相关方法

1. addSubRollout <subrollout> <rollout> [rolledUp:<Boolean>]

向 Subrollout 控件添加指定 Rollout，总是返回 OK。

2. removeSubRollout <subrollout> <rollout>

从 Subrollout 控件删除指定 Rollout，总是返回 OK。

例如：

```
rollout test "test" height:200
( subrollout test1 "test1"
subrollout test2 "test2"
)
```

```
rollout test1a "test1a"
(Spinner test1as "test1as"
)
```

```
rollout test1b "test1b"
(Spinner test1bs "test1bs"
)
```

```
createdialog test
```

```
AddSubRollout test.test1 test1a
AddSubRollout test.test1 test1b
```

```
test.test1.height += 100
test.test2.pos += [0,100]
```

```
rollout test2a "test2a"
(Spinner test2as "test2as"
)
```

```
AddSubRollout test.test2 test2a
test.test2.height += 50
```

上面例子生成的窗口如图 15-8 所示。

**注意** 改变 SubRollout 控件的.height 属性不会改变它里面任何控件的位置。

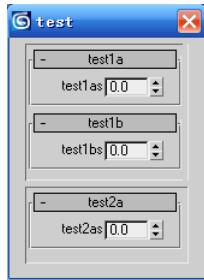


图 15-8 Subrollout 控件生成的窗口

### 15.10.29 Timer (计时器)

Timer 控件用来在 Rollout 上加入一个计时器，每隔指定的时间间隔会产生一个 tick 事件。Timer 控件在用户界面上是不可见的，但它并不是多余的，它使程序可以在没有用户干涉的情况下作出一些反应，比如播放动画，自动检测某些条件或事件等。其语法为：

```
timer <name> [ interval:<number> ] [ active:<Boolean> ]
```

例如：

```
Utility test "test1"
(
    timer clock "testClock" interval:400
    label test "1"
    on clock tick do
        (valUp = (test.text as Integer)+1
         test.text = valUp as String
        )
)
```

参数说明：

- ◆ interval: 表示以毫秒为单位的整数，指定两个 tick 事件之间的时间间隔，默认值为 1000 (1 秒)；
- ◆ active: 指定 Timer 控件是否开始计时，默认值为 True。

#### 属性

```
<timer>.interval: Integer
<timer>.active: Boolean
```

### 事件

on <timer> tick do <expr>

当 Timer 控件计时到 interval 长度时被调用。

## 15.11 图像按钮

MAXScript 里有四种 Rollout 控件 (Button、Checkbutton、Mapbutton 和 Materialbutton) 除了可以用文本作为标签以外，都还可以用图像来作为标签。用户可以用命令行参数来指定显示在按钮上的图像，参数格式为：

```
images:#(<image>, <maskImage>, <count_Integer>,
          <enabled_out_image_index>, <enabled_in_image_index>,
          <disabled_out_image_index>, <disabled_in_image_index>)
```

其中<image>和<maskImage>既可以是一个位图文件，也可以是一个 MAXScript 的 Bitmap 值。<count\_Integer>指定了 Bitmap 值里包含的子图像数，<xxx\_image\_index>指定了哪一种按钮状态使用哪一个子图像。如下面的例子将一个渲染位图作为按钮图像：

```
bm1 = render camera:$cam01 outputSize:[80,60]
...
checkbutton foo images:#(bm1, undefined, 1, 1, 1, 1)
```

如果参数<image>为一个文件名，系统将按下面的顺序查找文件：当前 MAXScript 路径→3ds max 的 startup 路径→MAXScript 路径→Bitmap 路径→image 路径→3ds max 执行路径→Windows 环境变量 Path 指定的路径。

四个图像分别为按钮的四种可能的状态指定了图像，按钮的四种状态分别为：Enabled-out、Enabled-in、Disabled-out 和 Disabled-in。这些图像来自单个位图文件或一个 Bitmap 值，在这个位图值里，按顺序存储着几个图像。我们称这个 Bitmap 值为一个图像列表 (Imagelist)。这个图像列表里的图像的宽度必须相同，并用序号指定哪一图像用于按钮的什么状态，这意味着用户既可以将所有按钮状态指定为一个图像，也可以为按钮的每一种状态指定不同的图像。

用户还可以用参数<maskImage>为按钮图像指定一个黑白蒙版图像，如果没有指定蒙版文件，应将参数<maskImage>设为 undefined。

用户可以通过给属性.images 赋值来改变按钮的图像，如：

```
bm1 = render camera:$cam01 outputSize:[80,60]
foo.images = #(bm1, undefined, 1, 1, 1, 1)
```

在下面的例子中，按钮图像存储在文件 dcybt�ns.bmp 里，而蒙版文件存储在文件 dcymask.bmp 里，这两个文件里都包含 6 个图像。

```
checkbutton decay images:#("dcybt�ns.bmp", "dcymask.bmp", 6, 1, 4, 1, 4)
```

# 第 16 章 RcMenu (右键菜单)

MAXScript 提供了一系列的类和一些特别的语法让用户可以构造自己的鼠标右键菜单，并可以将它们加入到 3ds max 用户界面里去。用户可以通过右击鼠标来快速调出事先用 MAXScript 定制好的工具集。

在 MAXScript 里构造一个鼠标右键菜单，用 RcMenu 子句进行定义，然后用专用的函数进行注册。定义语法为：

```
rcmenu <var_name>(<rcmenu_body>)
```

参数说明：

- ◆ <var\_name>为一个变量名，系统自动将它设为全局变量，变量类型为 RcMenu，代表新构造的右键菜单。
- ◆ <rcmenu\_body>为一些子句序列，其必须用一对括号括起来，这些子句序列定义了要显示的菜单项及与用户进行交互的函数及事件处理程序。这些子句将在下一节详细介绍。

## 方法

下面的几个方法用于注册/取消注册右键菜单。

1. registerRightClickMenu <rcmenu>

注册指定右键菜单。

2. unRegisterRightClickMenu <rcmenu>

取消注册指定右键菜单。

3. unRegisterAllRightClickMenus()

取消注册所有右键菜单。

下面的脚本在右键菜单加入两个菜单项：Cast shadows 和 Receive shadows。这些菜单项只有在选择了一个场景对象后才可用。如果它们在可用情况下，菜单项会根据被选择对象的状态来调整自己的状态，而用户点击一个菜单项也会改变相对对象的属性。

```
rcmenu MyRCmenu
(
    menuItem mi_cs "Cast Shadows" checked:False
    menuItem mi_rs "Receive Shadows" checked:False
    --
    on MyRCmenu open do
        (local sel =(selection.count == 1)
    --
        --如果仅有一个对象被选择,启用本菜单项
        mi_cs.enabled = mi_rs.enabled = sel
```

```

--设置菜单项状态
if sel do
(   mi_cs.checked = $.castShadows
    mi_rs.checked = $.receiveShadows
)
)

--为菜单项建立事件处理程序
on mi_cs picked do $.castShadows =(not $.castShadows)
on mi_rs picked do $.receiveShadows =(not $.receiveShadows)
)

--注册右键菜单
registerRightClickMenu MyRCmenu

```

用户可以按自己的要求注册任意数量的 RcMenu，新注册的 RcMenu 里定义的新菜单项被加入到原右键菜单项的后面。如果一个同名的 RcMenu 被注册两次，新的 RcMenu 会覆盖旧的 RcMenu。如果在执行右键菜单时发生了运行错误，系统会在 Listener 窗口中显示出错误信息，但 RcMenu 不会被禁用。

## 16.1 RcMenu 子句

右键菜单定义中的<rcmenu\_body>由一系列的 RcMenu 子句组成，其语法定义如下：

<rcmenu\_body> ::= { <rcmenu\_clause> }+

其中<rcmenu\_clause>定义了右键菜单的三个组成部分：

- ◆ 局部变量、函数、结构定义。其作用域限于右键菜单之内。其生存期与右键菜单定义中全局变量<var\_name>相同。而全局变量<var\_name>在被取消注册或删除之前会一直存在。局部变量经常用于存储右键菜单的工作数据如拾取对象；
- ◆ 用户界面控件。如菜单项、分隔行和子菜单它们将被显示在右键菜单里；
- ◆ 事件处理程序。与特定用户界面项相连的特殊的函数，事件处理程序指定了当用户按下某一用户界面项时要作出的反应。用户的操作会产生一个命名的事件，如果编程者提供了一个事件处理程序，当事件发生时，系统就会调用事件处理程序。

局部变量的可见性和在外部程序代码中存取右键菜单内部局部变量的方法参见 Utility 相应章节。在 Utility 和 RcMenu 里，用户界面控件的名字都是代表用户界面的局部变量，并被用来联系与之相关的事件处理程序。但有一点区别读者应该注意：在 Utility 里控件名的作用域限于定义它的 Rollout 里，而 RcMenu 里用户界面控件的名字作用域在整个 RcMenu 定义体内，这意味着所有的用户界面项，即使是子菜单里的用户界面项，必须有一个惟一的名字。

<rcmenu\_clause>子句的定义为：

<rcmenu\_clause> ::= <local\_variable\_decl> |  
                   <local\_function\_decl> |

```
<local_struct_decl> |
<user_interface_item> |
<event_handler>
```

各项说明参见 15.5 节。

## 16.2 RcMenu 用户界面控件

共有三种用户界面控件可用来构造右键菜单，分别是：

- ◆ MenuItem
- ◆ Separator
- ◆ Submenu

下面分节详细阐述。

### 16.2.1 MenuItem (菜单项)

MenuItem 控件用来在 RcMenu 上加入一个可点取的菜单项。其语法为：

```
menuItem <name> <label> [ checked:<Boolean> ] \
[ enabled:<Boolean> ] [ filter:<function> ]
```

参数说明：

- ◆ name 是菜单项名字，用来联系事件处理程序；
- ◆ label 是显示在 RcMenu 中的字符串；
- ◆ checked: 指定是否在菜单项前面显示一个复选符号，如为 True，显示复选符号，否则不显示；
- ◆ enabled: 指定当 RcMenu 被打开时，该菜单项是否可用，如为 True 表示可用，否则不可用。
- ◆ filter: 为一个返回 Boolean 值的函数，当 RcMenu 被首次打开时，filter 函数被求值。如果 filter 函数返回 True，该菜单项被显示在 RcMenu 里，否则不显示该菜单项。

#### 属性

```
<menuItem>.text: String, 为参数 label: 指定的字符串  
<menuItem>.checked: Boolean  
<menuItem>.enabled
```

#### 事件

on <menuItem> picked do <expr>

当用户点击菜单项时被调用。

例如：

```

rcmenu RCmenuRenderable
(fn onlyOneSelected = selection.count == 1 --定义过滤函数
--在只有一个对象被选择时才显示右键菜单
menuItem mi_r "Renderable" filter: onlyOneSelected
on RCmenuRenderable open do      --定义事件处理程序 on rcmenu open
(if selection.count == 1 then --如果只有一个对象被选择
(mi_r.text = $.name + " | " + "Renderable" --改变菜单项
if isKindOf $ Shape then
$.renderable=$.renderable
mi_r.checked=$.renderable
)
)
--定义事件处理程序 on mi_r picked
on mi_r picked do $.renderable = not $.renderable
)
--
registerRightClickMenu RcmenuRenderable --注册右键菜单

```

### 16.2.2 Separator (分隔行)

Separator 控件用来在 RcMenu 上加入一个分隔行，经常被用来对 MenuItem 控件进行分组，其语法为：

separator <name> [filter:<function>]

参数说明：

- ◆ name 代表分隔行的一个局部变量；
- ◆ filter: 为一个返回 Boolean 值的函数，当 RcMenu 被首次打开时，filter 函数被求值。如果 filter 函数返回 True，该分隔行被显示在 RcMenu 里，否则不显示该分隔行。

例如：

```

rcmenu MyRCmenu
(
fn flt_objects =($ != undefined)--过滤对象
--
menuItem mi_cs "Cast Shadows" checked:False
menuItem mi_rs "Receive Shadows" checked:False
--
--仅在一个或多个对象被选择时才添加下面内容
separator sep1 filter:flt_objects
menuItem mi_st "Scale Twice" filter:flt_objects
menuItem mi_sh "Scale Half" filter:flt_objects
--
--下面可以定义事件处理程序...
)
registerRightClickMenu MyRcmenu --注册右键菜单

```

### 16.2.3 Submenu (子菜单)

Submenu 控件用来在 RcMenu 上放置一个菜单项，当鼠标放在它上面时，会打开一个子菜单，其语法为：

```
submenu <label> [ filter:<function> ](<submenu_body> )
```

其中<submenu\_body>也是由一些 Rcmenu 子句序列组成的，其语法定义如下：

```
<submenu_body> ::= { <rcmenu_clause> }+
```

参数说明：

- ◆ label 显示在 RcMenu 菜单上的字符串；
- ◆ filter:同 MenuItem。

例如：

```
rcmenu MyRcmenu
(
    fn flt_objects =($ != undefined)           --对象过滤
    fn flt_shapes =(isKindOf $ Shape)          --shape 对象过滤
    --
    menuItem mi_cs "Cast Shadows"      checked:False
    menuItem mi_rs "Receive Shadows"     checked:False
    --
    separator sep2 filter:flt_objects
    --
    subMenu "Modifiers" filter:flt_objects   --子菜单定义开始
    (--添加普通对象
        menuItem mi_bend "Bend"
        menuItem mi_twist "Twist"
    --
        --只添加应用于 shape 的修改器
        separator sep3      filter:flt_shapes
        subMenu   "Shape"   filter:flt_shapes   --嵌套子菜单定义开始
        (menuItem mi_extrude      "Extrude"
            menuItem mi_EditSpline "Edit Spline"
        )
    )
    --下面可以定义事件处理程序
    .....
)
registerRightClickMenu MyRcmenu           --注册右键菜单
```

# 第 17 章 宏脚本 (MacroScript)

## 17.1 定义宏脚本

宏脚本是与工具栏里某一按钮相连的脚本程序，当用户按下该工具栏按钮时，宏脚本被执行。宏脚本本质上是一段脚本代码，必须用 `macroScript()` 函数定义。宏脚本并不自动创建一个用户界面控件，用户必须按下面方法与工具栏连起来：在工具栏或选项卡上右击鼠标，在弹出菜单上选择 **Customize**，打开 **Customize User Interface** 对话框，在其中可进行设置。

宏脚本的定义语法为：

```
macroScript <name> [ category:<String> ] \
    [ buttonText:<String> ] \
    [ toolTip:<String> ] \
    [ icon:#(<String>, <index>)| icon:<String> ] \
    [ silentErrors:<Boolean> ] \
    (<macro_script_body>)
```

例如：

```
macroScript Free_Camera category: "Cameras" toolTip: "Free Camera"
    Icon:#("Cameras", 2)
(
    StartObjectCreation FreeCamera
)
macroScript Target_Camera category: "Cameras"
    tooltip: "Targeted Camera" Icon:#("Cameras", 1)
(
    StartObjectCreation TargetCamera
)
```

MAXScript 对 MacroScript 定义求值后，宏脚本定义会在 **Customize User Interface** 对话框里相应 **category** 的 **Action** 列表框中加入刚刚定义好的宏脚本，图 17-1 中显示了一个包含了前面例子中定义的两个宏脚本的 **Customize User Interface** 对话框。

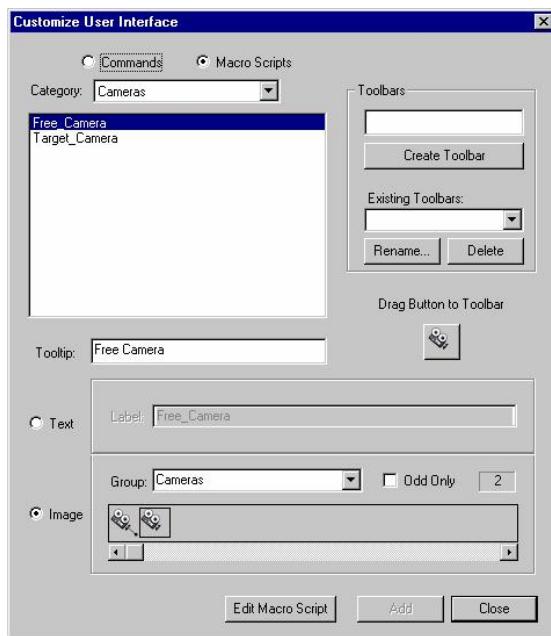


图 17-1 定义宏脚本 (MacroScript)

各参数说明如下：

- ◆ <name>是显示在 Customize User Interface 对话框中的名字，但 MacroScript 并不创建这个名字的变量，因为宏脚本是作为指针存储在文件里；
- ◆ category: 指定了宏脚本名字显示在 Customize User Interface 对话框的哪一个 Category 里。category: 参数的作用是帮助编程者将宏脚本划分成组。如果没有指定该参数，系统会自动将其指定为 unknown；
- ◆ toolTip: 指定工具栏按钮的提示字符，如果没有指定 toolTip 参数，就不会显示提示字符；
- ◆ buttonText: 指定显示在按钮上的文本，如果没有指定 buttonText: 参数，<name>参数指定的字符会用作显示在按钮上的文本；
- ◆ icon: 分别指定了图标位图文件和位图文件中哪一图像作为按钮图像。位图文件必须放在当前 3ds max 的 User Interface 路径里，图标位图文件名由一个“本名”，如“MyToolbar”，和一个后缀，如“\_24i.bmp”（指定单个图标的大小和图标位图文件类型）组成。icon: 变量中位图文件名<String>仅为本名而没有后缀。该本名被显示在 Customize User Interface 对话框的 Image Group 列表中。每个图标位图文件中可包含任意数量的单个图标。如果一个位图文件包含多个图标，变量<index>指定了要使用哪幅图标，最左端为第 1 幅。3ds max 内部图标没有存储在图标文件中，用户可以用一个空字符串（“”）来作为 icon: 变量。这样 icon: 变量既可以是一个二元数组，包含了图标位图文件的本名（数据类型为 String）和图标索引号（数据类型为 Integer），也可以只有一个指定本名的字符串，此时系统设定<index>为 1。

例如：

```

macroScript Box category: "Objects" tooltip: "Box"
    icon:#("standard", 1)--使用标准图标的第 1 个图标
(
    StartObjectCreation Box
)
macroScript Sphere category: "Objects" tooltip: "Sphere"
    icon:#("", 2)--使用 3ds max 内部图标的第 2 个图标
(
    StartObjectCreation Sphere
)
macroScript Cone category: "Objects" tooltip: "Cone"
    icon: "myicon" --使用图标文件 myicon 的第 1 个图标
(
    StartObjectCreation Cone
)

```

- ◆ **silentErrors:**指定在执行宏脚本时是否显示运行错误信息，如果为 True 表示不显示错误信息，默认值为 False。
- ◆ **<macro\_script\_body>:**为任何有效的 MAXScript 表达式，并且可包含全局变量、局部变量声明、函数和结构定义。**<macro\_script\_body>**中定义的局部变量、函数、结构的作用域仅限于宏脚本内。

在宏脚本内的局部变量与正常的局部变量有一点不同：正常的局部变量在自己的作用域内可见，其生存期为该作用域的一个执行周期；宏脚本内的局部变量同样在自己作用域内可见，但其生存期却是与它所在程序里顶级表达式的生存期相同。宏脚本的局部变量在用户首次运行宏脚本时被创建，并被赋予初始值 **undefined** 或用户指定的初始值。这些值被存储在一个单独的内存堆栈里。

因为宏脚本的名字（**.name** 属性）不会被用来创建一个变量，用户不能在宏脚本作用域之外存取其内部定义的局部变量。这样，如果在宏脚本内创建了一个 **Rollout**，那这个 **Rollout** 的事件处理程序可以存取定义在宏脚本内的局部变量，因为 **Rollout** 是在宏脚本的作用域之内，但是没有办法从另一个 **Utility** 或 **Listener** 窗口里存取定义在宏脚本内的局部变量。

当运行一个 **MacroScript** 定义后，系统返回一个整数，代表之前定义宏脚本的标识码 ID。MAXScript 内部把每个宏脚本的信息按数组形式组织，系统返回的这个标识码 ID 实际是一个指向这个数组的指针。

宏脚本程序存储在后缀为.mcr 的文本文件里，一个.mcr 文件里可以包含任意数量的宏脚本定义，系统为每个宏脚本定义分配一个指针指向定义开始的地方，当用户首次按下与脚本文件里某一个宏脚本相连的工具栏按钮后，脚本文件被调入系统并进行编译，用户还可以通过调用 **Macros.run()**方法来实现这一点。

共有五种方法供用户来定义宏脚本：

- ◆ 启动 3ds max 时系统会自动在路径.\UI 及其子路径下查找后缀为\*.mcr 的文件，并调入系统；
- ◆ 可以在脚本文件或 Startup.ms 文件里包含宏脚本定义；

- ◆ 如果用户在 Listener 窗口里对一个宏脚本定义求值，系统会自动创建一个包含该宏脚本定义的\*.mcr 文件，该文件被存储在路径.\UI\MacroScripts 里；
- ◆ MAXScript 还支持将文本拖拉至工具栏来创建宏脚本按钮。用户可以从任何文本窗口（如 Listener 窗口或 Editor 窗口）选择并拖拉文本至任何可见的工具栏上。当用户选择好文本并可以拖拉时，鼠标形式变成一个箭头加一个“+”号，当文本拖至工具栏上，松开鼠标后，会在该工具栏内创建一个宏脚本按钮，被拖拉的文本就是该按钮的宏脚本定义体。常用的方式是从 Listener 窗口的宏记录区域里选择一个刚刚记录好的事件序列文本，MAXScript 会自动将这些文本用一对括号括起来，放在一个宏脚本定义里，名字为 DragAndDrop\_MacroN，其中 N 为一个数字，以保证宏脚本名字的惟一性，而 category:参数为 DragAndDrop，toolTips:参数没有指定。和在 Listener 窗口里创建的宏脚本一样，系统同样会自动创建一个\*.mcr 文件来包含该宏脚本定义，该文件也被存储在路径.\UI\MacroScripts 里；
- ◆ 可以用 macros.new()方法来创建宏脚本，具体描述见下文：  
如果在移动或删除包含已装载宏脚本定义的文件后，再试图执行宏脚本，系统会给出一个错误信息。  
而如果用户编辑了包含宏脚本的文件后，注意要在存盘后对整个文件重新求值，以更新文件指针，否则会得到一个错误信息，提示当前装载的脚本定义与其文件不相匹配。

任何在 MAXScript 里定义的或从文本窗口拖拉文本至工具栏创建的宏脚本定义都有一个单独的.mcr 文件，存储在当前路径.\UI\MacroScripts 里，文件名的命名规则为：

<category\_name>-<macro\_name>.mcr

例如：

```
macroScript Macro12 category:  
--"DragAndDrop" 的名称为 DragAndDrop- Macro12.mcr  
macroScript Map_Updater category:  
--"NURBS" 的名称为 NURBS-Map_ Updater.mcr
```

MAXScript 提供了几个方法，让用户可以在脚本里存取、运行宏脚本。这些方法都有一个前缀：macros。下面分别列出这些方法：

1. macros.load [ <path\_name\_String> ]

装载路径.\UI 或指定路径<path\_name\_String>下的所有宏脚本 (.mcr) 文件。

2. macros.new <name\_String> <category\_String> <toolTip\_String> \

<buttonText\_String> <body\_String>

用指定名称和指定 category 创建一个新宏脚本。系统会自动创建一个新文件来包含这个宏脚本定义，这个文件存储在路径.\UI 里。返回值为一个用来标识宏脚本的惟一整数。

3. macros.run <category\_String> <name\_String>

macros.run <macro\_id\_Integer>

运行指定宏脚本。宏脚本既可以用名称和 category 来指定，也可以用标识码 ID 来指定。

4. macros.edit <category\_String> <name\_String>

```
macros.edit <macro_id_Integer>
```

在一个 Script Editor 窗口里打开包含指定宏脚本的脚本文件 (.mcr)。宏脚本既可以用名称和 category 来指定，也可以用标识码来指定。

例如：

```
macros.load "f:/gametools/macros"
macros.edit "objects" "box"
macros.run 132
macros.run "modifiers" "bend"
```

下面的宏脚本例子将活动视窗存储到一个位图文件里：

```
MacroScript GrabViewport category: "Tools" tooltip: "Grab Viewport"
(
MacroScript GrabViewport category: "Tools" tooltip: "Grab Viewport"
(
--功能：给活动视窗制作“快照”，并存储到位图里
--默认位图文件名格式为：
--VPGRAB_Max 文件名_视窗名_当前帧号，图像格式后缀
-----
--变量初始化
local grab_bmp          --存储快照的位图
local bmp_name            --位图文件名
local get_viewport_name   --视窗名
--保存 ActiveCamera、CurrentTime、MaxFileName 的变量
--local gac,gct,mfn
--
--宏脚本开始
grab_bmp = gw.getViewportDib()          --获取视窗位图
get_viewport_name = viewport.GetType() --获取视窗名
gvn = get_viewport_name as String      --将视窗名转换为字符串
gvn = subString gvn 6(gvn.count-5)    --裁剪字符串
if gvn == "camera" then               --如果省下字符串为 camera
  (gac = getActiveCamera()           --获取摄像机
  gvn = gac.name                    --获取摄像机名称
)
mfn = MaxFileName                --获取 max 文件名
if mfn == "" then                 --如果没有文件名
  mfn = "Untitled"                --使用 Untitled 作为文件名
else
  mfn = getFileNameFile mfn --将文件名中的.MAX 后缀去掉
gct = SliderTime as String        --获取当前时间帧
--
bmp_name = "VPGRAB_" + mfn + "_" + gvn + "_" + gct --合成输出文件名
--
--显示 file save 对话框
bmp_name = getSaveFileName caption: "Save Viewport to:" \
filename:bmp_name types: "BMP(*.bmp)|*.bmp|TIFF(*.tif)|*.tif| \
JPG(*.jpg)|*.jpg|TGA(*.tga)|*.tga|"
```

```

if bmp_name != undefined then      --如果用户确认或输入合法的文件名
  (grab_bmp.filename = bmp_name   --将用户输入文件名作为输出文件名
  save grab_bmp                  --保存位图
  display grab_bmp                --将位图显示在一个 VFB 里
)
)                                  --脚本结束

```

下面的脚本演示了如何在宏中使用 Rollout 和 Rollout 浮动窗口。

```

--功能: 直接在 RamPlayer 里播放渲染结果
--
MacroScript RAM_Render category: "Tools" tooltip: "Render to Ram"
(
--声明局部变量、定义函数
local get_names existFile
function get_names name a = append a name
function existFile fname =(getfiles fname).count != 0
--
--定义一个 Rollout
Rollout r_size "Render Parameters"
(local p = 95
  local p2 = p+78
  group "Time Output"
  (radiobuttons r_time columns:1 align:#left \
    labels:#( "Single", "Active Time Segment", "Range: " )
    Spinner nth "Every Nth Frame:" pos:[215,24] fieldwidth:50 \
      type:#Integer range:[0,10000,1] enabled:False
    Spinner r_from fieldwidth:60 pos:[75,56] type:#Integer \
      range:[0,10000,1] enabled:False
    Spinner r_to "To:" fieldwidth:60 pos:[152,56] \
      type:#Integer range:[0,10000,100] enabled:False
  )
  group "Render Size"
  (Spinner rw "Width" fieldwidth:60 pos:[15,p+08] \
    type:#Integer range:[0,10000,320]
    Spinner rh "Height" fieldwidth:60 pos:[12,p+32] \
    type:#Integer range:[0,10000,240]
    Spinner aspect "Aspect" fieldwidth:60 pos:[10,p+56] \
    type:#Float range:[0,20,1.0]
    button s160 "160x120" pos:[125,p+06] width:75 height:19
    button s256 "256x243" pos:[125,p+30] width:75 height:19
    button s320 "320x240" pos:[205,p+06] width:75 height:19
    button s512 "512x486" pos:[205,p+30] width:75 height:19
    button s640 "640x480" pos:[285,p+06] width:75 height:19
    button s720 "720x486" pos:[285,p+30] width:75 height:19
    button conf_render "Configure" \
    pos:[125,p+54] width:115 height:19
    button wipe "Purge Files" pos:[245,p+54] width:115 height:19
    button go "Render" pos:[125,p+78] width:235 height:19
  )
  label abt0 "Render to RAM" pos:[8,p2+31]
)

```

```

label abt1 "Version 0.2a" pos:[8,p2+46]
label abt2 "Alexander Espeschit Bicalho" pos:[225,p2+31]
label abt3 "abicalho@brasilmail.com" pos:[249,p2+46]

--
--定义 Rollout 事件处理程序
on wipe pressed do
(local tempfilename_a = (getdir #image)+ "\\\ramplayertemp_a.avi"
 local tempfilename_b = (getdir #image)+ "\\\ramplayertemp_b.avi"
 if existfile tempfilename_a then deletefile tempfilename_a
 if existfile tempfilename_b then deletefile tempfilename_b
)

--
on r_time changed state do
(case state of
(1: nth.enabled = r_from.enabled = r_to.enabled = False
2:( nth.enabled = True
r_from.enabled = r_to.enabled = False
)
3: nth.enabled = r_from.enabled = r_to.enabled = True
)
)

--
on s160 pressed do (rw.value=160; rh.value=120; aspect.value=1.0)
on s320 pressed do (rw.value=320; rh.value=240; aspect.value=1.0)
on s256 pressed do (rw.value=256; rh.value=243;
aspect.value=1.266)
on s512 pressed do (rw.value=512; rh.value=486;
aspect.value=1.266)
on s640 pressed do (rw.value=640; rh.value=480; aspect.value=1.0)
on s720 pressed do (rw.value=720; rh.value=486; aspect.value=0.9)
--

on conf_render pressed do(max render scene)
--

on go pressed do
(local tempfilename_a =(getdir #image)+ "\\\ramplayertemp_a.avi"
local tempfilename_b =(getdir #image)+ "\\\ramplayertemp_b.avi"
if existfile tempfilename_b then
(deletefile tempfilename_a
copyfile tempfilename_b tempfilename_a
tempfilename = tempfilename_b
)--if 结束
else
(if existfile tempfilename_a then
tempfilename = tempfilename_b
else
(tempfilename = tempfilename_a
tempfilename_b = ""
)
)
case r_time.state of

```

```

(1: render outputheight:rh.value outputwidth:rw.value \
    pixelaspect:aspect.value \
    outputfile:tempfilename vfb:off
2: render outputheight:rh.value outputwidth:rw.value \
    pixelaspect:aspect.value \
    outputfile:tempfilename vfb:off \
    nthframe:nth.value framerange:#active
3: render outputheight:rh.value outputwidth:rw.value \
    pixelaspect:aspect.value \
    outputfile:tempfilename vfb:off \
    nthframe:nth.value fromframe:r_from.value \
    toframe:r_to.value
)
ramplayer tempfilename_a tempfilename_b
closeRolloutFloater r_dialogue
)--on go pressed 事件结束
)--Rollout r_size 定义结束

--关闭旧的 Rollout 浮动窗口 (如果有的话)
try(closeRolloutFloater r_dialogue);catch()
--显示新的 Rollout 浮动窗口并想里面添加 Rollout
r_dialogue = newRolloutFloater "Render to RAM" 400 300
addRollout r_size r_dialogue
)

```

## 17.2 创建图标位图文件

在 17.1 节中讲到宏脚本定义中参数 icon: 可以为按钮指定图像，本节讲述如何创建一个图标位图文件。

一个图标位图文件可以包含任意数量的单个图标，它们一个接一个地排列在位图文件里。如果用户想创建图标位图文件，其必须符合以下条件：

- ◆ 必须以\*.bmp 为后缀；
- ◆ 图标位图文件实际上是一些文件组，每个文件组至少应有两个图像文件：一个是包含一个  $16 \times 15$  像素的图像，名字以“\_16i.bmp”结束，另一个是包含一个  $24 \times 24$  像素的图像，名字以“\_24i.bmp”结束；
- ◆ 所有图标位图文件里的图标必须有相同的大小；
- ◆ 每个图标位图文件组必须包含相似的图标，并按相同的顺序；
- ◆ 如果提供单色与 Windows 相容的蒙版文件，其名字必须以“\_16m.bmp”和“\_24m.bmp”结束，如 myicon\_16m.bmp 和 myicon\_24m.bmp；
- ◆ 如果提供 8 位 alpha 通道数据，并且如果通道数据没有进行预乘 (Premultiply) 处理时，文件名必须以“\_xxa.bmp”结束，否则，文件名必须以“\_xxp.bmp”结束，其中 xx 为 16 或 24，如 myicon\_16a.bmp；
- ◆ 如果提供了 mask 通道位图文件，系统会使用 mask 通道位图文件；如果没有提供

mask 通道位图文件，alpha 通道位图文件将被使用；如果 mask 和 alpha 通道位图文件都没有提供，左上角像素的颜色会被作为透明色，系统会自动为图像生成一个 mask。

如果用户希望在 MAXScript 里调用 render() 函数来生成一个图标位图文件，请确认命名符合前述规则，并把.bmp 文件存储在当前 3ds max 用户界面路径下。用户可以用下面的函数获取当前用户界面路径：

```
local ui_dir = cui.getDir()
```

# 第 18 章 脚本鼠标工具

## 18.1 脚本鼠标工具定义

脚本鼠标工具就是用脚本编制的程序，来响应 3ds max 视窗里鼠标点击动作。下面是一个简单的例子：

```
tool foo
(
    local b
    on mousePoint clickno do
        if clickno == 1 then b = box pos:WorldPoint else #stop
    on mouseMove clickno do b.pos = WorldPoint
)
```

上述例子允许用户在视窗里第一次点击鼠标左键时，在点击处创建一个 Box 类对象，还可以在创建 Box 后拖拉鼠标，此时 Box 对象会跟随鼠标移动，松开鼠标，鼠标工具停止。

鼠标工具的定义语法和 Utility 相似，它们都有自己的局部变量、函数、事件处理程序，但鼠标工具没有用户界面控件，其定义语法为：

```
tool <tool_name> [ prompt:<String> ] [ numPoints:<number> ] (<tool_body> )
```

各参数说明如下：

- ◆ tool\_name 是鼠标工具的名称，系统会自动用它创建一个变量，代表了鼠标工具。其作用域为当前 MAXScript 的作用域。这种变量的类名为 MouseTool；
- ◆ prompt: 为可选参数，指定当执行鼠标工具时显示在 3ds max 提示行的提示信息；
- ◆ numPoints: 为可选参数，指定鼠标工具可执行的最大次数。当达到 numPoints: 指定的值时，用户如果试图继续执行鼠标工具，系统会停止运行并返回#stop；
- ◆ <tool\_body> 是一个子句序列，定义了函数和事件处理程序，其必须用一对括号括起来，子句的定义见后文。

下面的方法用来调用鼠标工具：

```
startTool <tool_name> [ prompt:<String> ] [ snap:#3D|#2D ] \
[ numPoints:<number> ]
```

其中：

- ◆ tool\_name 表示调用鼠标工具的名称；
- ◆ prompt: 的意义同 tool 子句中参数 prompt:，如果在 startTool 中指定 prompt: 参数，会覆盖鼠标工具定义中指定的 prompt: 参数；

- ◆ snap:为可选参数，指定捕捉的极限设置，指定 snap:参数并不能打开捕捉设置。假如 snap 参数设为#2D，当用户在用户界面上打开了 3D 捕捉方式，但在使用鼠标工具时仍然只能使用 2D 捕捉；
- ◆ numPoints:意义同 tool 子句中 numPoints:，如果在 startTool 中指定 numPoints:参数，会覆盖鼠标工具定义中指定的 numPoints:参数。

方法 startTool()的返回值分两种情况：

- ◆ #stop 调用事件处理程序返回#stop，或 numPoint 值已到达；
- ◆ #abort 如果用户用鼠标右键或 Esc 键中断鼠标工具的执行。

可以用下面方法来中断一个正在执行的鼠标工具：

```
stopTool <tool_name>
```

## 18.2 MouseTool 子句

鼠标工具定义中<tool\_body>由一个 tool 子句序列组成，其语法定义如下：

```
<tool_body> ::= { <tool_clause> }+
```

其中 tool 子句可以为下面两种表达式：

- ◆ 局部变量，函数和结构定义，其作用域仅限于鼠标工具；
- ◆ 事件处理程序，定义了当用户点击、移动、右击鼠标，开始、结束鼠标工具等事件发生时希望执行的工作。

tool 子句一般的定义语法如下：

```
<tool_clause> ::= <local_variable_decl> |
                  <local_function_decl> |
                  <local_struct_decl> |
                  <event_handler>
```

在 tool 的函数和事件处理程序中，有 14 个预定义的局部变量供用户使用，这些变量在实际中非常有用：

局部变量名	数据类型	说明
ViewPoint	Point2	鼠标在视窗 pixel 坐标系中的当前位置
WorldPoint	Point3	当前鼠标位置投影到 World 坐标系活动网格上的坐标值
WorldDist	Point3	当前鼠标点击处至上次点击处的距离矢量，以 World 坐标系表示
WorldAngle	Point3	上次鼠标点击处与当前鼠标点击处距离矢量与 X、Y、Z 轴之间的角度，以 World 坐标系表示
GridAngle	Point3	上次点击处与当前鼠标点击处距离矢量与 X、Y、Z 轴之间的角度，以活动 Grid 坐标系表示
GridPoint	Point3	当前鼠标位置在活动 Grid 坐标系里的投影坐标
GridDist	Point3	当前鼠标点击处至上次点击处的距离矢量在活动 Grid 坐标系里的投影矢量

(续表)

局部变量名	数据类型	说明
ShiftKey	Boolean	如果 Shift 键被按下, 返回 True
ctrlkey		如果 Ctrl 键被按下, 返回 True
altkey		如果 Alt 键被按下, 返回 True
lButton		如果鼠标左键被按下, 返回 True
mButton		如果鼠标中键被按下, 返回 True
rButton		如果鼠标右键被按下, 返回 True
如果鼠标工具在一个 Level 5 级脚本插件下使用, 还有一个特殊的局部变量:		
nodeTM	Matrix3	用来存取当前创建节点对象的转换矩阵, 在当前 Grid 坐标系下

当在编写 SimpleObject 类脚本插件时, 用户会经常使用 GridPoint、GridDist 和 GridAngle 而不是 WorldPoint、WorldDist 和 WorldAngle, 因为在 SimpleObject 类脚本插件里, 对象创建是在活动的 Grid 坐标系进行的。对 GridDist 而言, 其 X、Y 分量分别为当前鼠标点击点与上次点击点在当前 Grid 平面坐标上的距离, 而 Z 分量则是当前点在 Screen 坐标系里的 Y 分量, 投影在以上次点击点为原点的 Z 向矢量(在 Grid 坐标系里)上的长度。也即是说, Z 值是一个在 Z 矢量上的投影高度。对非正交的视窗而言, 这正是用户所希望的, 但对正交视窗而言(在这些视窗里用户经常会在 XY 平面内进行鼠标拖拉), Z 值实际上就是 GridDist.y。

对 WorldDist 而言, X、Y、Z 分量定义与 GridDist 类似, 但其投影高度分量取决于视窗的构造平面。对 Top、Bottom 和非正交视窗, Z 分量为投影高度, 而对 Left 和 Right 视窗而言, X 分量为投影高度, 对 Front 和 Back 视窗而言, Y 分量为投影高度。

当用户在 3ds max 用户界面里创建一个 Node 对象时, 其 Local 坐标系的 Z 轴垂直于构造平面, 而当用户用 MAXScript 创建一个 Node 对象时, 其默认的 Local 坐标系的 Z 轴与 World 坐标系 Z 轴方向一致, 如果用 tool 创建 Node 对象时, 用户必须考虑到构造平面的方向, 最简便的办法就是在 in coardsys Grid 关联表达式里创建 Node 对象, 并且在创建对象的同时在 Grid 坐标系里指定其.pos 属性。如下例:

```

tool PointCreator
(local p, createpoint
--定义一个函数来创建对象, 并将坐标系设为'Grid', 便于将对象的 z 轴
--垂直于构造 Grid。
fn createpoint = in coardsys Grid p=point pos:GridPoint
--
--当按下鼠标时, 创建对象; 拖拉鼠标时, 移动对象, 松开鼠标时, 释放对象
--
on mousePoint clickno do
(if clickno == 1 --如果 clickno == 1, 表明第一次按下鼠标
    then createPoint()
    else if p != undefined do(p.pos=WorldPoint;p=undefined)
)
--
--如果 p != undefined, 正在移动一个前面创建的对象

```

```
--如果 p == undefined, 且鼠标左键被按下, 创建一个对象
on mouseMove clickno do
(if  p != undefined
    then p.pos=WorldPoint
    else if lbutton do createPoint()
)
)
--开始鼠标工具, 要退出鼠标工具, 右击鼠标
startTool PointCreator
```

### 事件处理程序

1. on start do <expr>

当鼠标工具开始时被调用。

2. on end do <expr>

当鼠标工具结束时被调用。

3. on freeMove do <expr>

当鼠标在第一次点击之前移动时被调用。

4. on mousePoint <arg> do <expr>

每次按下鼠标时被调用, 变元<arg>为鼠标 Click 事件的次数。

5. on mouseMove <arg> do <expr>

当鼠标在第一次按下后任意移动时被调用。

6. on mouseAbort <arg> do <expr>

当用户右击鼠标或按 Esc 键取消 tool 运行时被调用。

以上变元<arg>让用户知道已经按下鼠标的 Click 事件的次数。鼠标 Click 事件实际上一般发生在用户松开鼠标时(除第一次按下鼠标以外)。如果用户在开始鼠标工具后用鼠标执行下面的动作: 按下、拖拉、松开、移动, 然后再按下、拖拉、松开, 此间 mousePoint 事件会被调用三次, 时间分别在: 第一次按下、第一次松开, 第二次松开, <arg>变元分别为 1、2、3。鼠标 Click 事件计数器在调用完 mousePoint 事件处理程序之后会自动加 1。

这样在 mouseMove 和 mouseAbort 事件处理程序中的变元<arg>实际上是下一次 click 事件的次数。在刚才所说的操作中, 在第一次按下和第一次松开之间的拖拉鼠标调用 mouseMove 事件, 其<arg>变元值为 2。

下面的脚本演示了何时调用各种事件程序。用户可右击鼠标或按 Esc 键来退出 tool 的执行。

```
tool foo
( on freeMove do print "Free Moving"
  on mousePoint clickno do format "Point: %\n" clickno
  on mouseAbort clickno do format "Abort: %\n" clickno
  on mouseMove clickno do format "Moving: %\n" clickno
  on start do print "Starting"
  on end do print "Ending"
)
startTool foo prompt: "Hello!"
```

所有的事件处理程序都可以返回特殊值#stop 来停止鼠标工具的执行。在下面的例子中，首次按下鼠标时会创建 Box 类对象，当松开鼠标时则会停止鼠标工具的运行，如果用户想要再次使用该鼠标工具，必须用 startTool 重新开始。

```
on mousePoint clickno do
    if clickno == 1
        then b = box pos:WorldPoint
    else #stop
```

下面的例子能实现以下功能：在第一次按下鼠标时会在鼠标点创建三个 SpotLight 类对象；当按住鼠标键拖拉鼠标时，会在 X-Y 平面上拖拉三个对象；而松开鼠标左键后再移动鼠标时，会改变对象的 Z 坐标。

```
tool three_lights
(
    local key, fill, back, targ
--
on mousePoint click do coordsys Grid
(if click == 1 then --鼠标按下时创建key、back 和 fill 三个 light 对象
    (
        targ = targetobject pos:GridPoint
        key = targetspot pos:GridPoint name: "key" target:targ
        back = targetspot pos:GridPoint name: "back" target:targ
        fill = targetspot pos:GridPoint name: "fill" target:targ
    )
    if click == 3 then #stop
)
--
on mouseMove click do
(if click == 2 then --在x-y 平面拖拉对象
    (coordsys Grid key.pos = GridPoint
        coordsys targ back.pos = - key.pos
        local p = if shiftKey then 90 else -90
        coordsys targ fill.pos = key.pos *((eulerangles 0 0 p)as quat)
    )
else if click == 3 then --向上拖拉提升 light
    (in coordsys targ
        (local Z = GridDist.z
            key.pos.z = Z
            back.pos.z = Z * 1.5
            fill.pos.z = Z * 0.5
        )
    )
)
)
```

## 第19章 脚本插件

MAXScript 支持几个级别的脚本插件：

- ◆ Level 1：插件作为一个新类出现在用户界面里，但不允许场景里存在这种新类的实例。这类插件允许用户定义一个鼠标工具来在场景里创建其他类型的对象，但一旦用户完成其他类型对象的创建，由该插件所定义的类并没有实例留下。如：3ds max 里的 system 类（如 RingArray）。
- ◆ Level 2：扩展 3ds max 系统现有的插件。可以添加或取代现有插件命令面板的 Parameter 卷展栏。如：用户可以创建一个新的名为 glass 的材质来扩展 standardMaterial 并且提供一个自己定制的 Material Editor 对话框。
- ◆ Level 3：脚本插件由编程者指定一个惟一、固定的 ID 类码，这样就可以在场景里创建实例并且被存储在场景文件里。所有的 3ds max 系统插件都有一个惟一的 ID 类码，每次打开场景文件时，3ds max 使用该 ID 类码来将存储的场景对象和处理这些对象的程序代码联系起来。因此 ID 类码必须是惟一和固定的。但对上面所述的 Level 1 插件，MAXScript 分配一个临时的 ID 类码给这些插件，这些临时 ID 类码在 3ds max 执行过程中并不固定，这样系统就可以保证在场景里没有这些类的实例了。
- ◆ Level 4：脚本插件会定义一个或几个参数块（Parameter Block），在参数块里定义的参数可以直接用来设置动画并存储在场景文件里，当打开场景文件时，又可以被恢复。这些参数是固定的。
- ◆ Level 5：一个参数块可以和插件里定义的某一卷展栏相连接，这样可以为用户提供一个调整这些参数的用户界面。

脚本插件定义的通常格式为：

```
plugin <superclass> <varname> {keyword:val} (<plugin_body>)
```

各参数说明如下：

- ◆ <superclass> 指定新插件所属的超类名。下面为 3ds max 系统支持的插件超类：

Geometry	Helper	TextureMap
SimpleObject	Modifier	RenderEffect
Shape	SimpleMod	Atmospheric
Light	Material	

- ◆ <varname> 为一个包含新建插件类的全局变量的插件名。其命名方式和通常的插件类名一样（如 Box、Sphere），用户可以用它来创建插件类的实例；

◆ keyword:<val>为关键值参数序列，这些关键值参数可以包括下面几个：

#### 1. name:<String>

指定插件在 3ds max 界面上的名字，如果插件的.superclass 属性为 Geometry 或 Light，name:参数会显示在 Create 面板里该插件的按钮上；如果插件的.superclass 属性为 Modifier，name:参数则会显示在 Modify 面板里该插件的按钮上；而如果插件的.superclass 为 Material 或 TextureMap，name:参数会显示在 Material/Map Browser 对话框里。name:参数的默认值为插件的 varname 参数。

#### 2. category:<String>

指定插件按钮显示在哪一 Category 里。这一参数仅对显示在 Create 面板里的插件有效。默认值为 Standard。注意在 Create 面板里每一 Category 都只能容纳固定数量的按钮，如果插件定义完后没有正确地显示在 Create 面板里，应该为它指定另一 Category。

#### 3. classID:#(<Integer>,<Integer>)

指定插件的 ID 类码，如果用户想让插件创建对象并存储在场景文件里，必须指定 classID 参数，Level 1 级插件（如 System 类）因为仅创建其他类型的场景对象，可不指定 classID 参数。classID 参数由一对随机整数组成，这样才不致与其他插件的 classID 值发生冲突。

在 MAXScript 里有一个函数来生成一个新的 ID 类码：

`genClassID()`

它会产生一个格式如#(ox967ea231,0xbdb86ef)的随机数对，并显示在 Listener 窗口里，用户可以将它粘贴到插件定义里来作为新插件的 classID 参数。

#### 4. extends:<maxClass>

指定新建插件是否基于 3ds max 里某一现有的插件。这样可以用来增加一新类，其内部工作机制和被扩展的类完全一样，但其界面被扩展或添加内容。extend:参数指定的类必须和新插件有相同的.superclass 属性。目前因为 3ds max 的限制，某些插件还不能被扩展，尤其那些带有定制创建参数的插件（包括 targetLight、Camera 和所有 System 类插件）。如果某一插件被扩展后，新插件的卷展栏不显示，则说明该插件不能被扩展。

当用一个扩展插件创建一场景对象时，MAXScript 会在系统内部同时创建一个被扩展的插件对象实例，并将绝大多数用户与新插件类对象的交互传递给内部对象，这种作用机制称为 Delegation（委托），而 extends:参数指定的插件称为 Delegate。在 Track View 窗口里，Delegate 对象是可见的，而且用户可以在脚本里对 Delegate 对象进行存取操作。如果参数 replaceUI 设为 True，则在 Track View 视窗里不显示 Delegate 对象。

#### 5. replaceUI:<Boolean>

该参数仅在指定了 extends:情况下有意义。用来指定插件里定义的卷展栏是添加到被扩展插件的卷展栏之后还是取代被扩展插件。默认值为 False，表示新的卷展栏会添加到被扩展插件卷展栏的末尾。目前的 3ds max 的另一个缺陷是：如果被扩展插件的对象实例是在 Create 面板里被创建，那么用户只能替换 Modifier 面板里相应的界面，而 Create 面板里的界面总是会与插件定义的 Rollout 一起显示，即使 replaceUI 参数的值为 True。

#### 6. version:<Integer>

可选参数，指定插件的版本号。当更新插件定义时，要用到该值。默认值为1。

#### 7. invisible:<Boolean>

可选参数，指定是否在Create面板的Object Type卷展栏或Material/Map Browser里显示新插件。这对那些仅要创建对象组的一个构件或控制对象（如System类插件）时是很有用的。如果要隐藏插件，将本参数设为True。

#### 8. silentErrors:<Boolean>

可选参数，控制当执行脚本插件的事件处理程序时，如果发生运行异常，是否显示错误信息。如果本参数设为True，错误信息不被显示。这样当编程者不希望错误信息影响用户，可将本参数设为True。

- ◆ <plugin\_body>是一个用一对括号括起来的子句序列，这些子句定义了局部变量、函数、参数、用户界面控件和事件处理程序。

如果装载的一个场景文件中要用到尚没有被定义的插件，系统会在装载完成后显示一个Missing DLL对话框，指出找不到该插件的定义，然后系统会像处理其他找不到DLL定义的对象一样，用一个替代品来表示这些插件未定义的对象。和3ds max的系统C++插件一样，必须将脚本插件的定义放在3ds max的Plugins子目录下，或在装载场景文件之前在已运行过的脚本文件中被定义。

## 19.1 Plug-in子句

<plugin\_body>的定义语法如下：

<plugin\_body> ::= { <plugin\_clause> }+

<plugin\_clause>为<plugin\_body>的基本成员，其语法定义如下：

```
<plugin_clause> ::= <local_variable_decl> |
                    <local_function_decl> |
                    <local_struct_decl> |
                    <parameters> |
                    <tools> |
                    <Rollouts> |
                    <event_handler>
```

在<plugin\_body>里各种子句和Utility、鼠标工具很相似，但其中的<parameters>子句是Utility 和鼠标工具里所没有的，它定义了一些参数，可以与场景文件一起存储、恢复并在Track View视窗里可见。下面分节讨论Plug-in子句的五大内容。

### 19.1.1 局部变量声明和函数、结构定义

<local\_variable\_decl>（局部变量声明）

<local\_function\_decl>（局部函数定义）

## &lt;local\_struct\_decl&gt; (局部结构定义)

包括局部变量声明和函数、结构定义作用域限于插件内，其生存期与插件实例的生存期相同，其可见性参见 Utility 里有关章节。其语法定义分别为：

## ◆ 局部变量声明

```
<local_variable_decl> ::= local <decl> { , <decl> }
<decl> ::= <name> [ = <expr> ]
```

## ◆ 局部函数定义

```
<local_function_decl> ::= [ mapped ](function | fn)<name> { <argument> } = <expr>
```

## ◆ 局部结构定义

```
<local_struct_decl> ::= struct <name>(<member> { , <member> } )
```

全局变量声明不能作为<plugin\_clause>，但可以在事件处理程序里声明全局变量。为了确保某一变量名为全局变量，可以在插件定义之前进行声明。

插件局部变量隶属于每一插件对象，每一新的插件对象都有它自己的局部变量存储区域，当用户在插件函数或事件处理程序里引用这些局部变量时，系统会在当前活动对象的局部变量存储区域里存取局部变量。用户可以为局部变量赋初始值，当包含插件对象的场景文件在存盘后被重新打开时，插件的局部变量会按指定初始值被重新初始化。用户也可以通过设定插件的 Create 事件处理程序来达到上述相同目的。

插件局部变量可以作为插件对象的属性来从外部进行存取，但如果插件里有与局部变量同名的参数或通用属性（如.pos），按上面存取方法得到的值会是同名参数或通用属性的值，当存取一个插件对象的属性时，MAXScript 会首先在通用属性（如：.pos、.scale）里查找，然后在<parameters>块里查找，最后是在局部变量里查找，一旦查找到一个与之相匹配的名字，就停止查找。

在所有脚本插件里有三个预先定义的局部变量。

## 1. version Integer

返回插件对象的版本值，包含插件定义里指定的版本号。

## 2. this

返回当前选中脚本插件的实例。变量 this 总是包含与当前插件对象相应的 MAXScript 值，使用它可确保在脚本插件代码里对新对象的参数进行存取操作，而不是对与参数同名的局部变量或全局变量进行存取操作，如果用户想把插件存储到一个变量里，可以用下面语句：

```
a=this
```

## 3. delegate

返回由参数 extends 指定类型的对象实例。变量 delegate 包含了与被扩展的插件实例相对应的 MAXScript 值。如果要对被扩展插件实例的通用属性或.subAnim 属性进行存取，必须通过 delegate 变量，例如：

```
delegate.width=thisWidth
```

如果插件定义里没有指定 `extends` 参数, `delegate` 变量返回值 `undefined`。

### 19.1.2 参数块`<parameters>`

参数块定义了插件的参数, 这些参数和局部变量类似, 又有下面的区别:

- ◆ 可以直接对这些参数设置动画;
- ◆ 它们是存储在场景文件里的;
- ◆ 它们可以在重新装载场景文件时被恢复。

用户可以将`<parameter>`每一参数与插件里某一卷展栏指定的界面控件相联系, 参数和与之相联的界面控件如 Spinner 或 Checkbox 就“绑定”在一起了, 其中一方的更新都会自动更新对方的值, 这样用户就不必为界面控件的 `changed` 事件准备事件处理程序了。

用户可以在一个脚本插件里定义一个或多个`<parameters>`, 一个`<parameters>`块为插件定义一套参数, 其定义格式如下:

```
parameters <name> [type:#class] [Rollout:<name>]
(
    { <param_defs> }+
    { <event_handler> }
)
```

各参数说明如下:

- ◆ `<name>`用来命名参数块, MAXScript 用它来识别不同的参数块, 并在装载场景文件时把插件对象与其参数正确地链接。当用户对插件定义修改后, 如果别的场景文件中仍然有按旧插件定义创建的对象时, 这一点非常重要。在装载旧版对象时, MAXScript 会尝试着将它按新的参数块定义转换成新版的对象, 为了正确地做到这一点, 参数块里每一参数都必须有一个固定的名字。所以如果在别的场景文件中尚存有旧版插件生成的对象时, 不要随意改变参数块的`.name` 属性, 否则它们将不能正确地被装载;
- ◆ `type:#class` 指定参数块里的参数为`#class` 型参数, 也就是每一个插件实例都有一份全部参数的副本。3ds max 系统里 Primitive Geometry 类对象的创建参数和 `type_in` 类参数就是`#class` 型参数的典型例子;
- ◆ `Rollout:<name>`指定`<plugin_body>`里定义的某一卷展栏与参数块绑定, 用它来作为该参数块的用户界面卷展栏。目前 MAXScript 的一个局限为每一参数块只能与一个单独的卷展栏相连, 而每一个卷展栏只能与一个参数块相连。这意味着在插件用户界面里的每一个卷展栏都必须有它自己的参数块。
- ◆ `<param_defs>`定义了参数块里每个参数, 其格式为:

```
<name> type:<#name> [tabSize:<Integer>] \
    [tabSizeVariable:<Boolean>] \
```

```
[default:<operand>] [animatable:<Boolean>] \
[subAnim:<Boolean>] [ui:<ui_def>]
```

<param\_defs>的参数说明如下：

- <name>参数名。和参数块的 name 参数一样，如果在有的场景文件中包含有旧版插件创建的对象时，不要随意改变或重复使用参数名。
- type:<#name>指定了参数数据类型，可供选择的参数数据类型如下（标记为“可动画”的类型表示可设置动画）：

#Float: 可动画	#Filename
#Integer: 可动画	#ColorChannel: 可动画
#Color: 可动画	#Time: 可动画
#Point3: 可动画	#RadiobtnIndex
#Boolean: 可动画	#Material
#Angle: 可动画	#Texturemap
#Percent: 可动画	#Bitmap
#WorldUnits: 可动画	#Node
#String	

或者为下面类型的数组，其元素由上面类型的数据组成：

#FloatTab	#colorChannelTab
#intTab	#timeTab
#colorTab	#radiobtnIndexTab
#Point3Tab	#materialTab
#boolTab	#texturemapTab
#angleTab	#bitmapTab
#percentTab	#nodeTab
#WorldUnitsTab	#maxObjectTab
#StringTab	#WorldUnits
#filenameTab	

#Worldunits 和#WorldunitsTab 类型参数表示参数值为一个 World 坐标系下的距离值，实际数据类型为 Float，如：

```
parameters main Rollout:params
(
    height type:#WorldUnits ui:heightSpin
)
```

**注意** 这样设定不会自动将由参数 height “绑定”的界面控件 heightSpin 设为 type:#WorldUnits，用户必须另外在 params Rollout 的 Spinner 控件定义中进行指定。

#node 和#nodeTab 用来存储场景对象的数据，存储在这些参数里的对象不能进行循环

引用。比如：如果用户创建了一个脚本 Helper 插件，并将该 Helper 作为一个 Camera 对象的目标（Target）对象或父（Parent）对象，就不能再将 Camera 对象存储在#node 和#nodeTab 参数里。同样，如果用户创建了一个 Modifier 或 Material 插件，并将一个对象存储在#node 和#nodeTab 参数里，就不能再将该 Modifier 或 Material 赋给#node 和#nodeTab 参数里的对象了。

各种 xxTab 类型参数允许用户在参数里存储一个相应类型的数组，数组的初始长度由可选参数 tabSize 指定。如果指定可选参数 tabSizeVariable 为 True，当用户将一个索引号超过现有数组长度的元素赋给数组时，数组的长度会自动扩充。例如：

```
parameters main Rollout:params
(
    mapAmounts type:#FloatTab tabSize:4 tabSizeVariable:True \
        ui:(map1Amount, map2Amount, map3Amount)
)
```

上例定义了一个初始长度为 4 的参数 mapAmounts，其元素数据类型为 Float。用户可以为每一元素指定单独的 Rollout 控件与之绑定。当在插件程序里进行存取时，这些参数的存取方式和数组一样，如：

```
a = mapAmounts[i]
```

如果指定 tabsizeVariable:True，下面的语句会将数组 mapAmounts 的长度扩充为 10：

```
mapAmounts[10] = a
```

用户也可以通过给数组型参数的.count 属性赋一个新值来增减数组的长度。

当数组元素的数据类型为某些类型（如 FloatTab、intTab 和 percentTab 等）时，数组的每一元素都是独立的，并可以制作动画。除非用户指定 animatable:False。用户可以通过 controller 属性存取每一元素的控制器。例如：

```
c = mapAmounts[i].controller
```

- default:<operand>指定参数的初始值，<operand>也可以为表达式，但它的返回值必须为指定的数据类型或可以由 MAXScript 转换成指定的数据类型。
- animatable:<Boolean>指定参数是否可用来设置动画，这样参数就可以在 Track View 视窗中看到。只有那些标记为“可动画”的参数才可以设置动画，本参数默认值为 True。
- subAnim:<Boolean>仅适用于 3ds max 对象类型#node、#material、#textureMap 和 #maxObject。如果该参数设为 True，这些 3ds max 对象会在 Track View 窗口里的 object 轨迹中出现。默认值为 False。
- ui:<ui\_def>指定与参数相连接的界面控件，该界面控件位于参数 Rollout:指定的 Rollout 里。一旦将参数和界面控件相连，对参数或控件的任意一方的改变都会自动刷新对方的值。在下面例子中，名为 main 的参数块与名为 params 的卷展栏相连，而参数块中的参数 height、spread 分别与卷展栏中的 Spinner 类控件 height 和 spread 绑定：

```

parameters main Rollout:params
(key      type:#node subAnim:True
 fill     type:#node subAnim:True
 back    type:#node subAnim:True
 height   type:#Float animatable:True default:10 ui:height
 spread   type:#Float animatable:True default:10 ui:spread
 on height set val do
(key.pos.z = val
 back.pos.z = val * 1.5
 fill.pos.z = val * 0.5
)
)
Rollout params "Light Parameters"
(Spinner height "Height"
 Spinner spread "Spread"
)

```

如果某一参数被用来设置动画，当 Time Slider 刚好位于某一关键帧位置时，Spinner 控件会以红框标出，正如 3ds max 系统插件一样。能与某些特定参数类型相绑定的界面控件的组合见下表：

参数类型	可用 Rollout 控件类型
#Integer	Spinner、Slider、RadioButtons、Checkbox、Checkbutton
#Float	Spinner、Slider
#time	Spinner、Slider
#color	Colorpicker
#angle	Spinner、Slider
#percent	Spinner、Slider
#colorChannel	Spinner、Slider
#Boolean	Checkbox、Checkbutton
#node	PickButton
#textureMap	MapButton
#material	MaterialButton
#WorldUnits	Spinner, Slider

◆ <event\_handler>参数事件处理程序。与参数相关的事件处理程序有两个：

1. on <name> set <arg> do <expr>

set 事件适用于参数块里所有参数。当参数赋值时被调用（可以是改变与之绑定的界面控件的值或直接在 MAXScript 里给参数赋值）。一般来说，应该给参数设置 set 事件，而不是给与之绑定的控件设置 changed 事件。在上面例子中，当参数 height 值改变时，其 set 事件会刷新对象的.pos 属性。

如果一个参数被设置动画，并设置了 set 事件，当 3ds max 的时间改变（即拖拉 Time Slider 或在一次动画渲染内），set 事件也会被调用。

2. no <name> get <arg> do <expr>

get 事件适用于参数块里所有参数。当读取参数值时会被调用：当界面刷新、脚本读取

参数、视窗被重画或渲染事件发生时都会调用参数的 get 事件。在上面几种情况下，设置动画的参数值会被读取一次或多次。get 事件调用时，需要传递一个变元<arg>，其值为参数的当前值。变量 currentTime 为参数被读取时的时间，这一时间并不一定就是当前 Time Slider 显示的时间。如果执行一个 get 事件，必须返回一个值，该值会作为参数的新值，如果你不想改变该原有参数值，而仅仅是想监视读取过程，可以直接返回变元<arg>。

### 19.1.3 鼠标工具<tools>

定义鼠标工具用来执行一系列基于鼠标在 3ds max 视窗里 Click 事件的动作。在插件里虽然可以定义任意数量的鼠标工具，并在需要时使用它们，但一般我们仅定义一个名字为 create 的鼠标工具。这个鼠标工具用于在 Create 面板下自动被调用来创建一个脚本插件对象。有关 tools 的定义详见第 18 章。

可以创建场景对象的脚本插件必须定义一个 create tool，对那些扩展现有插件的脚本插件 create tool 会覆盖插件.delegate 属性指定类的 create tool。当 create tool 被执行期间，Animate 按钮会被自动关闭。

在 create tool 内部，有一个预先定义的局部变量 nodeTM。该变量数据类型为 Matrix3，其值为当前活动的 Grid 坐标系。该变量允许 create tool 为当前正被创建的对象设置其.transform 属性。同时在 create tool 内，用户应该使用局部变量 GridXXX，而不应该是 WorldXXX，因为创建对象是在 Grid 坐标系下进行的。

### 19.1.4 定制用户界面卷展栏<Rollouts>

用来为脚本插件定制用户界面卷展栏，这些卷展栏的定义语法与 Utility 的语法基本相同。对那些扩展现有插件的脚本插件，Rollout 可以添加到现有插件的 Rollout 后面，也可以取代它们。有关卷展栏的定义详见第 15 章。

和 Utility 不同，脚本插件里的卷展栏会根据插件类型自动在合适的位置打开（如在命令面板或 Material Editor）。

**注意** 脚本插件 Rollout 定义的局部变量的生存期仅限于 Rollout 被打开期间，如果用户想存取当前编辑插件对象的 Spinner 或其他界面控件的属性，应在 on <Rollout> open 事件处理程序里进行。

### 19.1.5 事件处理程序<event\_handler>

和鼠标工具一样，脚本插件也可以响应用户的某些操作，并执行一个事件处理程序。所有类型的脚本插件都支持下面的事件，用户可以使用它们来执行一些特定的初始化工作，如加载局部变量或初始化插件 delegate 对象的属性。

#### 1. on create do <expr>

当一个插件对象被创建时被调用。在 create 事件被执行期间，Animate 按钮会自动关闭。

#### 2. on load do <expr>

当从场景文件里加载插件对象时被调用。在 load 事件被执行期间，Animate 按钮会自

动关闭。

### 3. on update do <expr>

当用户定义了一个脚本插件，并在场景里创建了一个插件实例，然后对脚本插件进行修改并重新加载插件定义，此时场景里每一插件对象都会调用脚本插件定义里的 update 事件。详见 19.3 节。

事件处理程序的运行异常会导致其失效，直到被重新定义。当某一事件处理程序失效时，所有的事件处理程序都不能被调用。用户必须更正错误后对插件定义重新求值才能重新使用所有的事件处理程序。

## 19.2 脚本插件方法

方法 addpluginRollouts()可在插件的 create tool 里使用，特别是对那些 Level 1 的插件，这些插件并不在场景里创建它们本类的对象，而是在 create tool 里创建其他类的对象。这个方法可以在 create 面板里为指定对象安装其定制的 Rollout（特别是那些在 create tool 里创建的对象），这样对象的参数可以直接在创建过程中进行调整。其格式为：

`addPluginRollouts <obj>`

下面的例子定义了两个可编辑的类：一个是类型为 Helper 的 LightMaster 用来作为 Light 系统的主对象，另一个是 Light 类型的 threeLights 用来创建一个 Light 系统。在用 threeLights 类创建 Light 系统的同时也创建了一个 lightMaster 类对象，并将它的 key、back、fill 参数分别设为三个 Light 对象。

```
plugin helper lightMaster
    name: "Light Master"
    classID:#(12345,54321)
    extends:Dummy
    replaceUI:True
    invisible:True
(
parameters main Rollout:params
(
key      type:#node subAnim:True
fill     type:#node subAnim:True
back     type:#node subAnim:True
height   type:#WorldUnits default:0 ui:height
spread   type:#WorldUnits default:0 ui:spread
angle    type:#angle      default:90 ui:angle
--
--因为在创建对象之前 Light 类对象不存在，所以要先检测变量
on height set val do if key != undefined then coordsys key.target
(
key.pos.z = val
    back.pos.z = val * 1.5
    fill.pos.z = val * 0.5
```

```

)
-- on spread set val do if key != undefined then coorsys key.target
(
--计算 Key 对象在 XY 平面的法线矢量，存在变量 kuv 里
--并重新计算 key 对象和 back 对象的.pos 属性
local kuv = normalize([key.pos.x, key.pos.y, 0])
key.pos = [kuv.x * spread, kuv.y * spread, height]
back.pos = [kuv.x * -spread, kuv.y * -spread, height * 1.5]
--将 fill 对象的位置置于 key 对象之下，并绕 key.target 对象旋转
fill.pos = [kuv.x * spread, kuv.y * spread, height * 0.5]
about key.target rotate fill angle z_axis
)

-- on angle set val do if key != undefined then coorsys key.target
(
fill.pos = [key.pos.x, key.pos.y, height * 0.5]
about key.target rotate fill angle z_axis
)
)

Rollout params "Light Parameters"
(
Spinner height "Height" type:#WorldUnits range:[-1e32, 1e32, 0]
Spinner spread "Spread" type:#WorldUnits range:[0, 1e32, 0]
Spinner angle "Angle" range:[-180, 180, 30]
-- on params open do flagForeGround #(key, back, fill)True
on params close do
if key != undefined and back != undefined and fill != undefined then
    flagForeGround #(key, back, fill)False
)
)

plugin light threeLights name: "Three Lights"
(
local master
--
tool create
(
on mousePoint click do
(
if click == 1 then --创建 key、back、fill 三个 light 类对象
(
coorsys Grid
(
master = lightMaster pos:GridPoint
master.dummy.boxsize = [10,10,10]
in master
(

```

```

local targObj=targetobject pos:GridPoint
    master.key =  targetspot pos:GridPoint name: "key" \
                            target:targObj
    master.back = targetspot pos:GridPoint name: "back" \
                            target:targObj
    master.fill = targetspot pos:GridPoint name: "fill" \
                            target:targObj
)
)
addPluginRollouts master --安装 master 对象的 Rollout
)
if click == 3 then
(
select master
    master = undefined
    #stop
)
)
-- on mouseMove click do
(
if click == 2 then --如果鼠标拖拉
(
master.key.pos = WorldPoint
    master.spread = distance master.key master.key.target
)
else if click == 3 then --向上提升 lights
    master.height = GridDist.z
)
-- on mouseAbort click do
(
if master != undefined then
(
if master.fill != undefined then delete master.fill
    if master.back != undefined then delete master.back
    if master.key != undefined then delete master.key
    delete master
)
)
)
)
)

```

### 19.3 脚本插件的更新

**MAXScript** 允许用户在实际工作中重新定义脚本插件，并且会自动将当前场景中及之后装载的场景文件中那些按旧定义创建的插件对象按新的定义进行更新。这种重新定义插件的能力被称为“规划移植”。**MAXScript** 主要通过新旧脚本插件定义中的局部变量、参数

块和参数的匹配来完成这项工作：系统保留新旧定义中同名的局部变量、参数块和参数，去掉那些没有出现在新定义中的局部变量、参数块和参数，并创建那些没有出现在旧定义中的局部变量、参数块和参数，且赋给它们以默认值。

对于参数类型的转换有下面几点限制：

- ◆ #instance 类型的参数块不能变成#class 类型的参数块，反之亦然；
- ◆ 参数不能在重新定义时改变数据类型；
- ◆ 参数不能在重新定义时从一个参数块移至另一个参数块；
- ◆ MAXScript 只有在新旧定义中有相同的可选参数 classID 时才能进行正确的更新。

用户可以在插件定义中指定一个 update 事件处理程序，这样当插件对象被更新时就会调用它。为了使其更有效地工作，MAXScript 在插件定义中设置了 version:参数，任一插件对象都可以通过预定义变量 version: 来获取其版本信息。当旧插件对象通过“规划移植”机制进行更新时，仍然使用旧的版本信息，但一旦更新工作完成，version 变量的值马上变为新定义中指定的版本值。如：

```
plugin helper lightMaster
    name: "Light Master"
    classID:#(12345, 54321)
    extends:Dummy
    invisible:True
    replaceUI:True
    version:3
(
    parameters main Rollout:params
    ...
    on update do
    (
        if version == 1 then ...
    )
    ...
)
```

**注意** 仅当用户想在上述系统自动更新机制之外还要进行一些特别的转换时才需要在脚本插件定义中定制自己的 update 事件。

最后，如果在当前场景或其他场景文件中尚没有创建任何插件对象，就可以简单地删除插件定义，并且可以对插件定义做任何修改。上述版本更新仅在用户希望插件同时兼容那些旧版插件对象时才有用。

## 19.4 Geometry（几何体）类脚本插件

插件定义中 superclass 参数为 Geometry 的插件就是几何体插件，几何体插件必须有 create tool，除非脚本插件是不可见、临时的或扩展另一插件。如果新插件是为了扩展另一

插件，而又指定了自己的 create tool，会覆盖其 Delegate 对象的 create tool。

例如：

```
plugin geometry foo_plugin_def name: "FooBar"
category: "Scripted Primitives"
(local boxes, clickAt
    tool create
(on mousePoint click do
    (clickAt = WorldPoint
        boxes = for i in 1 to 10 collect box pos:([i*20,0,0] + clickAt)
        #stop
    )
)
-- 
    Rollout frab "Parameters"
(Spinner frab "Frabulate" range:[-1000,1000,20]
    on frab changed val do
        for i in 1 to 10 do boxes[i].pos = [i*val,0,0] + clickAt
    )
)
```

上例在 Create 面板的 Geometry 里加入一个名为 FooBar 的按钮。鼠标按下 FooBar 按钮会进入 mouse command 模式，当用户点击视窗时，会创建一排共 10 个 Box 类对象，同时向命令面板添加一个名为 Parameters 的卷展栏，用来调整创建参数。这个插件与 System 类对象如 RingArray 非常相似（除了其处在 Geometry 选项卡以外）。这是一种最简单的脚本插件：既没有特别的场景对象，也没有可存储的参数。

在 tool 事件处理程序中，用户可以根据需要设置插件 delegate 的属性，如下例：

```
plugin geometry Cuboid
name: "Cuboid"
classID:#(0x133067, 0x54374)
category: "Scripted Primitives"
extends:Box
(fn fmax val1 val2 = if val1 > val2 then val1 else val2
    tool create
(on mousePoint click do
    case click of
        (1: nodeTM.translation = GridPoint
        2: #stop
    )
)
-- 
on mouseMove click do
    if click == 2 then
        delegate.width = delegate.length =
        delegate.height = 2 * fmax(abs GridDist.x)(abs GridDist.y)
)
```

## 19.5 SimpleObject 类脚本插件

SimpleObject 对象也是 Geometry 类对象，如 Box、Sphere、Cone 等。但这种对象是用 tri\_Mesh（三角面片）来定义的。定义一个 SimpleObject 类脚本插件时，用户要编写一个名为 buildMesh 的事件处理程序，3ds max 会自动处理场景显示、重叠测试、渲染、修改器适配等问题。一个 SimpleObject 类脚本插件必须声明其`<superclass>`为 SimpleObject。

SimpleObject 类脚本插件需要定义一个 create tool。

在 SimpleObject 插件定义中不能包含诸如创建场景对象或建立修改器堆栈等与场景有关的操作，否则会导致系统陷入混乱。在 SimpleObject 插件定义中仅包含 create tool 事件和 buildMesh 事件，前者用来调整参数，后者用来构造 Mesh 对象。

例如：

```
plugin simpleObject tower_plugin_def
    name: "Tower"
    classID:#(145345,543211)
    category: "Scripted Primitives"
(parameters main Rollout:params
    (height type:#WorldUnits ui:height default:0
        width type:#WorldUnits ui:width default:0
        depth type:#WorldUnits ui:depth default:0
    )
    --
    Rollout params "Two Faces Parameters"
    (Spinner height "Height" type:#Worldunits range:[-1000,1000,0]
        Spinner width "Width" type:#Worldunits range:[-1000,1000,0]
        Spinner depth "Depth" type:#Worldunits range:[-1000,1000,0]
    )
    --
    on buildMesh do
        (setMesh mesh \
            verts:#([0,0,0],[width,0,0],[width,depth,0],[0,depth,0])\
            faces:#([3,2,1], [1,4,3])
            extrudeFace mesh #(1,2)(height * 0.5)40 dir:#common
            extrudeFace mesh #(1,2)(height * 0.5)50 dir:#common
        )
    --
    tool create
    (on mousePoint click do
        case click of
            (1: nodeTM.translation = GridPoint
            3: #stop
            )
        on mouseMove click do
            case click of
                (2:(width = GridDist.x; depth = GridDist.y)
```

```
    3: height = GridDist.z  
      )  
    )
```

上例中定义了一个新的 Geometry 类对象 Tower。鼠标的第一次点击设置对象的基点，第一次拖拉设置对象的 Width 和 Depth 参数，第二次拖拉设置对象的 Height 参数。新插件包含三个可设置动画的参数 Height、Width 和 Depth，它们一起设置了对象的基本范围。插件定义中关键部分为 create tool 和 on buildMesh 事件处理程序，它们按一种相当简单的方式一起工作：create tool 设置了对象的参数值，而 on buildMesh 事件则按参数值构造一个 Mesh 对象。

在 create tool 里可以通过预定义局部变量 nodeTM 来存取新创建对象的位置。在本例中，变量 nodeTM 的平移分量 (.translation) 被设为第一次鼠标按下时的位置。在 on mouseMove 事件中，click2 和 click3 事件分别设置对象的 Depth 和 Height 参数。

在 on buildMesh 事件处理程序中，构造了一个 Mesh 对象，存储在预定义变量 mesh 里，每调用一次 buildMesh 事件，都会创建一个新的 Mesh 对象。在上例中，Mesh 对象最开始被设为一个简单的双面矩形平面（通过方法 setMesh()），然后被伸展成一个简单的塔状对象。

SimpleObject 插件有一个预定义局部变量 mesh，其数据类型为 triMesh，包含了新创建的 Mesh 对象。变量 mesh 可以在任何插件的事件处理程序里进行存取操作，但仅能在 on buildMesh 里被建立。新创建的 Mesh 对象必须是合法的，否则会导致 3ds max 系统的崩溃。Mesh 对象的所有面 (Face)、顶点 (vertex)、材质 ID (Material ID) 和纹理顶点数组 (texture vertex array) 都必须在 on buildMesh 事件中指定。用户既可以用 TriMesh 方法修改现有的 TriMesh 值，也可以构造一个新的 TriMesh 值，并将它赋给变量 mesh。

SimpleObject 类脚本插件还有三个特殊的事件：

1. on OKtoDisplay do <Boolean expr>

如果执行 OKtoDisplay 事件，`<Boolean_expr>`必须返回 True 或 False，决定了是否可以正常显示当前的 Mesh 对象。当 Mesh 对象的某些参数设置处在 Degenerate 状态，因而不需要在视窗里显示时 OKtoDisplay 事件非常有用。默认的执行结果为 True。注意：空 Mesh 对象也是可以显示的。

2. on hasUVW do <Boolean expr>

如果执行 hasUVW 事件，`<Boolean_expr>`必须返回 True 或 False，决定了是否显示 Mesh 对象的 UVW 坐标。在很多标准几何体对象的创建卷展栏里，都提供了一个名为 Generate Mapping Coords 的 Checkbox 类控件，来控制是否生成 UVW 坐标，hasUVW 事件处理程序必须返回该 Checkbox 控件的状态。默认的返回值可能为 True 或 False，取决于 Mesh 对象是否有指定纹理顶点。如果有指定，仅支持一个 Map 通道。

3. on setGenUVW <onOff> do <expr>

`setGenUVW` 事件当 3ds max 希望插件自动生成贴图坐标时被调用，就像第一次渲染一个应用了材质、但没有贴图坐标的对象时发生的一样。变元`<onOff>`是一个布尔值，如果为 True 会打开贴图坐标，否则会关闭贴图坐标。如下例：

```

on setGenUVW onOff do
(
    genMapCoords = onOff
    if genMapCoords and gen_mapping_coords()
)

```

上例中函数 `gen_mapping_coords()` 可以为 Mesh 对象生成一个贴图坐标, 如果生成成功, 返回 `True`, 否则返回 `False`。

在下面的例子中, 首先建立两个其他类的临时对象, 然后对两个临时对象应用 Mesh 布尔运算来生成最后的 Mesh 对象。

```

plugin simpleObject squareTube
    name: "SquareTube"
    classID:#(63445,55332)
    category: "Scripted Primitives"
    (local box1, box2
    --
    parameters main Rollout:params
        (length type:#WorldUnits ui:length default:1E-3
            width type:#WorldUnits ui:width default:1E-3
            height type:#WorldUnits ui:height default:1E-3
        )
    --
    Rollout params "SquareTube"
        (Spinner height "Height" type:#WorldUnits range:[1E-3,1E9,1E-3]
            Spinner width "Width" type:#WorldUnits range:[1E-3,1E9,1E-3]
            Spinner length "Length" type:#WorldUnits range:[-1E9,1E9,1E-3]
        )
    --
    on buildMesh do
        (if box1 == undefined then
            (box1 = createInstance box; box2 = createInstance box)
            box1.height = height; box2.height = height
            box1.width = width; box2.width = width * 2
            box1.length = length; box2.length = length * 2
            mesh = box2.mesh - box1.mesh
        )
    --
    tool create
    (on mousePoint click do
        case click of
            1: nodeTM.translation = GridPoint
            3: #stop
        )
        on mouseMove click do
            case click of
                2:(width = abs GridDist.x; length = abs GridDist.y)
                3: height = GridDist.z
            )
        )
    )
)

```

在上例中，参数和 Rollout 的定义形式与第一个例子类似，但是 buildMesh 事件用一种间接的方法生成 Mesh 对象。最后生成的对象是一个矩形管：一个矩形的中间打了一个矩形的洞。两个插件局部变量（box1 和 box2）为两个用 Box 基类生成的对象，其尺寸由插件参数设定，其中 box2 为外圈 Box，box1 为开洞 Box，最后的 Mesh 对象是用布尔减法从 box2 中减去 box1，这样一个新的 Mesh 对象被创建并赋给插件的局部变量 mesh，与第一个例子中直接用 setMesh()方法生成的 Mesh 对象不同。

上例中方法 createInstance()用来直接创建 Box 基类对象，这个方法并不真的在场景里创建对象，仅创建一个与指定对象相对应的几何体。因为在 Simple Object 插件里不允许执行诸如创建对象和修改器堆栈等与场景有关的动作。

## 19.6 Shape 类脚本插件

Shape 类脚本插件只能扩展现有的 Shape 插件。Shape 类脚本插件必须在插件定义中指定其<super\_class>为 shape。如果在插件定义中指定了一个 create tool，它会覆盖其 Delegate 对象的 create tool。

例如：

```
plugin shape Extended_Rect
    name: "Rectangle2"
    classID:#(0x133067, 0x54375)
    extends:rectangle version:1
    category: "Splines"
    (fn fmax val1 val2 = if val1 > val2 then val1 else val2
    tool create
    (local startPoint
        on mousePoint click do
            case click of
                (1: startPoint = nodeTM.translation = GridPoint
                3: #stop
                )
    --
        on mouseMove click do
            case click of
                (2:(delegate.width= abs GridDist.x
                    delegate.length= abs GridDist.y
                    nodeTM.translation = startpoint + GridDist/2.
                )
                3: delegate.corner_radius = fmax 0 -GridDist.x
                )
            )
    )
)
```

## 19.7 Light 类脚本插件

Light 类脚本插件只能扩展现有的 Light 插件。Light 类脚本插件必须在插件定义中指定其`<super_class>`为 Light。如果在插件定义中指定了一个 create tool，它会覆盖其 delegate 对象的 create tool。在 3ds max 里，有一些插件（如 targetLight 类插件）还不能被扩展。如果用户指定的新 Rollout 在 3ds max 界面下不显示，就说明该类插件不能被扩展。

Light 类脚本插件的例子参见 19.2 节。

## 19.8 Helper 类脚本插件

Helper 类脚本插件只能扩展现有的 Helper 类插件。Helper 类脚本插件必须在插件定义中指定其`<super_class>`为 Helper。如果在插件定义中指定了一个 create tool，它会覆盖其 Delegate 对象的 create tool。

Helper 类脚本插件的例子参见 19.2 节。

## 19.9 Modifier 类脚本插件

Modifier 类脚本插件只能扩展现有的 Modifier 插件，且必须在插件定义中指定其`<superclass>`为 Modifier。

例如：

```
plugin modifier myMod
    name: "Supa Mod"
    classID:#(685325,452281)
    extends:Bend replaceUI:True version:1
(parameters main Rollout:params
    (bendamt type:#Float animatable:True ui:bendamt default:0.0
        on bendamt set val do delegate.angle = val
    )
    --
    Rollout params "SupaCheka Parameters"
    (Spinner bendamt "Bendiness: "
    )
)
---
plugin modifier NSpline
    name: "Normal. Spline"
    classID:#(0x133067, 0x54374)
    extends:normalize_spline
    replaceUI:True version:1
```

```

(parameters main Rollout:params
  (seglen type:#Float animatable:True ui:seglen default:20
    on seglen set val do delegate.length = val
  )
  --
  Rollout params "Parameters"
  ( Spinner seglen "Seg Length:" range:[0.01,1e9,20]
  )
)

```

## 19.10 SimpleMod 类脚本插件

SimpleMod 类脚本插件是一种变形类 (Deformation\_type) 的修改器，它可以移动对象的顶点，但是不会改变其拓扑结构（增加、删除对象 Vertice、Face、Surface、Line 等）。在 3ds max 系统里与 SimpleMOD 类脚本插件类似的修改器有 Bend、Stretch 和 Taper。这些插件仅需编程者编写 map 事件处理程序，用于如何处理移动的顶点。SimpleMod 插件必须在定义中指定其<superclass>为 SimpleMod。

例如：

```

plugin simpleMod saddle
  name: "SaddleDeform"
  classID:#(685325,452281)
  version:1
  ( parameters main Rollout:params
    ( amount type:#Integer ui:amtSpin default:20
    )
  --
  Rollout params "Saddle Parameters"
  ( Spinner amtSpin "Amount: " type:#Integer range:[0,1000,20]
  )
  --
  on map i p do
    ( p.z += amount * sin((p.x * 22.5/extent.x)*(p.y * 22.5/extent.y))
      p
    )
  )

```

上例定义了一个新的 SimpleMod 类脚本插件，名字为 SaddleDeform。它会给对象施加一个 saddle\_type 变形，将两个对角向上卷，另两个对角向下卷（读者最好在一个平面对象上观察此变形）。该插件只有一个与插件卷展栏中 Spinner 类控件 amtSpin 绑定的参数 amount，指定了变形的大小。SimpleMod 插件中关键的部分是 map 事件处理程序。其格式为：

on map <index> <point> do <expr>

用户每次修改对象上的顶点后都会调用 map 事件，这些点可能是 Mesh 类顶点、Spline 类顶点、NURBS 控制点等。

参数说明如下：

- ◆ <index>变量指定了点的编号，但此编号并不一定与 Mesh 或 Spline 对象里的顶点编号一致。<index>变量从 1 开始，但是当 map 事件用于计算对象边界框 gizmo 里的点时，index 值为 0。这种情况发生在 SimpleMod 作用在对象的边界框 gizmo 上，且它正被显示的时候，比如在 Modifier 面板下，选中了某一场景对象，然后在 Modifier Stack 里选择了 SimpleMod modifier。
- ◆ <point>变量类型为 Point3，指定了当前点在 object 坐标系里的坐标。

在 map 事件处理程序中，必须对<point>指定的点作适当的调整，并在程序末尾返回它，以作为 map 事件处理的结果。在上例中 map 事件处理程序对<point>点的 Z 坐标施加了一个变形，(用下面的等式： $z=\sin(x*y)$ )，这样所有的点都保持它的 X 和 Y 坐标不变，而 Z 坐标向上或向下移动，就像在 saddle 函数中一样。注意，所有坐标值都是在 object 坐标系下的。

map 事件程序中传递的变量 index 的值要与修改对象的类型相匹配：如果修改对象的类型为 Mesh，变量 index 为 Mesh 对象的顶点序号；如果修改对象的类型为 Patch，变量 index 首先为控制点 (Control Point)，之后跟着每个控制点的 in 矢量和 out 矢量；对 Spline 来说，变量 index 为控制点、in 矢量 和 out 矢量，按曲线控制点的顺序（这个排序方式在以后的 3ds max 版本里可能会改变）。

即使在一个简单的对象里，map 事件被调用次数也可能非常多，因此用户应该在 map 事件里创建最少数量的变量。这将大大减少系统运行时内存收集的工作量，如果在 map 事件中确实需要用到几个局部变量，建议用户在插件定义中声明一个或几个 Point2 或 Point3 类型的变量，然后把要用到的值存放在 Point2 或 Point3 值的元素里，这样可以减少变量的个数，从而减少系统内存分配的工作量。

在 map 事件里不要存取正在被修改的对象，这样会导致系统对脚本插件 Modifier 的重新求值，并导致一个无限次的循环，最后导致 3ds max 系统中止运行。

在 map 事件里，也有 4 个预定义局部变量：

- ◆ Min 与 modifier 关联绑定框的最小坐标值
- ◆ Max 与 modifier 关联绑定框的最大坐标值
- ◆ Center 与 modifier 关联绑定框的中心坐标值
- ◆ extent 与 modifier 关联绑定框的长度

在上例的 map 事件中用到了变量 extend 来计算 saddle 函数的缩放比例，上面 4 个变量都处在特定的关联 Modifier 下，可能是单个对象，也可能是一组对象（如果 Modifier 作用于一个对象选择集），还可能是一个子对象选择集（如果在 Modifier Stack 里 SimpleMod 修改器的下面还有一个子对象集修改器如 Mesh\_select）。

用户还可以在 SimpleMod 类脚本插件里使用一个 modLimit 事件，就像在 Bend 和 Taper 里一样。格式如下：

```
on modLimitZMin do <expr>
on modLimitZMax do <expr>
on modLimitAxis do <expr>
```

如果其中任一个事件被调用，三个事件都会被执行，其中 on modLimitZMIN 和 on

modLimitZMax 事件处理程序的<expr>必须返回浮点数，指定 Z 坐标的最小和最大值，on modLimitAxis 事件的<expr>必须返回 Name 类值#X、#Y 或#Z 来指定极限轴。

MAXScript 全局变量 currentTime 记录了事件被调用时的时间（一般来说，简单的存取变量 currentTime 会返回当前的时间）。执行上面三个事件最简单的方法就是保持 limit 参数，并将它们绑定到 Spinner 或 Checkbox 类界面控件上。

## 19.11 Material 类脚本插件

Material 类脚本插件仅能扩展现有的 Material 插件，且应在定义中指定<superclass>为 Material。

在 Material 类脚本插件中必须至少定义有一个卷展栏。

在 Material 类脚本插件里，和参数块里的参数绑定的 Material 和 Map 类按钮控件并不调用按钮的 on picked 事件，所以，应当将按钮和一个参数块里的参数相连，并使用参数的 set 事件。

例如：

```
--本例扩展了 3ds max 系统的标准 Material 类,
--并将其界面替换为单个 Rollout 里包含两个 Spinner 控件
--和一个 Colorpicker 控件
plugin material myGlass
    name: "Supa Glass"
    classID:#(695425,446581)
    extends:Standard replaceUI:True version:1
(parameters main Rollout:params
    (trans type:#Float default:27 ui:trans
        refrac type:#Float default:1.5 ui:refrac
        col type:#color default:blue ui:col
    --
        on trans set val do delegate.opacity = val
        on refrac set val do delegate.ior = val
        on col set val do delegate.diffuse_color = val
    )
    --
    Rollout params "Glass Parameters"
    (Spinner trans "Transparency:" fieldwidth:45 offset:[-90,0]
        Spinner refrac "Index of Refraction:" fieldwidth:45 offset:[-90,0]
        colorpicker col "Base color:" align:#center
    )
    --
    on create do
        (--setup initial material
            delegate.opacityFalloff = 75
        )
    )
)
```

## 19.12 TextureMap 类脚本插件

TextureMap 类脚本插件仅能扩展现有的 TextureMap 插件，且应在定义中指定 <superclass> 为 TextureMap。

在 TextureMap 类脚本插件中必须至少定义一个卷展栏。

在 TextureMap 类脚本插件里，和参数块里的参数绑定的 Material 和 Map 类按钮控件并不调用按钮的 on picked 事件，所以，应当将按钮和一个参数块里的参数相连，并使用参数的 set 事件。

## 19.13 RenderEffect 类脚本插件

一个 RenderEffect 类脚本插件必须在定义中指定 <superclass> 为 RenderEffect，并在定义中提供一个 apply 事件处理程序。在插件被定义后，用户可以在 RenderEffect | Add 对话框中看到刚定义的插件。RenderEffect 类脚本插件的事件语法为：

```
on apply <bitmap> do <expr>
```

定义好的 RenderEffect 类脚本插件被添加到系统之后，当 Update Scene 或 Update Effect 按钮被按下时，apply 事件会被调用。变量 <bitmap> 被传递至事件，用户可以对它做指定的修改。该位图为当前视窗的渲染图像，且已施加所有前面的渲染效果。用户在 apply 事件处理程序里用 getpixel() 和 setpixel() 函数对它进行修改。位图修改完后被传递给下一个渲染效果继续进行处理。

为了允许 RenderEffect 类脚本插件能利用 g\_buffer 通道，系统为它定义了下面的事件：

1. on channelsRequired do <expr>

<expr> 的求值结果必须为一个元素数据类型为 G 缓存类 Name 的数组。有效的 g\_buffer Name 有以下几个：

#zDepth	#coverage
#matID	#node
#objectID	#shaderColor
#UVCoords	#shaderTransparency
#normal	#velocity
#unClamped	#weight

2. on preApply <bitmap> do <expr>

preApply 事件处理程序允许 RenderEffect 类脚本插件分析变量 <bitmap> 和它的通道，并对其 Delegate 对象的渲染效果进行预处理。

这样，如果用户想对渲染图像加入 #node 和 #coverage 通道，可以先加入一个 on channelsRequired 事件，再加入一个 on preApply 事件，可以用来将图像的 #node 通道作为蒙版输出，然后将它设为指定 Delegate 对象的蒙版。

在 RenderEffect 类脚本插件里，不能调用 render() 函数，这样会导致插件的递归调用。

下面是一个简单的 RenderEffect 类脚本插件例子，它扩展了现有的 Blur\_effect，并添加

了一个卷展栏，其中包含了一个 nodePick 按钮。如果用户选取了一个对象，它会生成一个 mask 通道，并将它用来对选取对象的 Blur 效果进行限制。

```

plugin RenderEffect myBlurFX
    name: "Super Blur FX"
    classID:#(6545,456581)
    extends:Blur
    version:1
    (local tx = bitmaptexture(), cm, g_channels = #(#node, #coverage)
    Rollout params "SupaFX Parameters"
    (label nn align:#center
        pickbutton nodepick "Pick Node"
    )
    --
    parameters main Rollout:params
    (thenode type:#node ui:nodepick
        on thenode set nd do
            params.nn.text = if nd == undefined then "" else nd.name
        )
    --
    on channelsRequired do g_channels
    --
    on preApply map do if theNode != undefined then
        (if cm == undefined then
            (cm = getChannelAsMask map #node node:theNode \
                fileName:(scriptsPath + "_fxtmp.bmp")
            save cm
            tx.bitmap = cm
        )
        else
            getChannelAsMask map #node node:theNode to:cm
        )
    --
    on create do
        (delegate.selMaskActive = True
        delegate.selImageActive = False
        delegate.selMaskMap = tx
        )
    )
    --
    plugin renderEffect myColorBalanceFx
        name: "Super Color Balance FX"
        classID:#(64425,45761)
        extends:Color_Balance version:1
    (parameters main Rollout:params
        (redness type:#Integer animatable:True ui:redness defaule:0.0
            on redness set val do delegate.red = val
        )
    --
    Rollout params "Super Color Balance Parameters"

```

```

    (Spinner redness "Redness:" type:#Integer range:[-100,100,0]
    )
)
---
plugin renderEffect Negative
    name: "Negative"
    classID:#(0xb7aa794c, 0xc3bd78ab)
(parameters main Rollout:params
(Color type:#color default:blue ui:Color
)
--
-- Rollout params "Negative Parameters"
(colorpicker Color "Base color:" align:#center
)
--
-- on apply bmp do
(for h=0 to(bmp.height-1)do
    (local sline = getPixels bmp [0,h] bmp.width
        for i=1 to sline.count do sline[i] = Color - sline[i]
        setPixels bmp [0,h] sline
    )
)
}

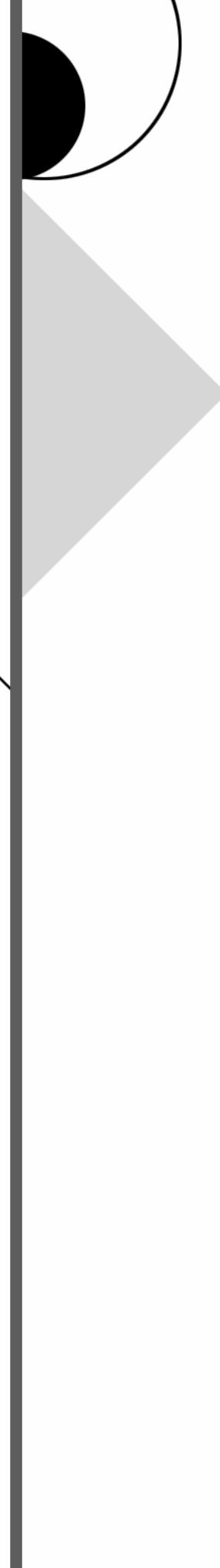
```

## 19.14 Atmospheric 类脚本插件

Atmospheric 类脚本插件仅能扩展现有的 Atmospheric 插件，且应在定义中指定 `<superclass>` 为 Atmospheric。

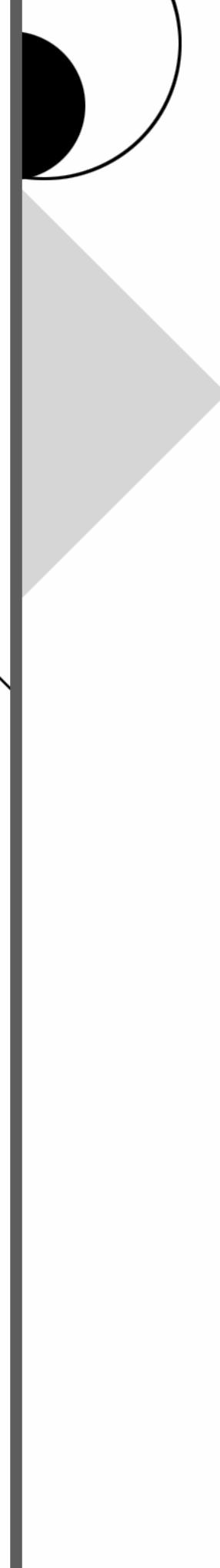
例子：

```
plugin atmospheric myFogEnv
    name: "Super Fog Env"
    classID:#(64433,27761)
    extends:Fog version:1 replaceUI:True
(parameters main Rollout:params
(fogcol type:#color animatable:True ui:col
    on fogcol set val do delegate.fog_color = val
)
--)
Rollout params "Super Fog Parameters"
(colorpicker col "Fog Color"
)
--
on create do delegate.fog_type = 1
)
```



第  
4  
部  
分

# MAXScript 的 高级应用



第  
4  
部  
分

# MAXScript 的 高级应用

# 第 20 章 在 MAXScript 里 与用户界面交互

本章将讲述如何在脚本程序里控制 3ds max 界面，包括按下命令按钮、打开和关闭触发器以及打开对话框等。

## 20.1 Main Toolbar (主工具栏)

下面所列的方法与 3ds max 界面中的 Main Toolbar (见图 20.1) 有关：



图 20.1 3ds max 的 Main Toolbar

1. enableUndo <Boolean>

启动或禁止 Undo 按钮。

2. hitByNameDlg()

打开 3ds max 标准的 Select By Name 对话框让用户选择对象。如果用户按 Cancel 退出对话框，该函数返回 False，否则返回 True。

3. toolMode.uniformScale()

将缩放模式设置为 Uniform Scale。

4. toolMode.nonUniformScale()

将缩放模式设置为 Non-Uniform Scale。

5. toolMode.squashScale()

将缩放模式设置为 Squash。

6. enableRefCoordSys <Boolean>

启动或禁止 Reference Coordinate System 下拉列表。

7. toolMode.coordsys <mode\_name>

设置 Reference Coordinate System 下拉列表的值，有效的<mode\_name>值有：#View、#Screen、#World、#Parent、#Local、#Grid。

8. getRefCoordSys()

setRefCoordSys <mode\_name>

存取 Reference Coordinate System 下拉列表的当前值，有效的<mode\_name>值有：

#View: View 坐标系

#screen: Screen 坐标系

#World: World 坐标系

#Parent: Parent 坐标系

#Local: Local 坐标系

#Object: 捲取对象或 Grid 坐标系 (不适用于 setRefCoordSys() 函数)

9. enableCoordCenter <Boolean>

启用/禁止 Coordinate System Center 图标。

10. getCoordCenter()

setCoordCenter <name>

存取坐标系中心，有效的<name> 值有：

- ◆ #Local 将 Pivot 点作为中心

- ◆ #Selection 将选择集中心作为中心

- ◆ #System 将转换坐标系中心作为中心

11. toolMode.transformCenter()

将坐标系中心设置为#System。

12. toolMode.selectionCenter()

将坐标系中心设置为#Selection。

13. toolMode.pivotCenter()

将坐标系中心设置为#Local。

14. getNumAxis()

本方法可以检测坐标系中心的当前值：如果返回值为#individual，表示坐标系中心为#Local，否则返回值为#all。

15. setToolBtnState <name> <Boolean>

打开或关闭指定工具栏按钮。本方法并不会让系统启动工具栏按钮，而仅仅改变按钮的开关状态，而且本方法仅仅改变指定按钮的状态，有效的<name>值有：

- ◆ #Move Move 按钮，On/Off

- ◆ #Rotate Rotate 按钮，On/Off

- ◆ #Nuscale Scale 按钮，On/Off，不改变缩放类型

- ◆ #Uscale Scale 按钮，On/Off，不改变缩放类型

- ◆ #Squash Scale 按钮，On/Off，不改变缩放类型

- ◆ #Select Select 按钮，On/Off

16. getToolbtnState <name>

返回指定工具栏按钮的开关状态，如果指定工具栏按钮当前被打开，返回 True，否则返回 False。有效的<name>值有：

- ◆ #Select Select 按钮

- ◆ #Move Move 按扭

- ◆ #Rotate Rotate 按钮

- ◆ #Uscale 如果任一 Scale 按钮被打开都返回 True

- ◆ #Nuscale 如果任一 Scale 按钮被打开都返回 True
- ◆ #Squash 如果任一 Scale 按钮被打开都返回 True

下面的方法被用来处理 Named Selection Set 下拉列表：

1. clearCurSelSet()

清除 Named Selection Set 下拉列表编辑框里的内容，但不会取消对当前对象的选择。

2. clearSubSelSets()

从 Named Selection Set 下拉列表中清除所有命名选择集。事实上，命名选择集仍然存在，只是不在下拉列表中显示。如果不是在 Modify 面板的 Sub-Object 模式下使用该命令，可能会带来一些麻烦，因为 3ds max 没有直接的方法重建 Named Selection Set 列表，而如果在 Modify 面板的 Sub-Object 模式下，方法 namedSelSetListChanged() 可重建 Named Selection Set 列表。

3. namedSelSetListChanged()

如果在 Modify 面板的 Sub-Object 模式下，本方法可重建 Named Selection Set 列表。

4. setCurNamedSelSet <String>

将 Named Selection Set 下拉列表的编辑框文本设为指定的字符串。本方法不会改变当前选择集，也不会将指定 <String> 添加到下拉列表清单中。

5. appendSubSelSet <String>

将指定的字符串添加到下拉列表清单中，本方法不会改变当前选择集，3ds max 的 Modifier 可以用本方法向下拉列表里加入一个子对象命名选择集。

下面的全局系统变量与 Main Toolbar 有关：

1. references.constantReferenceSystem

存取 Move、Rotate、Scale 按钮是否使用持续参照坐标系，True 表示使用持续参照坐标系。本系统变量对应 Customize | Preferences | General | Reference Coordinate System 组里的 Constant 复选框。

2. toolmode.commandmode

存取 3ds max 的命令模式。如果设置的命令模式是一个可识别的命令模式，返回值为指定的命令模式，否则返回一个整数。可识别的命令模式有：

#Select	#Modify
#Move	#Motion
#Rotate	#Animation
#Nuscale	#Camera
#Uscale	#Null
#Squash	#Display
#Viewport	#Spotlight
#Hierarchy	#Pick
#Create	

当设置 3ds max 的命令模式时，仅有下面的命令模式是有效的：

```

#Select          #Nuscale
#Move           #Uscale
#Rotate          #Squash
3. toolmode.axisConstraints
存取 3ds max 的 Axis Constraints 按钮的状态，有效值有：#X、#Y、#Z、#XY、YZ、ZX。

```

## 20.2 Status Bar (状态栏)

与 3ds max 界面 Status Bar 有关的方法分别在下面小节中描述。

### 20.2.1 Prompt Line (提示栏)

下面所列的方法与 3ds max 界面中 Prompt Line 有关：

1. `displayTempPrompt <String> <milliseconds_Integer>`

在 Prompt Line 里显示指定字符串，显示持续时间为指定的毫秒数 `<milliseconds_Integer>`，在指定时间之后，字符串会从 Prompt Line 里消失，而且该字符串不会被压入提示信息堆栈 (Prompt Stack)。本方法可用于在 Prompt Line 里显示一些临时提示信息。系统在执行完本方法后，会马上接着往下执行，而不会等待到指定的 `<milliseconds_Integer>` 时间完成。

2. `removeTempPrompt()`

将 Prompt Line 里的临时提示信息马上清除。

3. `replacePrompt()`

将 Prompt Line 的提示信息设为提示信息堆栈的顶端一行。

4. `pushPrompt <String>`

将当前显示提示信息压入提示信息堆栈，并在 Prompt Line 里显示指定字符串。

5. `popPrompt()`

将当前显示提示信息弹出提示信息堆栈，并在 Prompt Line 里显示堆栈里上一条提示信息。

### 20.2.2 Coordinate Display (坐标显示)

下面所列的方法与 3ds max 界面中 Coordinate Display 有关：

1. `disableStatusXYZ()`

禁用鼠标跟踪 (Mouse Tracking) 功能，X、Y、Z 状态盒不显示鼠标点的坐标。

2. `enableStatusXYZ()`

启用鼠标跟踪功能，X、Y、Z 状态盒显示鼠标点的坐标。

3. `setStatusXYZ <format_name> <Point3>`

按指定格式在 X、Y、Z 状态盒里显示 `<Point3>` 值的三个分量，有效的显示格式 `<format_name>` 有：

◆ `#universe` 用当前单位按 position 属性的格式显示 `<Point3>`；

- ◆ #scale 用百分比显示<Point3>, 如果元素值为 1, 显示 100%;
- ◆ #angle 用角度显示<Point3>, 系统按<Point3>里的元素以弧度为单位, 在显示时会自动将各元素乘以 (180/3.14159);
- ◆ #other 用浮点数显示<Point3>。

**注意** 如果在 setStatusXYZ() 里指定 <format\_name> 为 #other, 将不能正确显示指定 <Point3>, X 和 Y 坐标会显示在 Y、Z 状态盒里, 而 X 状态盒为空。

### 20.2.3 Progress Bar (进度栏)

下面的函数让用户在那些要花费较长时间的操作时在界面上显示一个 Progress Bar (进度栏), 就像 IK、Preview Rendering 一样:

1. progressStart “caption”

用指定的标题开始显示一个 Progress Bar。

2. progressUpdate <percent>

按指定的百分比<percent> (0~100) 更新 Progress Bar, 本函数也可用来检测用户是否在 Progress Bar 上按了 Cancel 按钮, 如果按了 Cancel 按钮, 函数返回 False, 否则返回 True。但我们一般会调用下面的 getProgressCancel() 来检测 Cancel 按钮的状态。

3. progressEnd()

结束操作, 并移走 Progress Bar。

4. getProgressCancel()

检测用户是否在 Progress Bar 上按了 Cancel 按钮。在脚本中如果存在深层嵌套循环, 用户会频繁调用该函数以及时检测到 Cancel 按钮的状态。getProgressCancel() 函数和 progressUpdate() 一样, 当用户按了 Cancel 按钮, 系统会显示一个确认对话框, 如果用户在确认对话框里按 Cancel 按钮, getProgressCancel() 函数返回 True, 否则返回 False。而 progressUpdate 函数正好相反, 如果用户在确认对话框里按 Cancel 按钮, 返回 False, 否则返回 True。

5. setProgressCancel <Boolean>

给 Progress Bar 设置或取消 Cancel 标记。如果<Boolean> 为 True, 将给 Progress Bar 设置 Cancel 标记, 并可以被 progressUpdate() 和 getProgressCancel() 函数检测到; 如果<Boolean> 为 False, Progress Bar 的 Cancel 标记被清除。

系统变量 escapeEnable 与 Progress Bar 有关, 请读者参见 4.5.3 节。

### 20.2.4 Status Bar Button (状态栏)

下面所列的方法与 3ds max 界面中 Status Bar 里的按钮有关:

1. getCrossing()

返回当前选择模式: True, Crossing 选择模式; False, Window 选择模式。

2. getPluginKeysEnabled()

setPluginKeysEnabled <Boolean>

存取是否启用 Plug-in Keyboard Shortcuts 按钮: True, 启用; False, 禁用。

3. `getSnapState()`

返回当前 Snap toggle 按钮的状态: True, On; False, Off。

4. `getSnapMode()`

返回当前捕捉类型: Absolute (绝对) 或 Relative (相对)。

5. `isSelectionFrozen()`

返回 Node Selection 按钮是否被冻结: True, 冻结; False, 没有冻结。

6. `freezeSelection()`

冻结被选择的场景对象。

7. `thawSelection()`

解冻被冻结的场景对象。

## 20.3 Time Control (时间控制)

下面所列的方法和系统变量与 3ds max 界面中 Time Control 有关, 其说明请读者参见 4.5.2 节:

`animButtonEnabled`

`animButtonState`

`SliderTime`

下面两个函数用来开始和停止 3ds max 动画播放, 相当于 3ds max 用户界面里的 Play 按钮:

`playAnimation [ #selOnly ]`

`stopAnimation()`

如果指定了可选变量#selOnly, 仅播放当前被选择的场景对象的动画。

当使用上面这些函数时, 读者有一个重要的约束要注意: 直到用户按下 Stop/Play 按钮或调用 `stopAnimation()` 函数来停止动画后, 函数才会返回。目前在 MAXScript 实现后一个目的的唯一方法是: 在脚本 Rollout 或脚本 Controller 的事件处理程序 (如按钮的 press 事件) 里调用 `stopAnimation()`, 而不能在 Listener 窗口里调用 `stopAnimation()` 函数。

## 20.4 Trackbar (轨迹栏)

下面所列方法与 Trackbar 有关:

1. `trackbar.getPreviousKeyTime()`

获取上一个关键帧时间。如果没有上一个关键帧时间, 返回 `undefined`。如:

```
at time SliderTime trackbar.getPreviousKeyTime()
```

2. `trackbar.getNextKeyTime()`

获取下一个关键帧时间。如果没有下一个关键帧时间，返回 undefined。

下面的 3ds max 系统变量 Trackbar 有关，其说明请读者参见 4.5.2 节：

trackbar.filter: Name

trackbar.visible: Boolean

## 20.5 Viewport (视窗)

### 20.5.1 存取当前视窗信息、类型和 Transform 信息

下面所列的系统变量与视窗有关，其说明请读者参见 4.5.2 节：

viewport.activeViewport

viewport.numViews

下面所列方法用来存取活动视窗信息：

1. `viewport.getLayout()`

`viewport.setLayout <view_layout_name>`

存取视窗布局。其中参数`<view_layout_name>`指定了视窗布局配置。如果设定了视窗布局，每一视窗的视图均基于 Customize|Viewport Configuration|Layout 里的配置。

参数`<view_layout_name>`的有效值有：

- ◆ #layout\_1 一个视窗
- ◆ #layout\_2v 两个视窗，垂直排列，相同大小
- ◆ #layout\_2h 两个视窗，水平排列，相同大小
- ◆ #layout\_2ht 两个视窗，水平排列，顶部视窗小一些
- ◆ #layout\_2hb 两个视窗，水平排列，顶部视窗大一些
- ◆ #layout\_3v1 三个视窗，左边两个，右边一个
- ◆ #layout\_3vr 三个视窗，左边一个，右边两个
- ◆ #layout\_3ht 三个视窗，顶部两个，底部一个
- ◆ #layout\_3hb 三个视窗，顶部一个，底部两个
- ◆ #layout\_4 四个视窗，相同大小
- ◆ #layout\_4v1 四个视窗，左边三个，右边一个
- ◆ #layout\_4vr 四个视窗，左边一个，右边三个
- ◆ #layout\_4ht 四个视窗，顶部三个，底部一个
- ◆ #layout\_4hb 四个视窗，顶部一个，底部三个

2. `viewport.getType()`

`viewport.setType <name>`

存取当前视窗的视图名称，有效的`<name>`如下：

- ◆ #view\_top Top 视图
- ◆ #view\_bottom Bottom 视图

- ◆ #view\_right Right 视图
- ◆ #view\_left Left 视图
- ◆ #view\_front Front 视图
- ◆ #view\_back Back 视图
- ◆ #view\_persp\_user Perspective 视图
- ◆ #view\_iso\_user User 视图
- ◆ #view\_camera Camera 视图
- ◆ #view\_spot Light 视图
- ◆ #view\_shape Shape 视图
- ◆ #view\_track Track View 窗口
- ◆ #view\_Grid Grid 视图

**注意** 如果当前视窗为一个扩展视窗(比如 Track View、Asset Manager 或 Listener)getType()函数将返回 undefined。如果在 setType()中指定<name>为#view\_camera 或 #view\_spot, 而当前又没有选择 Camera 或 Light 类对象, 系统会显示一个对话框让用户选择一个 Camera 或 Light 类对象。

viewport.setType()函数返回一个布尔值: True 表示设置成功, 否则返回 False。如果当前没有一个视窗处于焦点状态, viewport.setType()函数也可能返回特定值#view\_none, 而 viewport.GetType()函数返回 False。

例如:

```
viewport.setType #view_track --本命令后没有视窗处于焦点状态
viewport.getType() --返回 False
```

### 3. viewport.ResetAllViews()

重置所有视窗至 3ds max 默认布局。

### 4. getActiveCamera()

```
viewport.getCamera()
```

如果存在与当前焦点视图相连的 Camera 对象, 返回该 Camera 对象, 否则返回 undefined。

### 5. viewport.setCamera <camera\_node>

将焦点视窗设为 Camera, 并使用<camera\_node>指定的摄像机。

### 6. getViewTM()

```
viewport.getTM()
```

```
viewport.setTM <matrix3>
```

存取非正交视窗(如 Perspective 视窗)当前视图的 Screen-to-World 转换矩阵。SetTM 只能操作 Perspective 视窗, 并返回一个布尔值: True 表示转换矩阵设置成功; False 表示不成功。

下面自定义函数返回一个 Ray 类型的值, 函数需要视窗为非正交视窗:

```

fn getViewDirectionRay =
(
    --getViewTM()矩阵为从 World 坐标系到 view 坐标系,
    --因此要先求它的转置矩阵
    local coordSysTM = Inverse(getViewTM())
    --矩阵的 z 轴为视图方向
    local viewDir = -coordSysTM.row3
    --从矩阵获取视图位置
    local viewPt = coordSysTM.row4
    return ray viewPt viewDir
)

```

7. **getViewFOV()**

返回活动视窗的景深 (Field of View)，如果视窗为正交视窗，返回 57.2958 (1 弧度)。

8. **getViewSize()**

返回一个 Point2 值，代表活动视图的尺寸，以像素为单位。

9. **getScreenScaleFactor <Point3>**

返回活动视图的缩放因子。

10. **mapScreenToWorldRay <pixel\_coord\_Point2>**

返回一个 World 坐标系空间的 Ray 类型值，该 Ray 值穿过当前活动视图的<pixel\_coord\_Point2>点，并垂直于视窗平面。

11. **mapScreenToView <pixel\_coord\_Point2> <depth\_Float>**

[ <viewport\_size\_Point2> ]

返回一个 View 坐标系空间的 Point3 值。参数<pixel\_coord\_Point2>为活动视窗里的一个点（在视窗像素坐标系里），<depth\_Float>为在 View 坐标系里的深度。本方法将指定<pixel\_coord\_Point2>点映射到 View 坐标系里。如果提供可选参数<viewport\_size\_Point2>，其被用来代替实际的视窗尺寸。

12. **mapScreenToCP <pixel\_coord\_Point2> [ <viewport\_size\_Point2> ]**

将当前活动视图里的点<pixel\_coord\_Point2>映射到 World 坐标系的构造平面 (Construction Plane)。如果提供可选参数<viewport\_size\_Point2>，其被用来代替实际的视窗尺寸。

13. **getCPTM()**

返回从构造平面 (Construction plane) 到 World 坐标系的转换矩阵。

14. **gw.nonScalingObjectSize()**

从本方法返回的值可以用来抵消视窗缩放的比例因子。例如当用户在缩放视窗时，可以用这个比例因子来保持 Lights、Cameras 和 Tape Helper 等类型场景对象的大小不变。

本值受 Viewport Preferences 对话框里 Spinner 类控件 Non-Scaling Object Size 的影响。

15. **gw.getPointOnCP <pixel\_coord\_Point2>**

返回处在构造平面上与指定<pixel\_coord\_Point2>点对应的点坐标。

16. **gw.getCPDisp <base\_Point3> <dir\_Point3> \**

<start\_pixel\_coord\_Point2> <end\_pixel\_coord\_Point2>

本方法返回一个长度，该长度坐标系为 World，基于一个起点 (<start\_pixel\_coord\_Point2>)、一个终点 (<end\_pixel\_coord\_Point2>)、一个基点 (<base\_Point3>)，一个方向矢量(参数<dir\_Point3>)。如在创建一个 Cylinder 对象时，用户按下鼠标定义对象的 Cylinder 对象的中心（基点），然后鼠标拖拉指定半径，再拖拉鼠标来指定高度。本方法可用来计算 Cylinder 对象高度：其基点为 Cylinder 对象的中心，方向矢量为 Z 轴，起点为鼠标按下点，用户可通过移动鼠标来确定终点。

#### 17. gw.getVPWorldWidth <Point3>

返回 World 坐标系里指定点处的视窗宽度因子。

#### 18. gw.mapCPToWorld <Point3>

返回构造平面上指定点在 World 坐标系里的对应点。本方法是 gw.getPontOnCP()方法的逆操作。

#### 19. gw.IsPerspView()

如果活动视窗为 Perspective 视窗，返回 True，否则返回 False。

#### 20. gw.getFocalDist()

返回活动 Perspective 视窗的焦距 (focal distance)。

#### 21. gw.snapPoint <pixel\_coord\_Point2> [ snapType:<snapType\_name> ]

基于构造平面上的指定点、当前捕捉设置和可选的 snapType:参数，本方法返回一个三维点。有效的 <snapType\_name>值有：

- ◆ #in3d 捕捉所有点
- ◆ #inPlane 仅捕捉构造平面上的点
- ◆ #unSelObjs 当考虑捕捉点时忽略选择 Node 对象
- ◆ #selObjs 当考虑捕捉点时忽略未选择 Node 对象
- ◆ #unSelSubobj 当考虑捕捉点时忽略选择的子对象几何体
- ◆ #selSubobj 当考虑捕捉点时忽略未选择的子对象几何体
- ◆ #force3d 覆盖用户设置，强制捕捉模式为 3d

#### 22. gw.snapLength <length\_Float>

本方法返回按指定长度可捕捉到对象的最近的捕捉长度。

#### 23. units.formatValue <Float>

将指定<Float>按当前单位转化成一个<String>值。本方法有时会导致字符溢出，尤其当单位为 mile 或 kilometer 时。如果发生溢出时，系统会给出一个运行错误。

#### 24. units.decodeValue <String>

将指定<String>按当前单位转化成一个<Float>值，如果转化时发生错误，系统会给出一个运行错误。

### 20.5.2 刷新视窗

下面方法会强制视窗刷新，或当刷新视窗时进行控制：

#### 1. redrawViews()

按最佳方式马上刷新视窗，即只有那些有改变的视图部分被刷新。

#### 2. completeRedraw()

```
forceCompleteRedraw [ doDisabled:<Boolean> ]
```

调用本方法将引起视窗全部刷新，也即强制将所有内容（所有对象、所有屏幕矩形、所有视图）标记为 invalid，然后整个场景被重新生成。但单独对象的管道缓存并不会被更新。如果调用 ForceCompleteRedraw()时指定可选参数 doDisabled:<Boolean>为 True 时，禁止的视窗将不会被刷新。

#### 3. disableSceneRedraw()

关掉视窗刷新功能。

#### 4. enableSceneRedraw()

打开视窗刷新功能。

对 DisableSceneRedraw()或 EnableSceneRedraw()函数的调用必须成对进行，因为 3ds max 内部有一个计数器来控制“启动/禁止”视窗刷新功能。

#### 5. isSceneRedrawDisabled()

如果场景刷新功能可用，返回 True，否则返回 False。

#### 6. nodeInvalRect <node>

将指定对象在视窗中对应的矩形标记为 Invalid，这样在下一次屏幕刷新时对象会被更新。

### 20.5.3 视窗背景图像操作

下面的方法提供了对 Viewport Background 对话框的存取功能：

#### 1. getBkgFrameNum <time>

返回在指定时间视窗的背景图像帧，如果在指定时间没有指定图像帧，返回值 0。

#### 2. getBkgImageAnimate()

```
setBkgImageAnimate <Boolean>
```

存取 Animate Background 按钮是否被打开：True，打开；False，关闭。

#### 3. getBkgImageAspect()

```
setBkgImageAspect <name>
```

存取背景图像的 Aspect Ratio 设置，有效的<name>值有：

- ◆ #View 与 Viewport 匹配

- ◆ #Bitmap 与 Bitmap 匹配

- ◆ #Output 与 Rendering Output 匹配

#### 4. getBkgORType <start\_end\_Integer>

```
setBkgORType <start_end_Integer> <name>
```

存取背景图像的 Start Processing 和 End Procession 设置。<start\_end\_Integer> 等于 0 表示 Start Processing；等于 1 表示 End Processing。合法的<name>值有：#blank、#hold、#loop。

#### 5. getBkgRangeVal()

获取背景图像的 Use Frame range 设置，返回值为一个 Point2 值，其 x 分量为开始帧，y 分量为结束帧。

#### 6. setBkgFrameRange <Point3>

设置背景图像的 Use Frame range 和 step 设置，返回值为一个 Point2 值，其 x 分量为开始帧，y 分量为结束帧，z 分量为步长。

#### 7. getBkgStartTime()

setBkgStartTime <time>

存取背景图像的 Start At frame 设置。

#### 8. getBkgSyncFrame()

setBkgSyncFrame <Integer>

存取背景图像的 Synch Start To Frame frame 设置。

下面的 3ds max 系统变量与 Viewport Background 对话框有关：

**backgroundImageFileName**

存取一个 String 值，定义视窗背景图像位图文件名。包含了 Viewport Background 对话框里设置的位图文件名。

## 20.5.4 视窗网格 (Viewport Grid)

下面的方法用来控制视窗网格线 (Grid Line) 的显示和显示间距：

#### 1. getGridSpacing()

返回网格线显示间距。

#### 2. getGridMajorLines()

返回主网格线 (Major Grid Line) 之间的网格数量。

3ds max 系统变量 activeGrid 与视窗网格有关，请读者参见本书 4.5.2 节。

## 20.5.5 鼠标光标 (Mouse Cursor)

下面的方法用来控制鼠标光标：

#### 1. setWaitCursor()

setArrowCursor()

用来设置当进行较长时间的计算时，显示等待光标来取代标准光标。有时可以用等待光标来代替在视窗上放置一个 Progress Bar。

#### 2. setSysCur <name>

设置当前系统光标，有效的<name>值有：#Move、#Rotate、#Uscale、#Nuscale、#Squash、#Select、#Arrow。

**注意** 设置 setSysCur #squash 会显示一两个方向比例不一致的光标。

下面的 3ds max 系统变量与鼠标光标有关：

#### 1. mouse.mode

只读系统变量，返回一个<Integer>值，代表鼠标模式，返回值有：

0: Idle; 1: Point; 2: Move。

### 2. mouse.buttonStates

只读系统变量，返回一个三元素`<bitArray>`值，分别代表三个鼠标按键状态，其顺序为：

`#{Left, Middle, Right}。`

**注意** 右键状态并不总是正确。如果用右键打开一个右键菜单，然后又在视窗里右击鼠标取消菜单，此时 `mouse.buttonStates` 里右键仍然显示为按下状态。

### 3. mouse.pos

只读系统变量，返回一个`<Point2>`值，代表鼠标在当前活动视窗的位置。坐标值以像素为单位。如果当前活动视窗为二维视图（如 Track View、Schematic View、Listener），坐标值为第一个非二维视图。

## 20.5.6 在视窗里拾取点

```
1. pickPoint [ prompt:<String> ] [ snap:#2D|#3D ] \
    [ rubberBand:<start_Point3> ] \
    [ mouseMoveCallback:fn | #(fn,arg)] \
    [ terminators:#(<String>,<String>,...)]
```

函数 `PickPoint()`让用户在 3ds max 视窗里拾取或在 Listener 窗口里输入一个三维点。当调用该函数时，系统进入点击命令状态，光标变成十字丝。用户可以在 3ds max 视窗里拾取或在 Listener 窗口里输入一个三维点。函数返回一个 `Point3` 值，代表刚才输入的点，坐标系为 World。

各参数说明如下：

- ◆ `Prompt:`为可选参数，数据类型为 `String`，会打印在 Listener 窗口作为提示信息。
- ◆ `snap:#2D|#3D` 为可选参数，控制当捕捉功能被打开时，在视窗里如何拾取点。如果为`#3D`，光标可捕捉到 3D 空间里任何点，如果为`#2D`，光标仅能捕捉当前活动 Grid 的构造平面里的点。如果当前捕捉功能被关闭，鼠标点击点总是在当前活动 Grid 的构造平面上。
- 用户随时可以按 Esc 键或右击鼠标来取消点拾取操作，函数会立即返回一个特殊的 MAXScript 值`#escape`（如果按 Esc 键）或`#rightClick`（如果右击鼠标）。
- ◆ `rubberBand:<start_Point3>`为可选参数，可以让 `PickPoint` 函数在输入点时显示一个橡皮条虚线。橡皮条虚线从指定`<start_Point3>`开始，指向鼠标点。在需要输入多个点时，用户可将参数 `rubberBand:`指定为前一个输入点，来提示用户输入轨迹。参数 `<start_Point3>`指定的点必须在 World 坐标系。
- ◆ `mouseMoveCallback:`为可选参数，可让用户在输入点时设置一个鼠标移动 Callback 程序来跟踪鼠标移动。它有两种格式：

`mouseMoveCallback:fn`

`mouseMoveCallback:#(fn, arg)`

第二种格式允许用户在每次调用函数时指定参数，并将这个参数传递给 Callback 程序。

Callback 程序 fn 在第一种格式里必须有一个参数，在第二种格式里必须有两个参数。调用该 Callback 程序时，第一个参数总是当前鼠标在 World 坐标系里的位置，第二个参数为用户提供变量 arg。用户可以用 Callback 程序来执行一个更复杂的 rubberbanding，如用鼠标的移动来调整 Spline 对象的顶点或 Box 对象的高度，此时用户应确保用 update() 函数来更新 Mesh 或 Spline 对象。

用户还可以直接在 Listener 窗口里直接输入坐标点，有下面几种输入格式（基于 AutoCAD 的命令行输入语法）：

输入格式	说明
x, y, z	指定当前构造平面坐标系里的点
x, y	当前构造平面 (cp) 里的点
d	以前一点为原点沿鼠标方向偏移指定距离的点
@ x, y, z	相对坐标，以前一点为原点
@ x, y	在构造平面 (cp) 里从前一点偏移指定距离的点
d < a	在 cp 里的极坐标，从 cp 原点沿指定角度 a (绕 x 轴) 偏移指定距离 d 的点
@ d < a	相对坐标，以前一点为原点
d < a1 < a2	在 cp 里的球坐标，距离 cp 原点 d，距 xy 平面的 x 轴 a1，距 y 轴 a2
@d < a1 <a2	相对球坐标

**注意** 这些从键盘输入的坐标总是被解释为相对于当前活动 Grid 构造平面，而由 pickPoint() 函数返回的坐标总是在 World 空间。

如果键盘输入的内容不是上面所列的格式之一，函数返回值为包含输入内容的字符串，这样编程者可以由此作出判别。用户可以用函数 classof() 来判别返回值的数据类型，如下例：

```
p = pickPoint()
case of
(
  (p == undefined): ...      -- 用户按 ESC 键
  (p == #rightClick): ...    -- 用户按鼠标右键
  (classOf p == Point3): ... -- 用户输入了点
  (classOf p == String): ... -- 用户输入了非点数据
)
```

◆ terminators：为可选参数，后面必须提供一个长度为 1 或以上的字符串数组，每个元素包含一个或以上的字符。如果在 pickPoint() 函数中指定了该参数，若用户输入的字符串中出现了该字符串数组中的任一元素，pickPoint() 函数会马上返回，而不用等待用户按 ENTER 键。如果指定该参数，pickPoint() 函数返回值为一个二元数组：第一个元素为输入点，第二个元素为中断字符或 undefined（当用户用鼠标点击输入或用 ENTER 键结束键盘输入时）。

用户可以从键盘键入一个点，并用一个中断字符来结束输入；也可以仅仅输入一个中断字符。在第二种情况下，返回数组的第一个元素为 undefined。用户可以通过检测第二个元素来判别输入的中断字符，如：

```
pp = pickPoint prompt:"foo:" terminators:#("", "a", "oo")
if pp[2] == "a" then ...
```

我们经常在脚本程序里反复调用 `pickPoint()` 函数来输入多个点，如 `Line` 对象的顶点。如果用这种方法来构造场景对象，用户需要用一个新的函数 `redrawViews()` 来强制刷新视窗。`MAXScript` 通常只有等到一个脚本程序运行完成后才会刷新视窗。

```
2. pickOffsetDistance [ prompt:<String> ] [ pt2Prompt:<String> ] \
[ errPrompt:<String> ] [ snap:#2D|#3D ] \
[ keyword:<String> ]
```

功能：用户输入两个点，返回这两点在 `World` 空间的距离。

操作：本函数先向 `Listener` 窗口发布参数 `Prompt:`（如果有提供的话），然后等待用户输入，输入可以有下面几种情况：

- ◆ 在视窗或 `Listener` 窗口里输入点（和 `pickPoint()` 函数一样）；
- ◆ 从键盘输入一个以 `ENTER` 结束的 `Number`；
- ◆ 从键盘键入由可选参数 `keyword:` 指定的字符串全部或开始部分。

本函数的返回值分下面几种情况：

- ◆ 如果用户键入参数 `keyword:` 指定的字符串，函数返回值 `True`；
- ◆ 如果键入一个 `Number` 数据，返回该值；
- ◆ 如果用户输入一个点，函数会接着在 `Listener` 窗口里发布可选参数 `pt2Prompt:`（如果有提供的话）指定的提示信息，并等待用户输入第二点，然后返回两点在 `World` 空间里的距离；
- ◆ 如果用户键入一个既不是 `Number` 又与参数 `Keyword:` 指定的字符串不相匹配的字符串，函数会发布可选参数 `errPrompt:`（如果有提供的话）或 `Prompt:`（如果没有提供可选参数 `errPrompt:` 的话）指定的提示信息，并等待用户重新输入。

### 20.5.7 3ds max 图形系统的低级存取方法

本节讲述了一些对 `3ds max` 图形系统的低级存取方法。这些方法可让用户在脚本程序里进行任何图形操作，而这是用那些高级图形方法做不到的。

这些方法仅适用于当前活动的视窗。注意不要随意使用这些方法，因为这些方法虽然可以提高运行速度，但对编程者来说，要想掌握它们却不容易。

本节所描述的方法实际是在图形窗口进行操作。图形窗口与视窗并不是同一概念，但它们互相关联。每一视窗都有它自己的图形窗口，视窗上显示的内容可以看作是图形窗口内容的一个“快照”，因为把信息写入显存是一个相对耗费较长时间的操作，`3ds max` 先把它们写入图形窗口，然后当信息写入完成后，系统再用图形窗口的内容来刷新视窗。

1. `gw.getDriverString()`

本方法返回一个标识图形驱动器的字符串（如果有的话，还包括制造商的信息）。

2. `gw.querySupport <feature_name>`

判别图形驱动器是否支持指定特征，有效的`<feature_name>`有：

- ◆ #txtCorrect 用来设置启动或禁止视窗右键菜单选项;
- ◆ #geomAccel 用来通知 3ds max (主要是 Mesh 类) 图形驱动器希望处理所有三维数据。如果返回 True, 系统渲染 Mesh 对象时, 会将其三维空间信息传递给图形驱动器, 由它进行转换、裁剪、计算顶点光线等运算。如果返回 False, Mesh 对象自己处理所有这些运算, 然后调用 hPolygon 或 hPolyline 2 1/2D 方法为图形驱动器进行光栅处理 (标准几何体类对象仍然发送给 polygon/polyline 方法进行处理)。目前只有驱动器型号为 OpenGL 时, gw.querySupport #geomAccel 才会返回 True, 其他驱动器只有升级后才会返回 True, HEIDI 和 D3D 类驱动器在以后版本可能会改成返回 True;
- ◆ #triStrips 如果返回 True, 3ds max 会在调用渲染方法前尝试对 Mesh 对象进行条纹处理;
- ◆ #dualPlanes 如果驱动器支持双平面 (dual-plane), 返回 True, 否则返回 False;
- ◆ #swapModel 如果一旦视窗被显示, 3ds max 就必须刷新整个场景, 返回 True, 否则返回 False;
- ◆ #incrUpdate 如果驱动器能够刷新一个视窗矩形子集而且不会影响矩形之外的图像, 返回 True。对 OpenGL 显示驱动器而言, 如果驱动器执行 Microsoft glSwapRectHintWIN 扩展的话, 返回 True;
- ◆ #passDecal 如果驱动器仅可以由一个途径处理 decall 操作, 返回 True, 对 OpenGL 显示驱动器, 返回 True, 对 HEIDI 和 D3D, 返回 False;
- ◆ #driverConfig 如果驱动器有一个配置对话框, 返回 True。对 3ds max 的三个标准驱动器, 都返回 True;
- ◆ #texturedBkg 如果视窗背景按 Textured Rectangle(纹理矩形)方式渲染, 返回 True; 如果按 Blitted Bitmap 方式渲染, 返回 False;
- ◆ #virtualVpts 如果驱动器允许视窗大小超出与之相连的屏幕窗口, 返回 True。目前仅有 OpenGL 返回 True;
- ◆ #paintDoesBlit 如果 WM\_PAINT 信息会导致 blit backbuffer, 返回 True。和 #swapModel 标志一起, 3ds max 可以快速修复图形坏区;
- ◆ #wireframeStrips: 如果驱动器希望 3ds max 用三角条纹 (triangle strip) 方式而不是两点段束方式 (bundle of 2-point segments) 向下传送 Wireframe 模型, 返回 True。本设置仅用于 OpenGL 驱动器。

### 3. gw.isPerspectiveView()

如果视图是透视投影, 返回 True; 如果为正交投影, 返回 False。

### 4. gw.setTransform <matrix3>

将活动视窗的图形窗口之转换矩阵设置为指定矩阵<matrix3>, 并且按法向投影坐标系矩阵刷新模型。

### 5. gw.getFlipped()

如果当前转换矩阵的值为正, 返回 True; 如果为负, 返回 False。

### 6. gw.setPos <x\_Integer> <y\_Integer> <w\_Integer> <h\_Integer>

设置图形窗口的尺寸和位置。坐标值为在图形窗口的父窗口下的窗口坐标, 坐标格式

为 Windows 格式，原点在左上角。

各参数说明如下：

- ◆ <x\_Integer> 指定图形窗口左边界位置；
- ◆ <y\_Integer> 指定图形窗口上边界位置；
- ◆ <w\_Integer> 指定图形窗口宽度；
- ◆ <h\_Integer> 指定图形窗口高度。

#### 7. gw.getWinSizeX()

返回当前窗口在 X 轴向的大小。

#### 8. gw.getWinSizeY()

返回当前窗口在 Y 轴向的大小。

#### 9. gw.getWinDepth()

返回 z-buffer 景深。注意：本方法在 3ds max 和 VIZ R3 里的返回值并不是 Z 值。

#### 10. gw.getHitherCoord()

返回设备最大 Z 值。

#### 11. gw.getYonCoord()

返回设备最小 Z 值。

#### 12. gw.setViewportDib()

返回一个 Bitmap 值，包含活动视窗的图形窗口图像。位图大小与视窗相同。

#### 13. gw.resetUpdateRect()

本方法重置刷新矩形。刷新矩形是一个区域，其里面包含了需要刷新的对象（以反映对它们所做的修改）。当系统进行渲染时，其实主要是刷新视窗中被修改的区域。本方法实际将刷新矩形设为一个“空”值。如果紧接着调用 gw.enlargeUpdateRect() 函数，该函数指定的参数<Box2>会成为新的刷新矩形。

#### 14. gw.enlargeUpdateRect(<Box2> | #whole)

本方法会扩大刷新矩形以包含<Box2>指定的区域，如果参数为#whole，刷新矩形为整个视窗。

#### 15. gw.getUpdateRect()

返回一个 Box2 值，包含当前刷新矩形。如果当前没有视窗区域需要刷新，返回一个“空”的 Box2 值。

#### 16. gw.setRndLimits <render\_limits\_name\_array>

为初级函数调用设置渲染界限，其用途用于为 3ds max 系统处理显示对象的各部分之间的信息传递。参数<render\_limits\_name\_array>为一个元素为<render\_limits\_name>值的数据组，指定了用于视窗的渲染界限，有效的<render\_limits\_name>值有：

- ◆ #allEdges 显示对象的所有边（包括隐藏的边）；
- ◆ #boxMode 用绑定框（Bounding Box）显示对象；
- ◆ #backcull 使用“忽略背面”机制（不显示曲面法线与视窗方向不一致的对象）；
- ◆ #colorVerts 打开 color-per-vertex 显示；

- ◆ #flat 使用 Flat (facet) 着色模式;
- ◆ #illum 表示 Polygon 类对象为 color-per-vertex 显示，并且使用它们。如果 polygon 类对象为 color-per-vertex 显示，而没有设置本标记，顶点颜色被忽略;
- ◆ #lighting 与#illum 和#specular 设置相同;
- ◆ #noAtts 不设置任何标记;
- ◆ #perspCorrect 在本模式下，Perspective 视窗里的对象纹理被正确显示;
- ◆ #pick 指示 3ds max 执行碰撞检测（不是在渲染时）;
- ◆ #polyEdges 打开 Polygon 类对象的边;
- ◆ #shadeCverts 本设置会调整#colorVerts 设置。如果被设置，启动灯光，顶点颜色被用来调整灯光颜色；如果没有设置，每一顶点的颜色被直接用来对三角面片着色。当 3ds max 使用#shadeCverts 模式时，对象使用一种只有漫反射（diffuse-only）的材质，这样着色的对象看起来不会失真。

当#shadeCverts 为 Off 时，顶点颜色被直接使用，其效果等同于用一个纯白色自发光的材质对对象进行调整。当#shadeCverts 为 On 时，白色材质被场景灯光照亮，这样着色范围从黑色 ([0,0,0]) 到白色 ([255,255,255])，绝大多数顶点颜色为纯灰色。当顶点颜色被材质颜色调整时，顶点颜色被乘以一个小于 1 的系数，这样让它们看起来颜色更深一些。

颜色的 RGB 分量被统一地调整，这样不会发生颜色偏移，比如从红色到绿色。

- ◆ #specular 启用镜面高光显示模式（specular highlight display）;
- ◆ #texture 启用纹理显示模式（texture display）;
- ◆ #twoSided 所有面都会被显示，而不管它们的表面法线方向；
- ◆ #vertTicks 本模式是一种“准”模式，实际上它不会引起图形驱动器的任何变化，当它被设为 on 时，仅仅用于 Mesh 类：当它们向下发送信息时，顶点被标记；
- ◆ #wireframe 设置 Wireframe 渲染模式；
- ◆ #zBuffer 当为标准几何体对象指定坐标时，都有 x、y 和 z 值，但在视窗中绘制对象时，我们有时希望忽略它的 Z 坐标，如显示在视窗左上角表示视窗类型的文字（如 Front, Left）和 Arc-Rotate 圆、三角轴标志。设置本标志可以在绘制视窗时忽略这些对象的 Z 坐标，而总是将它们放在视窗的最前面。

#### 17. gw.getRndLimits()

返回一个元素类型为 Name 的数组，包含渲染界限。Name 值列表详见 gw.setRndLimits()。

#### 18. gw.getRndMode()

返回一个元素类型为 Name 的数组，包含当前视窗渲染模式，它是渲染界限的一个子集。Name 值列表详见 gw.setRndLimits()。

#### 19. gw.setSkipCount <skip\_count\_Integer>

当视窗被设置为 Fast View Display 模式时，设置跳过的三角面片数目。如果设置参数 <skip\_count\_Integer> 为 1 表示禁止 Fast View Display 模式。渲染 Mesh 类对象时，因为 3ds max 每次只向图形驱动器传送一个三角形面，本参数让 3ds max 按指定间隔

<skip\_count\_Integer> 传递三角形面给图形驱动器。

#### 20. gw.getSkipCount()

返回当前设置的 gw.setSkipCount 数目。

下面的脚本是一些上面函数应用的例子：

```
--获取当前渲染界限
lim = gw.getRndLimits()
--添加一个界限
append lim #polyEdges
--设置新的渲染界限
gw.setRndLimits lim
--获取渲染界限
gw.getRndLimits()
--获取渲染界限模式
gw.getRndMode()
```

#### 21. gw.getMaxLights()

返回可用于交互渲染器的最大灯光数目。

下面方法将点从图形窗口的当前坐标系映射到设备空间。如果图形窗口的当前坐标系为一个单位矩阵 (Identity Matrix)，映射变为从 World 空间到设备空间；否则映射为将点用图形窗口的当前转换矩阵转换到 World 空间。这样，如果要从 World 空间转换到 Screen 空间，用户必须先用下面的方法把图形窗口的转换矩阵设置为单位矩阵：

```
gw.setTransform(Matrix3 1)
```

#### 22. gw.hTransPoint <Point3>

本方法将指定点<Point3>转换为一个 h 格式的设备坐标 (Device Coordinate)。返回值的各分量都是整数。对 HEIDI 和 OpenGL 显示器，坐标原点为左下角，对 Direct3D 显示器坐标原点为左上角。

#### 23. gw.wTransPoint <Point3>

本方法将指定点<Point3>转换为一个 w 格式的设备坐标。返回值的各分量都是整数。坐标原点为左上角。

#### 24. gw.transPoint <Point3>

本方法将指定点<Point3>转换为一个 h 浮点格式的设备坐标 (device coordinate)。返回值的各分量都是浮点数。坐标原点为左上角。本方法是一个辅助程序，以避免运算时发生四舍五入错误，主要用于 IK。

下面列出的函数可以直接在视图中进行图形绘制。这些函数中，以 h 开始的方法使用整型设备坐标，原点在左下角。以 w 开始的方法使用 Windows 设备坐标，原点在左上角。坐标系都是左手坐标。不以 h 和 w 开始的方法将指定点从图形窗口的当前转换坐标映射到 World 空间，坐标系是右手坐标。

在调用下面方法在图形窗口中完成绘制后，需要调用 gw.updateScreen() 函数以刷新视窗显示。

#### 25. gw.updateScreen()

刷新视窗以显示调用下述方法在图形窗口中写入的 Text、Marker、Polyline、Polygons

或 Tristrip。

```
26. gw.text <Point3><String> [ color:<color> ]
gw.hText <Point3><String> [ color:<color> ]
gw.wText <Point3><String> [ color:<color> ]
```

用指定颜色（可选）在指定位置<Point3>写入指定的二维文本。如果没有指定颜色，使用红色作为默认颜色。注意如果文本超出屏幕显示界限，系统会执行裁剪操作。

```
27. gw.Marker <Point3><marker_name> [ color:<color> ]
gw.hMarker <Point3><marker_name> [ color:<color> ]
gw.wMarker <Point3><marker_name> [ color:<color> ]
```

用指定颜色（可选）在指定位置<Point3>做一个标记。如果没有指定颜色，使用红色作为默认颜色。本方法可以与 pickpoint() 函数连用以快速显示刚刚选择的点的位置。这个标记是临时的，视窗一旦被刷新，标记就会被删除。有效的标记类型<marker\_name>有：

- ◆ #point 点标记
- ◆ #hollowBox 中空框标记
- ◆ #plusSign “+”号标记
- ◆ #asterisk “\*”号标记
- ◆ #xMarker “X”标记
- ◆ #bigBox 大框标记
- ◆ #circle 圆标记
- ◆ #triangle 三角形标记
- ◆ #diamond 钻石标记
- ◆ #smallHollowBox 小中空框标记
- ◆ #smallCircle 小圆标记
- ◆ #smallTriangle 小三角形标记
- ◆ #smallDiamond 小钻石标记

下面是一个创建各种类型标记的例子：

```
arr = #("point", "hollowBox", "plusSign", "asterisk", "xMarker",
"bigBox", "circle", "triangle", "diamond", "smallHollowBox",
"smallCircle", "smallTriangle", "smallDiamond")
for i=1 to arr.count do
gw.hMarker [100,(50 + i*10), 50](arr[i] as name)
gw.enlargeUpdateRect #whole
gw.updateScreen()
```

```
28. gw.Polyline <vertex_Point3_array><isClosed_Boolean> [ rgb:<color_array> ]
gw.hPolyline <vertex_Point3_array><isClosed_Boolean> [ rgb:<color_array> ]
gw.wPolyline <vertex_Point3_array><isClosed_Boolean> [ rgb:<color_array> ]
```

上述方法绘制一个多段 Polyline 对象，数组<vertex\_Point3\_array>里的每一元素都是 polyline 的一个顶点。如果参数<isClosed\_Boolean>为 True，表示第一点与最后一点相连，

也即新创建的 Polyline 对象是封闭的；如果为 False，Polyline 对象为打开的。如果有指定可选颜色数组 `rgb:`，着色模式为 Smooth，Polyline 对象采用 Gouraud 着色模式；如果 `rgb:` 没有指定，Line 使用函数 `gw.setColor()` 设置的颜色进行绘制。数组`<color_array>`和`<vertex_Point3_array>`的元素数目必须相同。

```
29. gw.Polygon <vertex_Point3_array> <color_array> <uvw_Point3_array>
   gw.hPolygon <vertex_Point3_array> <color_array> <uvw_Point3_array>
   gw.wPolygon <vertex_Point3_array> <color_array> <uvw_Point3_array>
```

上述方法绘制一个由多点 polyline，数组`<vertex_Point3_array>`里的每一元素都是 polyline 的一个顶点。数组`<color_array>`指定了每一顶点的颜色，注意为了在渲染时使用这些指定的颜色，渲染模式（用 `gw.setRndLimits()` 设定）里必须包含#illum。`<uvw_Point3_array>`指定了每一顶点的 UVW 坐标。每一个参数数组的元素数目必须相同。

```
30. gw.hTriStrip <vertex_Point3_array> <color_array> <uvw_Point3_array>
   gw.wTriStrip <vertex_Point3_array> <color_array> \ <uvw_Point3_array>
```

本方法用来绘制一系列的三角，这些三角组成一个三角带。数组元素数目必须大于或等于 3。本方法不会对超出显示界限的顶点进行剪辑操作，因此必须保证指定的顶点都在显示界限之内。本方法参数和函数 `gw.Polygon()` 相同，但是对`<vertex_Point3_array>`参数的处理方法不同：除前两个顶点外，每一个新的顶点都创建一个新的三角。比如，创建一个四边形时，前三个顶点组成一个三角形，第四个顶点与第二、三个顶点组成第二个三角形，并完成四边形的创建。

下面的脚本是使用上面函数的例子：

```
--绘制一些 primitive 对象
gw.hPolyline #([300,50,16], [300,200,8], [450,250,4])True
--
gw.hPolygon #([200,100,16], [280,100,8], [250,200,4])\
    #(red, blue, green)\
    #([1.0,.5,0], [0.5,0.5,0], [0,0,0.5])
--
gw.hTriStrip #([50,50,0], [175,100,0], [25,100,0], [150,250,0])\
    #(red, blue, green, white)\
    #([1.0,.5,0], [0.5,0.5,0], [0,0,0.5], [0.5,1,0])
--
--刷新视窗
gw.enlargeUpdateRect #whole
gw.updateScreen()
```

31. `gw.setColor <type_name> <color_value>`

为指定绘制类型`<type_name>`设置 RGB 颜色。有效的`<type_name>`有：

- ◆ `#line` Line 对象绘制颜色
- ◆ `#fill` Polygon 对象填充颜色
- ◆ `#text` Text 对象绘制颜色
- ◆ `#clear` 当调用函数 `gw.clearScreen()` 清除视窗时所用的颜色

### 32. gw.clearScreen <Box2> [ useBkg:<Boolean> ]

清除屏幕上指定的矩形区域。如果可选参数 useBkg: 为 False，矩形区域被#clear 颜色填充（见上面 gw.setColor() 函数），如果 useBkg: 为 True，使用图形背景颜色填充矩形区域。默认值为 False。

## 20.5.8 其他视窗方法和系统变量

### 1. createPreview()

使用对话框 Make Preview 的当前设置创建一个视窗预览。

### 2. getVPortBGColor()

setVPortBGColor()

存取视窗背景颜色。

### 3. isCPEdgeOnInView()

如果构造平面（Construction Plane）与当前视窗垂直，返回 True。比如：如果构造平面为 XY，这时如果用户从 Front 视窗观察，本方法返回 True。

### 4. axisTripodLocked()

如果视窗三角轴架被锁住（也即：当用户移动对象或子对象时，它们不会跟着移动），返回 True，否则返回 False。

### 5. lockAxisTripods <Boolean>

如果<Boolean>为 True，锁住视窗三角轴架，它们不会刷新，如果为 False，视窗三角轴架会和视图一起被刷新。

### 6. flashNodes <node\_array>

本方法用来将指定的对象迅速地删除后并在视窗里对它们进行重绘。通常用来作为一些操作的确认，如确认点取对象操作的完成。

#### 注意

flashNodes()方法在将指定对象显示为白色的格栅（Wireframes）后并不重绘视窗，如果要正确地重绘视窗，在调用 flashNodes() 后应马上调用 forceCompleteredraw() 函数。

下面的 3ds max 系统变量可用于视窗，其详细说明参见本书 4.5.2 节：

displaySafeFrames

preferences.useLargeVertexDots

preferences.useTransformGizmos

preferences.useVertexDots

## 20.6 3ds max 用户界面颜色

下列方法用于存取 3ds max 用户界面颜色。

### 1. GetUIColor <index\_Integer>

返回一个 Point3 值，包含在 3ds max 用户界面里绘制指定条款类型<index\_Integer>设

置的颜色值。每一数组元素取值范围为 0~1，对应颜色值为 0~255，<index\_Integer>对应对象类型见下表。

### 2. GetDefaultUIColor <index\_Integer>

返回一个 Point3 值，包含在 3ds max 用户界面里绘制指定<index\_Integer>对应的条款类型设置的默认颜色值。每一数组元素取值范围为 0~1，对应颜色值 0~255，<index\_Integer>对应对象类型见下表。返回值不受 SetUIColor() 方法设置的影响。

### 3. SetUIColor <index\_Integer> <Point3>

设置在 3ds max 视窗里绘制指定条款类型<index\_Integer>的颜色值。每一数组元素取值范围为 0~1，对应颜色值为 0~255，<index\_Integer>的取值范围为 0~103。

有效的<index\_integer>值和它对应的条款类型见下表：

<index_Integer>	对应条款	所在用户界面	说明
0	COLOR_SELECTION	Main UI	选择对象
1	COLOR_SUBSELECTION	Main UI	子对象选择集
2	COLOR_FREEZE	Main UI	冻结
3	COLOR_GRID	Grids	Grid
4	COLOR_GRIDINTENS	Grids	Grid 亮度
5	COLOR_SFLIVE	Main UI	当前 Safe Frame
6	COLOR_SFACTION	Main UI	Safe Frame 动作
7	COLOR_SFTITLE	Main UI	Safe Frame 标题
8	COLOR_VPLABELS	Main UI	视窗标签
9	COLOR_VPINACTIVE	Main UI	非活动视窗标签
10	COLOR_ARCBALL	Main UI	Arcball
11	COLOR_ARCBALLHILITE	Main UI	Arcball 高亮部分
12	COLOR_ANIMBUTTON	Main UI	Animate 按钮
14	COLOR_LINKLINES	Gizmos & Apparatuses	Bone 和连接线
15	COLOR_TRAJECTORY	Gizmos & Apparatuses	轨迹
16	COLOR_ACTIVEAXIS	Gizmos & Apparatuses	活动轴
17	COLOR_INACTIVEAXIS	Gizmos & Apparatuses	非活动轴
18	COLOR_SPACEWARPSS	Objects	SpaceWarp 对象
19	COLOR_DUMMYOBJ	Objects	Dummy 对象
20	COLOR_POINTOBJ	Objects	Point 对象
21	COLOR_POINTAXES	Objects	Point 轴
22	COLOR_TAPEOBJ	Objects	Tape 对象

(续表)

<index_Integer>	对应条款	所在用户界面	说明
24	COLOR_GIZMOS	Gizmos & Apparatuses	Gizmo
25	COLOR_SELGIZMOS	Gizmos & Apparatuses	被选择的 Gizmo
26	COLOR_SPLINEVECS	Main UI	Spline 矢量
27	COLOR_SPLINEHANDLES	Main UI	Spline 句柄
29	COLOR_PARTICLEEM	Gizmos & Apparatuses	粒子发射器
30	COLOR_CAMERAOBJ	Objects	Camera 对象
31	COLOR_CAMERACONE	Gizmos & Apparatuses	Camera 光锥
32	COLOR_CAMERAHORIZ	Gizmos & Apparatuses	Camera 地平线
33	COLOR_NEARRANGE	Gizmos & Apparatuses	Camera 的近范围
34	COLOR_FARRANGE	Gizmos & Apparatuses	Camera 的远范围
35	COLOR_LIGHTOBJ	Objects	Light 对象
36	COLOR_TARGETLINE	Gizmos & Apparatuses	目标线
37	COLOR_HOTSPOT	Gizmos & Apparatuses	Light Hotspot
38	COLOR_FALLOFF	Gizmos & Apparatuses	Light Falloff
39	COLOR_STARTRANGE	Gizmos & Apparatuses	Light Far Start
40	COLOR_ENDRANGE	Gizmos & Apparatuses	Light Far End
41	COLOR_VIEWPORTBKG	Main UI	视窗背景
42	COLOR_TRAJTICS1	Gizmos & Apparatuses	轨迹帧 1
43	COLOR_TRAJTICS2	Gizmos & Apparatuses	轨迹帧 2
44	COLOR_TRAJTICS3	Gizmos & Apparatuses	轨迹帧 3
45	COLOR_GHOSTBEFORE	Main UI	Ghost (Before)
46	COLOR_GHOSTAFTER	Main UI	Ghost (After)
47	COLOR_12FIELDGRID	Main UI	12-Field Grid
48	COLOR_STARTRANGE1	Gizmos & Apparatuses	Light Near Start
49	COLOR_ENDRANGE1	Gizmos & Apparatuses	Light Near End
50	COLOR_CAMERACLIP	Gizmos & Apparatuses	Camera Clip
51	COLOR_NURBSCV	Main UI	NURBS CV 点
52	COLOR_NURBSLATTICE	Main UI	NURBS 晶格点
53	COLOR_NURBSCP	Main UI	NURBS 的 Con 点
54	COLOR_NURBSFP	Main UI	NURBS 的 Fit 点
55	COLOR_NURBSDEP	Main UI	NURBS Dep 对象
56	COLOR_NURBSERROR	Main UI	NURBS Error 对象
57	COLOR_NURBSHILITE	Main UI	NURBS Hilite
58	COLOR_NURBSFUSE	Main UI	NURBS Fuse
59	COLOR_ENDEFFECTOR	Gizmos & Apparatuses	End Effector

(续表)

<index_Integer>	对应条款	所在用户界面	说明
60	COLOR_ENDEFFECTORSTRING	Gizmos & Apparatuses	End Effector 字符串
61	COLOR_JOINTLIMITSEL	Gizmos & Apparatuses	Selected Joint Limits
62	COLOR_JOINTLIMITUNSEL	Gizmos & Apparatuses	Joint Limits
63	COLOR_IKTERMINATOR	Gizmos & Apparatuses	IK Terminator
64	COLOR_SFUSER	Main UI	Safe Frame User
65	COLOR_VERTTICKS	Main UI	顶点记号
66	COLOR_XRAY	Main UI	See Through
67	COLOR_GROUPOBJ	Objects	组对象
68	COLOR_MANIPULATORX	Gizmos & Apparatuses	Transform Gizmo X
69	COLOR_MANIPULATORY	Gizmos & Apparatuses	Transform Gizmo Y
70	COLOR_MANIPULATORZ	Gizmos & Apparatuses	Transform Gizmo Z
71	COLOR_MANIPULATORACTIVE	Gizmos & Apparatuses	活动的 Transform Gizmo
72	COLOR_VPTCLIPPING	Main UI	视窗裁剪
73	COLOR_DECAYRADIUS	Gizmos & Apparatuses	Light 衰减开始
74	COLOR_VERTNUMBERS	Main UI	顶点号码
75	COLOR_CROSSHAIRCURSOR	Main UI	十字光标
76	COLOR_SVWINBK	其他视图	SV Window 背景颜色
77	COLOR_SVNODEBK	其他视图	SV Node 背景颜色
78	COLOR_SVSELNODEBK	其他视图	SV Selected Node 背景颜色
79	COLOR_SVNODEHIGHLIGHT	其他视图	SV Node Highlight
80	COLOR_SVMATERIALHIGHLIGHT	其他视图	SV Material Highlight
81	COLOR_SVMODIFIERHIGHLIGHT	其他视图	SV Modifier Highlight
82	COLOR_SVPLUGINHIGHLIGHT	其他视图	SV Plugin Highlight
83	COLOR_SVSUBANIMLINE	其他视图	SV Sub-Anim Line
84	COLOR_SVCHILDLINE	其他视图	SV Child Line
85	COLOR_SVFRAME	其他视图	SV Frame
86	COLOR_SVSELTEXT	其他视图	SV Selected Label
87	COLOR_SVTEXT	其他视图	SV Label
88	COLOR_UNSELTAB	Main UI	Unselected Tabs
89	COLOR_ATMOSAPPARATUS	Gizmos & Apparatuses	大气装置
90	COLOR_SUBSELECTIONHARD	Main UI	SubSelection Hard
91	COLOR_SUBSELECTIONMEDIUM	Main UI	SubSelection Medium
92	COLOR_SUBSELECTIONSOFT	Main UI	SubSelection Soft
93	COLOR_SVUNFOLDBUTTON	其他视图	SV Unfold 子按钮
94	COLOR_SVGEOMOBJECTBK	其他视图	SV Geometry Node 背景颜色

(续表)

<index_Integer>	对应条款	所在用户界面	说明
95	COLOR_SVLIGHTBK	其他视图	SV Light Node 背景颜色
96	COLOR_SVCAMERABK	其他视图	SV Camera Node 背景颜色
97	COLOR_SVSHAPEBK	其他视图	SV Shape Node 背景颜色
98	COLOR_SVHELPERBK	其他视图	SV Helper Node 背景颜色
99	COLOR_SVSYSTEMBK	其他视图	SV System Node 背景颜色
100	COLOR_SVCONTROLLERBK	其他视图	SV Controller Node 背景颜色
101	COLOR_SVMODIFIERBK	其他视图	SV Modifier Node 背景颜色
102	COLOR_SVMATERIALBK	其他视图	SV Material Node 背景颜色
103	COLOR_SVMAPBK	其他视图	SV Map Node 背景颜色

## 20.7 Material Editor

下面这些函数专用于 Material 和 Material Editor 窗口：

```
1. renderMap <textureMap> [ into:<bitmap> ]      \
                           [ size:<Point2> ]      \
                           [ filename:<String> ]   \
                           [ scale:<Float> ]      \
                           [ filter:<Boolean> ]   \
                           [ display:<Boolean> ]
```

在 Material Editor 窗口里调用 RenderMap 函数。返回值为一个 Bitmap 值，指定 <textureMap> 的渲染结果。如果指定了可选参数 into:，函数会将渲染图像存放在指定的位图文件中，新位图的尺寸和其他参数与指定位图相同。如果没有指定可选参数 into:，函数会用可选参数 size: 和 filename: 指定的尺寸和文件名创建一个新的 Bitmap 值。如果 into: 和 size: 都没有指定，位图默认大小为 [200,200]。

Scale: 可选参数为作用于 3D TextureMaps 的缩放系数，用来控制贴图在贴图坐标 UV 上的表面因子，这样可以控制贴图在 Bitmap 里的数量。

filter: 可选参数，布尔值，如果为 True，位图会被过滤，这是一个需要消耗较长时间的操作。默认值为 False。

display: 可选参数，布尔值，如果为 True，结果位图被显示在 VFB（虚拟帧缓存）里，否则不显示。默认值为 False。

例子：

```
rm = renderMap $foo.material.diffuseMap size:[640,480] \
fileName: "foodif.bmp"
save rm
close rm
```

上例把一个 Map 渲染成一个 Bitmap 值，并将它存在一个.bmp 文件里。

## 2. showTextureMap <material> <textMap> <Boolean>

控制着色视窗中纹理的可见性。指定的<material>为一个包含纹理贴图的材质，<textMap>指定了要控制的纹理贴图，<Boolean>控制是否显示指定<textMap>。

例子：

```
showTextureMap $foo.material $foo.material.diffuseMap on
tm = checker()
mat = standardMaterial diffuseMap:tm
mm = multimaterial()
mm[1] = mat
$box01.material = mm
showTextureMap mm[1] tm on
```

**注意** 对多维材质 (Multimaterial)，必须用[]索引指定子材质。

下面的 3ds max 系统变量用于 Material Editor 窗口。

currentMaterialLibrary

meditMaterials

sceneMaterials

有关说明参见 4.5.2 节。

## 20.8 轨迹视图 (Track View)

Track View 窗口函数在前面都必须加一个前缀 trackView，我们可以将这些函数理解成一个结构名为 trackView 的成员函数。它们一般要在在一个命名的 trackView 窗口下进行操作。在许多情况下，用户都是通过窗口名来识别 trackView 窗口。

### 1. trackView.open <name\_String>

打开指定名称的轨迹视图（如果已存在），或以指定名称创建一个新的轨迹视图（如果尚不存在），返回值为 True。

### 2. trackView.zoomSelected <name\_String>

缩放指定名称的轨迹视图里当前被选定的对象，被操作轨迹视图的 World 轨迹必须处于打开状态。Object 轨迹视图如果没有被打开，会被自动打开。层级显示会被上下翻滚，以确保被选择对象的轨迹处于窗口顶部。如果选择了多个对象，层级中被选中的第一个对象会位于窗口顶部。如果在指定轨迹视图里没有选择对象，不会发生任何操作，本方法返回值为 True，除非指定的轨迹视图不存在或没有被打开，此时返回 False。

### 3. trackView.close <name\_String>

关闭指定名称的轨迹视图。本方法返回值为 True，除非指定的轨迹视图不存在或没有被打开，此时返回 False。

### 4. trackView.numTrackViews()

返回已定义的命名轨迹视图的数目。

## 5. trackView.getTrackViewName &lt;index\_Integer&gt;

返回一个字符串，包含指定序号的轨迹视图的名称，索引从 1 开始。

## 6. trackView.setFilter &lt;name\_String&gt; {&lt;filter\_flag\_name&gt;} [#noRedraw]

## trackView.clearFilter &lt;name\_String&gt; {&lt;filter\_flag\_name&gt;} [#noRedraw]

上面两种方法用于为指定轨迹视图设置和清除过滤标记，以控制指定轨迹视图里哪些对象是可见的。返回值为 True，除非指定轨迹视图不存在或没有被打开，此时返回 False。用户在一次调用里可以指定任意数量的过滤标记，有效的过滤标记有：

- ◆ #all 设置或清除所有过滤标记
- ◆ #default 设置默认视图标记
- ◆ #selectedObjects 选择对象
- ◆ #selectedTracks 选择轨迹
- ◆ #animatedTracks 动画轨迹
- ◆ #spacewarpBindings 空间扭曲绑定
- ◆ #modifiedObjects 被施加 Modifier 的对象
- ◆ #transforms 转换
- ◆ #baseObjects 基对象
- ◆ #positionX X 轴坐标
- ◆ #positionY Y 轴坐标
- ◆ #positionZ Z 轴坐标
- ◆ #rotationX X 轴旋转
- ◆ #rotationY Y 轴旋转
- ◆ #rotationZ Z 轴旋转
- ◆ #scaleX X 轴缩放
- ◆ #scaleY Y 轴缩放
- ◆ #scaleZ Z 轴缩放
- ◆ #red 红色
- ◆ #green 绿色
- ◆ #blue 蓝色
- ◆ #controllerTypes 控制器类型对象
- ◆ #noteTracks
- ◆ #sound
- ◆ #materialsMaps 材质、贴图
- ◆ #materialParameters 材质参数
- ◆ #visibilityTracks 可见的轨迹
- ◆ #hideGeometry 隐藏 Geometry 对象
- ◆ #hideShapes 隐藏 Shape 对象
- ◆ #hideLights 隐藏 Light 对象
- ◆ #hideCameras 隐藏 Camera 对象

- ◆ #hideHelpers 隐藏 Helper 对象
- ◆ #hideSpacewarps 隐藏 Spacewarp 对象
- ◆ #visibleObjects 可见对象
- ◆ #position
- ◆ #rotation
- ◆ #scale
- ◆ #curveX
- ◆ #curveY
- ◆ #curveZ
- ◆ #staticValue
- ◆ #hierarchy
- ◆ #objects

下面的 3ds max 系统变量用于 Track View:

globalTracks

trackViewNodes

videoPostTracks

有关说明参见 4.5.2 节。

## 20.9 渲染场景 (Render Scene) 对话框

下面的方法可以存取 Render Scene 对话框里的各选项:

1. getRendApertureWidth()

获取当前渲染器靶区宽度, 以 mm 为单位。

2. getRendImageAspect()

获取当前渲染器的图像高宽比 (Aspect Ratio)。

3. getUseDraftRenderer()

如果默认渲染器为 Draft, 返回 True; 如果默认渲染器为 Production, 返回 False。

4. SetUseDraftRenderer <Boolean>

设定当前活动渲染器: Draft 或 Production。<Boolean>为 True, 使用 Draft 渲染器;  
<Boolean>为 False, 使用 Production 渲染器。

下面的 3ds max 系统变量用于 Render Scene 对话框:

renderer

renderDisplacements

renderEffects

renderHeight

renderPixelAspect

renderWidth

`rendOutputFilename`

`skipRenderedFrames`

有关说明参见 4.5.2 节。

下表中系统变量必须在 Render Scene 对话框没有打开时使用，否则对这些系统变量的修改不会刷新对话框的设置。

系统变量	数据类型	说明
<code>rendTimeType</code>	<code>Integer</code>	存取渲染时间范围类型，有效的类型值为： 1：单帧 2：当前活动的时间段 3：用户指定的时间范围 4：用户指定的时间范围字符串，如：“1,3,5-12”
<code>rendStart</code>	<code>Time</code>	存取渲染开始时间
<code>rendEnd</code>	<code>Time</code>	存取渲染结束时间
<code>rendNThFrame</code>	<code>Integer</code>	存取渲染时间间隔，最小值为 1
<code>rendShowVFB</code>	<code>Boolean</code>	存取是否将渲染结果显示在 VFB 里
<code>rendSaveFile</code>	<code>Boolean</code>	存取渲染器的 save file 标记
<code>rendUseDevice</code>	<code>Boolean</code>	存取渲染器的 use device 标记
<code>rendUseNet</code>	<code>Boolean</code>	存取渲染器的 use net 标记
<code>rendFieldRender</code>	<code>Boolean</code>	存取渲染器的 field render 标记
<code>rendColorCheck</code>	<code>Boolean</code>	存取渲染器的 color check 标记
<code>rendSuperBlack</code>	<code>Boolean</code>	存取渲染器的 super black 标记
<code>rendHidden</code>	<code>Boolean</code>	存取渲染器的 render hidden objects 标记
<code>rendForce2Side</code>	<code>Boolean</code>	存取渲染器的 force two-sided 标记
<code>rendAtmosphere</code>	<code>Boolean</code>	存取渲染器的 uses atmospheric effects 标记
<code>rendDitherTrue</code>	<code>Boolean</code>	存取渲染器的 dither True color 标记
<code>rendDither256</code>	<code>Boolean</code>	存取渲染器的 dither 256 color 标记
<code>rendMultiThread</code>	<code>Boolean</code>	存取渲染器的 multi-threaded 标记
<code>rendNThSerial</code>	<code>Boolean</code>	存取渲染器的 output file sequencing nth serial numbering 标记
<code>rendVidCorrectMethod</code>	<code>Integer</code>	存取视频颜色检测方法，有效值及对应的意义为： 1：FLAG 2：SCALE_LUMA 3：SCALE_SAT
<code>rendFieldOrder</code>	<code>Integer</code>	存取渲染顺序：1：偶数帧；2：奇数帧
<code>rendNTSC_PAL</code>	<code>Integer</code>	存取视频颜色检测采用的制式： 1：NTSC 制式；2：PAL 制式
<code>rendSuperBlackThresh</code>	<code>Integer</code>	存取 Super Black 颜色的阈值，取值范围为[0 255]
<code>rendFileNameBase</code>	<code>Integer</code>	存取 Render Scene 对话框的 Common Parameters 卷展栏里 File Number Base 项的设置
<code>rendPickupFrames</code>	<code>String</code>	存取 Render Scene 对话框的 Common Parameters 卷展栏里 Frames 文本框里的字符串

下面的 3ds max 系统变量也是用于 Renderer，但是与 Preference Settings 对话框的 Rendering 选项卡里的参数 Gbuffer Layers Maximum Number 对应：

`preferences.maximumGBufferLayers`

存取渲染过程中生成的 G-buffer 层的最小数目。

下面这些 3ds max 系统变量专用于 3ds max 默认 scanline A-Buffer renderer。如果当前渲染器不是 scanline A-Buffer renderer，这些变量返回值为 undefined。有关这些系统变量的说明请参见 4.5.2 节。

`scanlineRender.antiAliasFilter`

`scanlineRender.antiAliasFilterSize`

`scanlineRender.enablePixelSampler`

## 20.10 图解视图 (Schematic View)

图解视图窗口函数都放在一个名称为 schematicView 的结构包下。这些函数的操作对象为命名的图解视图窗口，而且在很多情况下，用户都通过该名称来识别图解视图窗口。

1. `schematicView.open <name_String>`

如果已存在<name\_String>指定的图解视图窗口，则打开该窗口；否则用<name\_String>创建一个新的图解视图窗口。返回值为 True。

2. `schematicView.zoomSelected <name_String>`

缩放指定图解视图窗口以全部显示当前选定的对象。如果没有选定对象，本函数不执行任何操作。本方法一般返回 True，但如果指定图解视图窗口不存在或还没有被打开，返回 False。

3. `schematicView.close <name_String>`

关闭指定图解视图窗口。本方法一般返回 True，但如果指定图解视图不存在或还没有被打开，返回 False。

4. `schematicView.numSchematicViews()`

返回已命名的图解视图窗口个数。

5. `schematicView.getSchematicViewName <index_Integer>`

返回字符串，包含指定序号位置图解视图窗口的名称。序号从 1 开始。

## 20.11 Time Configuration 对话框

下面的方法存取 Time Configuration 对话框里的设置：

1. `getKeyStepsPos()`

`setKeyStepsPos <Boolean>`

存取当 Key Mode Toggle 触发器打开时，是否跳到下一个 Position 关键帧。

2. `getKeyStepsRot()`

`setKeyStepsRot <Boolean>`

存取当 Key Mode Toggle 触发器打开时，是否跳到下一个 Rotation 关键帧。

3. `getKeyStepsScale()`

`setKeyStepsScale <Boolean>`

存取当 Key Mode Toggle 触发器打开时，是否跳到下一个 Scale 关键帧。

4. `getKeyStepsSelOnly()`

`setKeyStepsSelOnly <Boolean>`

存取当 Key Mode Toggle 触发器打开时，是否使用 Selected Objects Only 复选框的设置。

5. `getKeyStepsUseTrans()`

`setKeyStepsUseTrans <Boolean>`

存取当 Key Mode Toggle 触发器打开时，是否使用 Use Current Transform 复选框的设置。

下面的 3ds max 系统变量也是用于设置 Time Configuration 对话框：

`animationRange`

`frameRate`

`playActiveOnly`

`realTimePlayback`

上面几个系统变量的说明参见本书 4.5.2 节。

6. `timeConfiguration.playActiveOnly`

存取按下 Play 按钮时是否仅播放活动视窗。它包含了 Time Configuration 对话框里相应的复选框的设置。如果为 True 表示仅播放活动视窗，如果为 False，表示播放全部视窗。

7. `timeConfiguration.playbackSpeed`

存取播放速度，有效值为：

- ◆ 1 1/4x
- ◆ 2 1/2x
- ◆ 3 1x
- ◆ 4 2x
- ◆ 5 4x

8. `timeConfiguration.realTimePlayback`

存取是否在实时（Real Time）模式下播放动画。它包含了 Time Configuration 对话框里相应的复选框的设置。如果为 True 表示在实时模式下播放动画。

9. `timeConfiguration.useTrackBar`

存取 Time Configuration 对话框里 Key Steps/Use TrackBar 复选框的状态。

## 20.12 RAMPlayer

下面的方法与 RAMPlayer 播放器有关：

`RAMPlayer <filename1_String> <filename2_String>`

打开 RAMPlayer 并把文件<filename1\_String>作为通道 A, 文件<filename2\_String>作为通道 B, 用户可以将一个空字符（“”）作为文件名, 这样没有文件被加载至指定通道。

## 20.13 Track View Pick 对话框

Track View Pick 对话框以一种和 Track View 相类似的方式显示了 3ds max 场景的层级结构。用户可以从对话框选择一个或多个项目。下面的方法可以显示 Track View Pick 对话框:

```
trackView.pickTrackDlg [#multiple]
```

上面方法用来显示 Track View Pick 对话框, 如果用户选择某一轨迹并按 OK 按钮, 函数会返回一个 TrackViewPick 值, 如果按 Cancel 按钮, 返回 undefined。如果函数调用中包含可选参数#multiple, 可以在对话框中选择多个轨迹, 并返回一个元素类型为 TrackViewPick 值的数组。

TrackViewPick 为 MAXScript 的一种数据类型, 存储用户从 Track View Pick 对话框选择的结果。一个 TrackViewPick 值有下面的属性:

1. <trackViewPick>.name: String

被选择项目的名称, 和 Track View Pick 对话框里显示的一致。

2. <trackViewPick>.anim: subAnim

被选择项目的.subAnim 属性。

3. <trackViewPick>.client: MAXWrapper

被选择项目.subAnim 属性的所有者。如果其所有者为 MAXWrapper 的一个子类, 返回值为 MAXWrapper 值, 否则返回 undefined。

4. <trackViewPick>.subNum: Integer

被选择项目的.anim 在.client 里的索引号。

下面是一个例子:

```
--先创建一个 Sphere 对象并对它施加一个 Bend_Modifier。
tvp=trackview.pickTrackDlg()
--pick objects->Sphere01->Modified object->Bend->Angle
tvp.anim          --返回 SubAnim:Angle
tvp.client        --返回 Bend:Bend
tvp.subNum        --返回 3
tvp.name          --返回 "Angle"
tvp.client[tvp.subNum] --返回 SubAnim:Angle
```

## 20.14 选择场景对象

下面讲述了几种选取场景对象的方法。

### 20.14.1 点击选取场景对象

```
pickObject [ message:<String> ] [ prompt:<String> ] \
    [ count:n|#multiple] [ filter:fn ] [ select:<Boolean> ]
```

函数功能：让用户在 3ds max 视窗中用鼠标点击来选择一个或多个场景对象。

它的几个可选参数说明如下：

- ◆ **message:** 数据类型为 String，指定一个提示信息，显示在 3ds max 界面的信息提示栏（Status Bar Prompt）里；
- ◆ **prompt:** 数据类型为 String，指定一个提示信息，显示在 Listener 窗口里；
- ◆ **count:** 为正整数或符号#multiple，以指示一次可以选取多少个对象（默认值为 1），如果 count 变量大于 1 或为#multiple，返回值为包含所有被选对象的数组。符号#multiple 表示可以选择任意数量的对象，用户可以通过右击鼠标来中止选择，如果 Listener 窗口正处在活动状态，也可以通过按下键盘上任意字符来中止选择。如果 Listener 窗口正处在活动状态时按下 Esc 键，函数返回值为#escape；如果视窗正处在活动状态时按下 Esc 键，对象选择被中止，返回值为前面已选择的对象。如果 count 被指定为#multiple，但选择中止时没有选择任何对象，返回值为一个空数组；如果 count: 不为#multiple，且选择中止时没有选择任何对象，返回值为 undefined；
- ◆ **filter:** 为一个 MAXScript 函数，它只有一个变量。当鼠标停留在某一场景对象上时，系统调用该函数，并将鼠标停留下的场景对象作为参数传递给该函数。如果场景对象符合选择条件，函数返回 True，否则返回 False。一般而言，这种函数会包含一些对象类型检测语句，如下例：

```
fn shapeFilt o = superClassOf o == Shape
```

这样我们可以将它用于下面的 pickObject() 函数：

```
pickObject prompt: "enter a shape" filter:shapeFilt
```

- ◆ **select:** 控制是否将被选择对象变成 3ds max 场景里当前选择集。如果指定 True，当前选择集变成刚刚选择的对象，默认值 False 表示刚刚选择的对象不会影响当前选择集。

### 20.14.2 用对象名选择场景对象

可以用 selectByName() 函数来打开一个标准的 Select By Name 对话框，让用户在对话框里选择对象，然后返回一个元素类型为 MAXScript 对象的数组。其格式为：

```
selectByName [ title:<String> ] [ buttonText:<String> ] \
    [ filter:<fn>] [ showHidden:<Boolean> ] [ single:<Boolean>]
```

其可选参数详细说明如下：

- ◆ **title:** 数据类型为 String，指定对话框窗口标题；
- ◆ **buttonText:** 数据类型为 String，指定对话框里 Accept 按钮的标签，默认值为 Select；
- ◆ **filter:<fn>** 指定一个过滤函数，用来决定哪些对象显示在对象列表中。它和

MAXScript 里其他地方的过滤函数类似。它有一个变量，用来检测是否包含当前对象，如果函数返回 True，在对象列表中包含当前对象，如果函数返回 False，在对话框的对象列表中排除当前对象。如果没有指定过滤器，列表中显示所有对象。例如下面的函数仅仅在列表中显示 Shape 类对象：

```
fn shape_filt obj = isKindOf obj Shape
```

- ◆ showHidden:<Boolean>控制是否显示那些被隐藏和冻结的对象，默认值为 False；
- ◆ single:<Boolean>控制本次选择可以选择单个还是多个对象。默认值为 False，表示可以选择多个对象，返回值为包含所选对象的数组。如果指定 single:True，用户仅可以选择单个对象，返回值为该对象，而不是一个数组。

如果用户按 Cancel 退出对话框，函数返回值为 undefined。

### 20.14.3 用区域来选择场景对象

下面方法允许用户基于对象在视窗中的位置来选择一个或多个对象。在本方法中使用的坐标值为 Windows 格式，原点为左上角。可以用方法 gw.getWinSizeX() 和 gw.getWinSizeY() 来判定当前视窗的大小（以像素为单位）。用这些方法拾取的对象会被 3ds max 的 Main Toolbar 里的 Selection Filter 自动进行过滤。

1. boxPickNode <box2> [crossing:<Boolean>]

返回一个数组，表示位置在矩形区域内部的对象集。

- ◆ 参数<box2>指定活动窗口里矩形区域的两个角点坐标（以像素为单位）。
- ◆ 参数 crossing: 如果为 True，那些仅仅部分处于矩形区域内的对象也会被拾取，如果为 False，只有完全处于矩形区域内的对象才会被拾取。默认值为 True。如：

```
boxPickNode(box2 [0,0] [1000,1000])
```

2. circlePickNode <box2> [crossing:<Boolean>]

返回一个数组，表示位置在圆形区域内部的对象集。

- ◆ 参数<box2>分别指定活动窗口里圆形区域的圆心和该圆形上的一点（以像素为单位）。
- ◆ 参数 crossing: 同上一条。
- ◆ 参数<Box2>的值必须用一种特殊的方式建立，如下面脚本所示：

```
vpCenter =(Point2(gw.getWinSizeX())(_gw.getWinSizeY()))/2
vpQuarter = Point2(vpCenter.x/2)vpCenter.y
circleRegion = box2 0 0 0 0 --初始化变量 circleRegion 为<box2>值
circleRegion.left = vpCenter.x --圆心
circleRegion.bottom = vpCenter.y
circleRegion.right = vpCenter.x/2 --圆上一点
circleRegion.top = vpCenter.y
for obj in(circlePickNode circleRegion crossing:False)\n
do print obj.name
```

上面脚本会打印指定圆形区域内所有场景对象的.name 属性。圆形区域的圆心为视窗的中心，圆上一点位于离圆心 1/4 视窗宽度的地方。

### 3. fencePickNode <Point2\_array> [crossing:<Boolean>]

返回一个数组，表示位置在指定围栏区域内部的场景对象集。围栏区域的各边界点由数组<Point2\_array>指定，可选参数 crossing:的默认值为 True。

## 20.15 提示信息框和询问对话框

下面函数可以在 MAXScript 里显示一个信息框或 Yes/No 询问对话框：

### 1. messageBox <message\_String> [ title:<window\_title\_String> ] [ beep:<Boolean> ]

显示一个有模式信息框，包含指定的信息字符串<message\_String>和一个 OK 按钮。

◆ title:为可选参数，设定信息框标题；

◆ beep:为可选参数，设定显示信息框时是否伴随一个蜂鸣声。默认值为 True。

### 2. queryBox <message\_String> [ title:<window\_title\_String> ] [ beep:<Boolean> ]

显示一个类似于 messageBox() 函数创建的有模式信息框，但它包含 Yes 和 No 两个按钮。如果用户按 Yes 按钮，函数返回 True；按 No 按钮，函数返回 False。

### 3. yesNoCancelBox <message\_String> [ title:<window\_title\_String> ] [ beep:<Boolean> ]

显示一个类似于 messageBox() 函数创建的有模式信息框，但它包含 Yes、No 和 Cancel 三个按钮。如果用户按这三个按钮退出信息框，函数分别返回 Name 类值#yes、#no、#cancel。

例子：

```
messageBox "You shouldn't have done that"
if queryBox "Do you want to continue? " beep:False then ...
```



在某些情况下，脚本中处在包含以上信息框函数调用语句后面的程序语句可能会在您预料的运行时间之前被执行，请看下面脚本：

```
for _t=0 to 3 do
(
    messagebox "Press Me"
    format "_t= %\n" _t
)
print "Should print last"
在 Listener 窗口里运行结果如下:
_t= 0
"Should print last"
"Should print last"
_t= 1
_t= 2
_t= 3
```

而 MessageBox 被显示四次，每次都在变量 \_t 被输出到 Listener 窗口里之前。其原因为 Listener 是在后台进程里查找表达式来编译、运行。在上例中，有两个表达式：一个 for 循环和最后的 print 表达式。messageBox() 函数会等待用户的反应，系统将在 Main UI 进程里处理用户的反应，但编译器会继续往下执行打印语句 should print last。

为了避免上面这种情形出现，用户需要将上面脚本放在一对括号()里，这样它们就变成了一个表达式，而不是两个了。

## 20.16 其他对话框

### 1. exclusionListDlg()

为 Light 类对象显示 Exclude/Include 对话框，返回被选择到 Exclude/Include 列表中的对象名称数组。

### 2. configureBitmapPaths()

本方法显示 Configure Bitmap Paths 对话框，让用户指定 Bitmap 装载路径。如果用户按 Cancel 退出对话框，返回 False，否则返回 True。

### 3. materialBrowseDlg [#mats] [#maps] [#incNone] [#instanceOnly]

本方法显示材质浏览器 Material/Map Browser。如果没有拾取 Material 或 TextureMap，返回 False，否则返回被拾取 Material 或 TextureMap 的一个实例。

参数说明如下：

- ◆ #mats 仅显示 Material；
- ◆ #maps 仅显示 TextureMap；
- ◆ #incNone 将值 None 作为一个 Material 或 TextureMap；
- ◆ #instanceOnly 如果选择的 Material 或 TextureMap 已在场景中存在，返回值为一个 Instance 副本，否则会显示一个对话框让用户选择建立一个 Copy 类副本或 Instance 类副本。

如果#mats 或#maps 都没有指定，Material 和 TextureMap 都会被显示；如果#mats 或#maps 都被指定，仅显示 TextureMap。

### 4. mtlBrowser.browseFrom [#mtlLibrary | #mtlEditor | #activeSlot | #selected | #scene | #new]

为 Material 浮动窗口设置材质浏览源。

### 5. nodeColorPicker()

显示 Object Color Picker 对话框，返回一个 Color 值，包含刚拾取到的颜色。

### 6. OSnap()

显示 Grid and Snap Settings 对话框。

### 7. USetup()

显示 Units Setup 对话框。

## 20.17 键盘输入

下面的键盘输入函数用于在 MAXScript 里与 Listener 或 Mini Listener 窗口（即主界面左下角的脚本控制区）进行一些简单的交互。如果 Listener 窗口处于关闭状态，所有键盘输入将从 Mini Listener 窗口接收，但如果 Listener 窗口处于打开状态，所有键盘输入将从 Listener 接收。

### 1. getKBValue [ prompt:<String> ]

让用户在 Listener 窗口里输入一个 MAXScript 值。可选参数 prompt: 为显示在 Listener 窗口的提示信息。可以输入任何合法的 MAXScript 表达式，它们被求值，其结果被作为 getKBValue() 函数的返回值。输入表达式可以为下面类型的字面常量：Integer, Float, String, Name, Array, Point2, Point3 和场景对象路径名，也可以包含数学运算和函数（包括自定义函数）、系统变量等。用户可以用 MAXScript 的 classOf 函数来检测返回值的数据类型。

例如：

```
v = getKBValue prompt: "Enter object count: "
if classOf v != Integer then
print "Value entered must be an Integer"
```

用户可以用 Esc 键来中止输入，此时函数返回值为#escape。

### 2. getKBLine [ prompt:<String> ]

#### getKBChar [ prompt:<String> ]

让用户在 Listener 窗口里输入一些字符，并将它们作为一个字符串（String）返回。

getKBLine() 返回 ENTER 键之前的所有字符，而 getKBChar() 每输入一个字符，马上将它返回。用户可以用 ESC 键来中止输入，此时函数返回值为#undefined。

### 3. getKBPoint [ prompt:<String> ]

让用户在 Listener 窗口里输入一个三维点（Point3），其输入语法与 pickPoint() 函数相同，请读者参考上面有关 pickPoint() 函数的说明。输入点被视为相对于当前活动 Grid 构造平面，返回值为一个在 World 空间里的 Point3 值。

下面系统变量与键盘有关，其说明请参见 4.5.2 节：

- ◆ keyboard.shiftPressed
- ◆ keyboard.controlPressed
- ◆ keyboard.altPressed

## 20.18 3ds max 系统路径

下面这些方法用来存取 3ds max 系统路径。

### 1. GetDir <filetype\_name>

存取 3ds max 系统路径，返回一个字符串，包含由 Customize | Configure Paths 对话框

里指定的系统路径。有效的<filetype\_name>有：#autoback、#drivers、#export、#expression、#font、#help、#image、#import、#matlib、#maxroot、#maxstart、#plugcfg、#preview、#scene、#scripts、#sound、#startupScripts、#ui、#vpost。

用户可用下面的函数获取、添加或删除 Bitmap 和 Xref 路径，对应 3ds max 里 Configure Paths 对话框里 Bitmaps 和 XRefs 选项卡。通过下面函数所作的路径变化会马上反映到 3dsmax.ini 文件里，这样这些变化是持久的。

2. mapPaths.add <path\_String>

将指定路径添加到 Bitmap 搜索路径清单里。

3. mapPaths.count()

返回已经定义的 Bitmap 搜索路径数量。

4. mapPaths.get <index>

返回指定索引位置的 Bitmap 搜索路径，数据类型为 String，索引从 1 开始。

5. mapPaths.delete()<index>

删除指定索引位置的 Bitmap 搜索路径。索引从 1 开始。

6. sysinfo.getSystemMemoryInfo()

返回一个元素个数为 7 的数组，包含系统内存状态数据。

例如：

```
(  
r=sysinfo.getSystemMemoryInfo()  
for i=2 to 7 do r[i] /=(1024*1024.)  
format "percent of memory in use:\t%\n" r[1]  
format "total physical memory:\t% MB\n" r[2]  
format "free physical memory:\t% MB\n" r[3]  
format "used physical memory:\t% MB\n" (r[2]-r[3])  
format "total paging file size:\t% MB\n" r[4]  
format "free paging file size:\t% MB\n" r[5]  
format "used paging file size:\t% MB\n" (r[4]-r[5])  
format "total virtual memory:\t% MB\n" r[6]  
format "free virtual memory:\t\t% MB\n" r[7]  
format "used virtual memory:\t\t% MB\n" (r[6]-r[7])  
ok  
)
```

Listener 窗口返回值：

```
percent of memory in use: 0  
total physical memory: 255.359 MB  
free physical memory: 16.5156 MB  
used physical memory: 238.844 MB  
total paging file size: 1016.3 MB  
free paging file size: 757.898 MB  
used paging file size: 258.398 MB  
total virtual memory: 2047.88 MB  
free virtual memory: 1846.55 MB  
used virtual memory: 201.328 MB
```

ok

## 7. sysinfo.getMAXMemoryInfo()

返回一个元素个数为 9 的数组，包含 3ds max 内存状态数据。

例如：

Listener 窗口返回值：

```
Page Fault Count: 32948
Peak Working Set Size: 70.3594 MB
Working Set Size: 70.3594 MB
Quota Peak Paged Pool Usage: 0.166186 MB
Quota Paged Pool Usage: 0.161236 MB
Quota Peak NonPaged Pool Usage: 0.0213509 MB
Quota NonPaged Pool Usage: 0.0213509 MB
Pagefile Usage: 58.9023 MB
Peak Pagefile Usage: 58.9219 MB
```

#### 8. xrefPaths.add <path\_String>

将指定路径添加到 XRef 搜索路径清单里。

## 9. xrefPaths.count()

返回已经定义的 XRef 搜索路径数量。

## 10. xrefPaths.get <index>

返回指定索引位置的 XRef 搜索路径，数据类型为 String，索引从 1 开始。

## 11. xrefPaths.delete <index>

删除指定索引位置的 XRef 搜索路径。索引从 1 开始。

## 20.19 3ds max 场景文件属性

3ds max 场景文件属性可以用本节讲述的方法在 MAXScript 进行存取。场景文件属性被组织成三个集合，这三个集合直接和 File Properties 对话框里的三个选项卡对应，它们是：

- ◆ #summary: 包含 Summary 选项卡里的 Title、Subject、Author、Keywords、Comments;
- ◆ #contents: 包含 Contents 选项卡里的 Manager、Company、Category 和一个包含所有标题的数组（包含了如 General、Mesh Totals 等内容）。注意 Contents 选项卡里的内容只有在文件被存盘后才会被刷新。用户可以执行一个 max hold 命令，它会强制使场景文件进行保存，这样 Contents 选项卡里的内容会被刷新；
- ◆ #custom: 包含 Custom 选项卡里所有内容。

下面的方法用来对上面集合内容进行存取，其中变量<set\_name>可以为上面三个集合值的任意一个：

1. fileProperties.getNumProperties <set\_name>

返回一个整数，表示指定集合里所有的属性数量，例如：

```
numProps=fileProperties.getNumProperties #summary
```

2. fileProperties.getPropertyName <set\_name> <index>

返回一个字符串，表示包含指定集合里指定索引的属性名。例如：

```
nameProp=fileProperties.getPropertyName #summary 1
```

3. fileProperties.getPropertyValue <set\_name> <index>

返回指定集合里指定索引的属性值。目前在 3ds max 的 File Properties 对话框里支持的五种数值类型如下，在方括号“[ ]”里的说明为该数值类型与 MAXScript 里相对应的数值类型：

- ◆ Text [String]
- ◆ Date [String]: 字符串里包含日期，如“03/23/99”
- ◆ Number [Integer 或 Float]
- ◆ Yes/No [Boolean]
- ◆ Headers [String 类数组]

其中 Headers 为包含显示在 Contents 选项卡里的各项如 General、Mesh Totals 等内容的一个数组。比如我们可以用下面方法获取 Contents 选项卡里的一个 Headers 数组：

```
fileProperties.getPropertyValue #contents 1
```

返回值为：#(“General”，“Mesh Totals”，“Scene Totals”，“External Dependencies”，“Objects”，“Materials”，“Used Plug-Ins”，“Render Data”）。

4. fileProperties.getItems <header\_String>

返回一个 Headers 数组，包含指定 header 里包含的所有项目，比如：

```
fileProperties.getItems "Mesh Totals"
```

返回值为：#(“Vertices: 488”，“Faces: 968”）。

5. fileProperties.findProperty <set\_name> <prop\_name\_String>

返回与指定集合名和属性名（String）对应的属性索引号，如果没有找到，返回 0。比如：

```
fileProperties.findProperty #custom "BoolVal"
```

## 6. fileProperties.addProperty &lt;set\_name&gt; &lt;prop\_name\_String&gt; &lt;prop\_value&gt; [#date]

向指定集合里添加一个新属性，其属性名和属性值分别为<prop\_name\_String>和<prop\_value>，属性值数据类型可以为上面所列除 Array 之外的几种，因为用户不能向 Contents 选项卡里添加内容。

如果<prop\_value>为包含一个日期的字符串，必须指定可选变量#date。例如：

```
fileProperties.addProperty #custom "DateVal" "03/23/99" #date
```

将向 Custom 选项卡里添加一个名为 DateVal 的日期型属性。

## 7. fileProperties.deleteProperty &lt;set\_name&gt; &lt;prop\_name\_String&gt;

删除指定集合里的指定属性。

下面的例子会按层级模式打印当前场景的所有文件属性：

```
--添加属性
fileProperties.addProperty    #summary    "Title"      "Title val"
fileProperties.addProperty    #custom     "IntVal"     30
fileProperties.addProperty    #custom     "FloatVal"   30.034
fileProperties.addProperty    #custom     "BoolVal"    True
fileProperties.addProperty    #custom     "DateVal"    "03/23/99" #date
--
--执行一个场景 hold 命令来刷新 Contents 集合
max hold
--
--获取所有属性
pages = #(#summary, #contents, #custom)
for pg in pages do
  (--for pg 开始
    format "----% ---\n" (pg as String)
    for i=1 to(fileProperties.getNumProperties pg)do
      (--for i 开始
        local pname =(fileProperties.getPropertyName pg i)
        local pval =(fileProperties.getPropertyValue pg i)
        format "\t%:" pname
        if(pname == "Headers")then
          (--if 开始
            format "\n"
            for hdr in pval do
              (--for hdr 开始
                format "\t\t%\n" hdr
                local docs = fileProperties.getItems hdr
                if docs != undefined then
                  for d in docs do format "\t\t\t%\n" d
                )--for hdr 结束
              )--if 结束
            else format "%\n" pval
          )--for i 结束
        )--for gp 结束
      )--for i 结束
    )--for pg 结束
  )--for pages 结束
```

# 第21章 在MAXScript里存取文件

## 21.1 3ds max场景文件的装载和保存

下面方法用于装载、保存、合并、导入、导出3ds max场景文件：

1. `loadMaxFile <filename_String>`

装载指定文件。如果装载成功，返回True。

2. `mergeMAXFile <filename_String> [ <name_array> ] [ #prompt ] \`

`[ #select ] #noRedraw ] \`

`[ #deleteOldDups | #mergeDups | #skipDups | #promptDups ]`

将指定文件合并到当前场景。函数返回值：如果文件被找到并成功加载，返回True。

如果用户没有指定一个完整的路径，系统会先在3ds max里执行路径，然后在系统环境变量PATH指定的路径列表里搜索。

除合并文件名<filename\_String>外，其他参数都是可选的。标记参数可以按任意顺序排列。各可选参数说明如下：

- ◆ <name\_array> 元素数据类型为Name或String的数组。指定源场景里哪些对象要被合并，如果本参数未指定，合并所有对象；
- ◆ #prompt 如果指定该参数，打开标准merge对话框；
- ◆ #select 如果指定该参数，新合并的对象在合并后被选择；
- ◆ #noRedraw 如果指定该参数，推迟屏幕刷新。如果用户要连续几次调用mergeMAXFile函数，为了节约时间，可以指定本参数，这样直到最后一次合并才刷新屏幕；
- ◆ #deleteOldDups 如果合并对象中有与场景中同名的对象，删除场景中同名的对象。相当于一种替代；
- ◆ #mergeDups 忽略命名冲突。这样可能会导致场景中存在两个重名的对象；
- ◆ #promptDups 显示Duplicates Resolution对话框供用户选择。

3. `getMAXSaveFileName filename: <seed_filename_String>`

存储文件，显示标准File Save对话框，返回文件名或undefined（如果按Cancel退出对话框）。

4. `getMAXOpenFileName filename: \`

`<seed_filename_String> dir:<seed_directory_String>`

打开文件，显示标准File Open对话框，返回文件名或undefined（如果按Cancel退出对话框）。

5. `getMAXFileObjectNames <max_filename_String>`

返回一个元素类型为 `Name` 的数组，每一个元素为指定场景文件中的一个对象名。这样可以让用户在另一个场景文件里建立指定文件包含对象的预览（按对象名字），让用户在使用 `mergeMAXFile()` 函数时选择要合并的对象。如：

```
p=[1000,1000,1000]
for i = 1 to 5 do box pos:(random p -p)--创建一些 Box 对象
savemaxfile "mergetest.max" --文件保存
for obj in objects do obj.name = "_" +obj.name --修改 Box 对象名称
objects.pos += [0,-1000,0] --移动对象
--从文件里获取对象名
fobj_names = getmaxfileobjectnames "mergetest.max"
deleteitem fobj_names 3 --从数组里删除第三个对象的名字
mergemaxfile "mergetest.max" fobj_names #select --合并对象并选择它们
selection.count --返回 4
objects.count --返回 9
```

6. `saveMaxFile <filename_String>`

存储场景文件。如果文件名`<filename_String>`没有指定路径，文件被存储在 3ds max 的 `Scene` 路径下，如果没有指定扩展名，系统自动给它添加扩展名`.max`。

7. `holdMaxFile()`

等同于 3ds max 菜单操作命令 `Edit | Hold`。

8. `fetchMaxFile()`

等同于 3ds max 菜单操作命令 `Edit | Fetch`。

9. `importFile <filename_String> [ #noPrompt ]`

导入指定文件。导入文件类型取决于文件名后缀，如：`.dx`、`.3ds`。

如果指定`#noPrompt` 标记，在导入过程中不会显示任何配置或控制对话框，而使用默认配置。

10. `exportFile <filename_String> [ #noPrompt ]`

导出到指定文件。导入文件类型取决于文件名后缀，如：`.dx`、`.3ds`。

如果指定`#noPrompt` 标记，在导出过程中不会显示任何配置或控制对话框，而使用默认配置。

11. `saveNodes <node_collection> <filename_String>`

用指定文件名创建一个新的`.max` 文件，并将指定 `Node` 集合存储在其中。参数`<node_collection>`可以为单个 `Node`，也可以为 `Node` 数组或路径通配符、3ds max 对象集如 `selection` 或 `lights`，还可以为`<node>.children` 数组。如果`<filename_String>`没有指定扩展名，系统自动给文件添加扩展名`.max`。

12. `checkForSave()`

如果场景文件自上次存盘操作后又被修改过，调用本函数会显示一个信息框提示用户场景已被修改过，需要保存。如果用户按 `Cancel` 退出对话框，返回 `False`，否则返回 `True`。

## 21.2 与 Bitmap 文件有关的方法

下面的方法用来收集场景中的 Bitmap 文件。与 Bitmap 值有关的属性、方法，包括 Bitmap 文件的读、写，请参见本书 3.2.18 节。

```
enumerateFiles [ <maxwrapper_obj> ] <function> [ <arg> ] \
[ #inactive ] [ #videoPost ] [ #render ] [ #missing ] \
[ #localOnly ]
```

让用户遍历在当前场景中全部对象或某一单独对象中用到的所有 Bitmap 文件，并对这些 Bitmap 文件进行处理。也可以为其设置过滤器，这样本函数仅处理指定的 Bitmap 文件，比如仅仅处理用于 Video Post 或 Renderer 的 Bitmap 文件。

enumerator 的工作流程为：每找到一个满足过滤条件的 Bitmap 文件，就调用一次指定函数<function>。下面详细列出了函数的参数，其中只有<function>一个为必须的参数：

- ◆ <maxwrapper\_obj> 一个 3ds max 对象，如一个场景对象或 Material。指定本参数使本函数仅考虑与指定对象有关的 Bitmap 文件。有关更多信息，参见#localOnly 开关；
- ◆ <function> 函数<function>会被每一个找到的 Bitmap 文件调用，它必须有一个或两个参数，第一个参数为文件名字符串；
- ◆ <arg> 如果有指定，为函数<function>的第二个参数；
- ◆ #inactive 表示包含那些不活动的 Bitmap 文件；
- ◆ #videoPost 表示包含那些用在 Video Post 里的 Bitmap 文件；
- ◆ #render 表示包含那些在渲染时用到的 Bitmap 文件；
- ◆ #missing 表示包含那些在文件系统里找不到的 Bitmap 文件。如果没有指定任何过滤标记，表示遍历所有的 Bitmap 文件；
- ◆ #localOnly 本参数必须与<maxwrapper\_obj>参数连用。如果指定本标记，表示仅遍历那些直接用于指定对象的 Bitmap 文件；如果没有指定，表示遍历所有与指定对象相关的 Bitmap 文件。

例如：

```
function get_names name a = append a name
files = #()
enumerateFiles get_names files #missing
```

下面的例子会把当前场景文件中使用的 Bitmap 文件排序后打印出来。

```
(  
    local mapfiles=#()  
    fn addmap mapfile =  
    (local mapfileN=mapfile as name  
        local index=finditem mapfiles mapfileN  
        if index == 0 do append mapfiles mapfileN
```

```

)
enumeratefiles addmap
sort mapfiles
for mapfile in mapfiles do print(mapfile as String)
)

```

如果把第 8 行改成：

```
enumeratefiles addmap #missing
```

则只有那些在文件系统里找不到的 Bitmap 被打印。

## 21.3 标准文件打开、存储对话框

下面方法显示标准 3ds max 的 File | Open、File | Save 或 Folder Selection Browser 对话框：

```

1. getOpenFileName [ caption:<title> ] \
    [ filename:<seed_filename_String> ] \
    [ types:<description1>|<pattern1>|<description2>|<pattern2>|...| ]
getSaveFileName [ caption:<title> ] \
    [ filename:<seed_filename_String> ] \
    [ types:<description1>|<pattern1>|<description2>|<pattern2>|...| ]

```

上面两函数都返回一个包含完整路径名的文件名或值 undefined（如果用户按 Cancel 退出对话框）。可选参数 types: 用来指定文件类型和后缀，其使用一种特殊的格式：

<description1>|<pattern1>|<description2>|<pattern2>|...|

例如：

```
f = getOpenFileName \
types: "Data(*.dat)|*.dat|Excel (*.csv)|*.csv|All|*.*|"
```

为对话框的 File type 下拉列表指定了三种文件类型。

```
2. getSavePath [ caption:<window_caption_String> ]
```

本函数显示一个 Folder Selection Browser 对话框供用户选择一个文件夹。如果用户选择了文件夹，返回该文件夹的字符串；如果按 Cancel 退出，返回 undefined。

## 21.4 文件名提取

1. filenameFromPath <filename\_String>

返回包括完整路径的文件名及扩展名。

2. getFilenamePath <filename\_String>

返回指定完整文件名<filename\_String>的路径部分。

3. getFilenameFile <filename\_String>

返回指定完整文件名<filename\_String>的名称部分。

## 4. getFileType &lt;filename\_String&gt;

返回指定完整文件名<filename\_String>的扩展名部分。

## 5. doesFileExist &lt;filename\_String&gt;

如果指定文件在磁盘中存在，返回 True，否则返回 False。

例如：

```
file="g:\\subdir1\\subdir2\\myImage.jpg"
filenameFromPath file --返回: "myImage.jpg"
getFilenamePath file --返回: "g:\\subdir1\\subdir2\\"
getFilenameFile file --返回: "myImage"
getFileType file --返回: ".jpg"
```

## 21.5 外部文件方法

## 1. getFiles &lt;wild\_card\_filename\_String&gt;

返回一个数组，包含与路径通配符<wild\_card\_filename\_String>匹配的所有文件名。下面例子先获取路径 c:\foo 下面所有.max 场景文件，然后对数组元素循环，打印每一文件里的对象：

```
files = getFiles "c:\\foo\\*.max"
for f in files do(loadMAXFile f; print objects)
```

getFiles()函数也可用来判断一个文件是否存在。请看下面的自定义函数，如果指定文件在磁盘中存在，函数返回 True。

```
fn existFile fname =(getfiles fname).count != 0
```

## 2. getDirectories &lt;wild\_card\_directory\_name\_String&gt;

返回一个数组，包含与路径通配符匹配的所有路径。

## 3. makeDir &lt;directory\_path\_String&gt;

用指定名称创建一个新的路径。如果创建成功，返回 True，否则返回 False。

## 4. deleteFile &lt;filename\_String&gt;

删除指定文件。如果该文件正被 MAXScript 打开，将无法删除。如果删除成功，返回 True，否则返回 False。

## 5. renameFile &lt;old\_filename\_String&gt; &lt;new\_filename\_String&gt;

将指定旧文件名改为新文件名。也可用来将文件在不同路径之间移动。如果旧文件正被 MAXScript 打开或新文件名已存在，将无法更名。如果更名成功，返回 True，否则返回 False。

## 6. copyFile &lt;existing\_filename\_String&gt; &lt;new\_filename\_String&gt;

文件复制。如果新文件名已存在或旧文件正被 MAXScript 打开，复制将会失败。如果复制成功，返回 True，否则返回 False。

## 7. getFileAttribute &lt;filename\_String&gt; &lt;attribute&gt;

```
setFileAttribute <filename_String> <attribute> <Boolean>
```

存取与文件有关的属性。GetFileAttribute()函数返回 True 或 False，取决于指定属性的状态；

`setFileAttribute()`函数每次只能改变一个属性的状态（其他属性仍然保持原来的状态），有效的`<attribute>`有：`#readOnly`、`#hidden`、`#system`、`#directory`、`#archive`、`#temporary`、`#normal`。

#### 8. `getFileModDate <filename_String>`

返回一个字符串，包含指定文件的修改日期，如：“1/29/99 1:52:05 PM”。

#### 9. `getFileCreateDate <filename_String>`

返回一个字符串，包含指定文件的创建日期。

#### 10. `getFileVersion <filename_String>`

返回指定文件的产品版本号，如果指定文件包含数据为未知，返回 `undefined`。通常文件类型为可执行文件或应用扩展文件（如.dll 文件）。例如：

```
GetFileVersion "g:\\3ds max25\\3ds max.exe" --返回"2,5,0,0 2,5,0,0"
```

例子：

```
for f in getFiles "3ds max\\maps\\*.jpg" do deleteFile f
for d in getDirectories "D:\\foo\\*" do
    for f in getFiles(d + ".*")do
        copyFile f("C:\\temp\\\" + getFilenameFile f + getFilenameType f)
if(getFiles "foo.max").count == 0 then print "File missing"
```

## 21.6 加密文件

#### 1. `encryptFile <in_filename_String> <out_filename_String> <key_Integer>`

用指定整数密码`<key_Integer>`给文件`<in_filename_String>`加密，生成加密文件`<out_filename_String>`。

#### 2. `openEncryptedFile <filename_String> <key_Integer>`

用指定整数密码`<key_Integer>`打开加密文件`<filename_String>`，并返回一个 `FileStream` 值，用户可对它进行读操作（和用函数 `openFile()` 打开的文件一样）。

`encryptFile()` 和 `openEncryptedFile()` 让用户用自己的密码来加密一个文本文件，然后再用这个密码来打开它读取数据。有时我们可以将系统变量 `hardwareLockID` 写入一个文件，然后读取它，这样可以判别用户是否获得程序的授权。下面例子创建一个包含程序授权码的文本文件：

```
f = createFile "lock.tmp"
format "%" hardwareLockID to:f
close f
encryptFile "lock.tmp" "lock.dat" 5476557
deleteFile "lock.tmp"
```

下面的代码用来读取并检测 lock ID：

```
f = openEncryptedFile "lock.dat" 5476557
id = readValue f
close f
if id != hardwareLockID then
(
```

```

message "Lock ID's don't match"
return 0
)

```

## 21.7 存取.INI 文件

下面方法让用户可以读写.INI 文件的 Key 值：

1. `getINISetting <filename_String> <section_String> <key_String>`

读取指定文件的 INI 设置，首先查找由`<section_String>`指定的段，然后在段里查找由`<key_String>`指定的 Key，并返回该 Key 的值。如果指定文件、段或 Key 没有找到，返回一个空字符串。例如：

```
GetINISetting "c:\\3ds max2\\3ds max.ini" "Directories" "Scenes"
```

2. `setINISetting <filename_String> <section_String> <key_String> <key_value_String>`

在指定文件里写入 INI 设置。如果写入成功，返回 True，否则返回 False。首先查找由`<section_String>`指定的段，然后在该段里查找由`<key_String>`指定的 Key，并将由`<key_value_String>`指定的 Key 值写入文件。如：

```
setINISetting "c:\\3ds max2\\3ds max.ini" "Directories" "Scenes" \
"c:\\3ds max\\scenes"
```

如果指定文件、段或 Key 没有找到，一个新文件、新段或 Key 将被创建；如果指定文件为只读文件，或正被 MAXScript 打开，Key 值将不能被写入。

## 21.8 存取.CUI 文件

下面方法可用来存取 CUI (Custom User Interface) 文件，这些方法都有一个前缀 cui。

1. `cui.getDir()`

返回一个字符串，包含当前活动 CUI 文件的路径。

2. `cui.getConfigFile()`

返回一个字符串，包含当前活动 CUI 文件名。

3. `cui.setConfigFile <filename_String>`

将指定文件设为当前活动 CUI 文件。

4. `cui.saveConfig()`

将当前活动 CUI 配置数据存储到当前活动 CUI 文件。

5. `cui.saveConfigAs <filename_String>`

将当前活动 CUI 配置数据存储到指定 CUI 文件，该.cui 文件将变成当前活动 CUI 文件。

6. `cui.loadConfig <filename_String>`

从指定文件中装载 CUI 配置数据，该.cui 文件将变成当前活动 CUI 文件。

# 第 22 章 事件侦测和信号反馈机制

3ds max 系统可以侦测场景对象或 3ds max 系统内部的某些事件（如对象移动、顶点编辑、对象删除、名称改变等）的发生，并用脚本定制一些代码来响应这些事件。

事件侦测程序被用于单个 MAXWrapper 对象，用来专门侦测对象某一指定属性的变化。当该属性被改变时，执行一个表达式。可以被侦测的属性有：geometry、name、transform 和 parameters。事件侦测程序没有存储在场景文件中。

信号反馈（Callback）可用来注册一些函数，当 3ds max 系统内部的某些指定事件发生时会调用这些函数。可以监测的事件包括用户界面下 Time 滑标的时间改变和视窗刷新。信号反馈程序也没有存储在 3ds max 场景里。

## 22.1 事件侦测和 when 构造函数

事件侦测程序用 when 构造函数来创建，每次当 when 构造函数被执行后，系统会创建一个数据类型为 ChangeHandler 的值，来代表刚刚建立的事件侦测程序。我们可以将这个 ChangeHandler 值存到一个变量或数组里。

一旦用 when 构造函数为一个或多个对象定义好 ChangeHandler 事件后，无论什么时候事件发生时，系统会自动调用该事件处理函数。

when 构造函数的定义格式有两种：

```
when <attribute> <objects> change[s] [ id:<name> ]      \
    [ handleAt:#redrawViews|#timeChange ] \
    [ <object_parameter> ] do <expr>
when <objects> deleted [ id:<name> ]           \
    [ handleAt:#redrawView|#timeChange ]  \
    [ <object_parameter> ] do <expr>
```

在第一种格式中，<attribute> 可以为下面几种：topology、geometry、name[s]、transform、select、parameters、subAnimStructure、controller、children。它们用来侦测指定对象特定属性的变化。

第二种格式专门用来侦测对象删除操作（可以直接在场景中删除，也可以在别的脚本中进行删除）。当删除操作发生后，ChangeHandler 函数被调用。所以用户不可以在 ChangeHandler 函数中对这些对象执行任何 3ds max 操作。这种格式主要用于下面这种情况：如果用户有一个包含 MAXScript 对象的数组或结构，当场景中有些对象被删除后，用户要把它从该数组或结构中清除。

在以上两种格式中，参数`<objects>`都可以是下面几种：

- ◆ 单个 3ds max 对象，比如场景对象、Controller、Modifier 或 Material；
- ◆ 一个对象集，如'selection'或'cameras'；
- ◆ 路径通配符如\$foo\*；
- ◆ 一个元素为 3ds max 对象的数组。

如果参数`<objects>`指定了不止一个对象，对象集合中任一对象的指定属性发生改变时，都会导致系统自动调用该事件处理函数。下面详细说明各可选参数：

- ◆ `Id` 为可选参数，为一个或多个 handler 指定一个 ID 码，其数据类型为 Name，ID 码可用来在以后的脚本中删除 Handler，如果有几个 Handler 共用一个 ID 码，它们会作为一组被一起删除；
- ◆ `handleAT` 为可选参数，告诉系统并不在改变发生后马上执行 Handler 表达式，如果指定 `handleAT:#redrawViews`，延迟至视窗刷新后执行；如果指定 `handleAT :#timeChange`，延迟至当前 3ds max 动画事件改变后执行。这些延迟执行的做法非常有用，比如：

```
when select $ changes id:#foo handleAt:#redrawViews do ...
```

- ◆ `<object_parameter>` 为可选参数，让用户指定一个变量名，用它来代表刚刚被改变的对象，以便在 `do <expr>` 语句中使用这些对象。这个参数主要用于`<objects>`为多个对象时，用它来判别哪些对象被修改；
- ◆ `<expr>` 为可选参数，可以为单个表达式，也可以为块表达式，表示当侦测到指定事件发生时，用户希望执行的动作。

例如：

```
--当对象 box01 的转换类属性发生改变时
when transform $box01 changes do
    $box02.pos = $box01.pos + delta
--当选择集里某些对象的.name 属性被改变时
when names selection change obj do update_name_table obj
when #($foo, $baz, $bar)deleted obj do
(
    messagebox "Warning!"
    deleteItem obj_table(findItem obj_table obj)
)
```

`<attribute>` 可被改变的属性说明如下：

- ◆ `topology` 表示当对象的拓扑结构被改变时(如执行 Mesh 对象的 smooth、optimize、顶点删除等操作)，`ChangeHandler` 被调用；
- ◆ `geometry` 表示当对象的 Geometry 在 Modify 面板里被改变时（如移动顶点），`ChangeHandler` 被调用；
- ◆ `name` 或 `names` 表示当对象的.name 属性被改变时，`ChangeHandler` 被调用。用户

每改变一个字符，Handler 就被调用一次；

- ◆ transform 表示当对象的.transform 属性被改变时（比如被执行 Move、Rotate、Scale 操作），ChangeHandler 被调用；
- ◆ select 表示当一个场景对象被加入或移出当前选择集时，ChangeHandler 被调用。可以用<node>.isSelected 来判别对象的最新选择状态；
- ◆ parameters 表示当对象的任何参数被改变时，ChangeHandler 被调用；
- ◆ subAnimStructure 表示当动态 subAnim 结构被改变时（比如 Editable Mesh 对象的一个新顶点的变动被设置动画，或在 controller list 里加入一个新的 controller），ChangeHandler 被调用。
- ◆ controller 表示当一个新的 Controller 被赋给对象的某一属性时，ChangeHandler 被调用；
- ◆ children 表示当指定对象有一个子对象被添加或移走时，ChangeHandler 被调用。

可以用下面两种方法来删除 ChangeHandler：

1. `deleteChangeHandler <change_handler>`

删除指定 ChangeHandler，<change\_handler>为由 when 构造函数返回的值。

2. `deleteAllChangeHandlers [ id:<name> ]`

如果可选参数 ID:没有指定，删除所有 ChangeHandler；如果指定了 ID:，删除所有 ID 为指定值的 ChangeHandler。例如：

```
deleteAllChangeHandlers id:#foo
```

从运行效率上考虑，用户当然不希望运行那些不相关的 ChangeHandler，所以用户应该删除那些不再需要的 ChangeHandler。

关于 ChangeHandler，下面几点需要特别注意：

- ◆ 如果运行 ChangeHandler 事件侦测程序时发生运行错误，系统会显示一个错误信息，此后该 ChangeHandler 事件侦测程序就一直被禁用；
- ◆ 如果由 ChangeHandler 侦测的所有对象都被删除，那么 ChangeHandler 事件侦测程序也被删除了；
- ◆ ChangeHandler 事件侦测程序里的 do <expr>代码运行在一个特殊的关联语句里，而不是在创建它的代码里。在<expr>代码里不可以引用 when 作用域以外的局部变量，因为这些变量在调用 ChangeHandler 时并不在执行堆栈里。编译器会自动检测在 when 里是否包含对 when 作用域以外的局部变量的引用，如果有，会给出一个编译信息。但也有一种例外的情况：在 Utility 和 Rollout 里的局部函数、Rollout 变量和嵌套 Rollout 里，当处在 Rollout 代码作用域里时，用户可以在 when 代码里对它们进行引用，因为它们是直接与它们的 Rollout 或 Utility 对象相连的。

如果改变发生在某一被设置动画的对象属性里时，ChangeHandler 事件侦测程序并不会被调用。例如：当用户移动某一对象时，会调用它的 transform 类型的 ChangeHandler 侦测事件，而如果该对象的.position 属性被设置动画，当播放动画时，对象的.transform 属性也被改变，但此时并不会调用它的 ChangeHandler 事件处理程序。

如果将一个 ChangeHandler 事件侦测程序赋给多个对象的某一属性，每一个指定属性被改变的对象都会调用一次 ChangeHandler。例如：

```
when select $ changes obj do update_modifier_list obj
```

当每次有对象被选择或解除选择时，都会调用函数 update\_modifier\_list。如果用户执行下拉菜单命令 Edit | Select All，update\_modifier\_list 函数会被当前每一个被选择的对象调用一次（当它们被解除选择时），然后又被场景里所有对象调用一次（当它们被全部选择时）。如果该函数里有大量的计算，此时会导致系统运行速度明显变慢。

ChangeHandler 事件处理程序仅针对 when 构造函数运行之前已存在的对象，并不自动适用于此后新创建的对象。

如果定义了多个 ChangeHandler 事件侦测程序，其被调用的顺序是随意的。

由于 3ds max 内部处理信号的方式，用“\$”来代表当前被选择对象的方法最好不要用在 ChangeHandler 里来代表选择集，而应用 selection 来代表选择集。

**注意** ChangeHandler 系统的运行实际上基于 3ds max 内部的信号系统，而信号系统实际上管理着所有动画和用户与 3ds max 之间的交互。在实际中可能存在这种情形：系统为同一个状态的改变发出许多的信号，这样如果为许多对象建立 ChangeHandler 事件侦测程序，会导致运算量的大量增加，严重影响系统的运行速度。所以编程者要谨慎使用 ChangeHandler 事件侦测程序，不能把它作为控制脚本流程的主要方法，如果一定要这么做，应该使用参数 handleAt: 来延迟处理。

## 22.2 时间改变信号反馈机制

用户可以注册一个或几个函数，在当前 3ds max 动画时间改变时（如用户拖动 Time Slider 或播放动画）会被调用。下面的方法用来注册建立或取消这种信号反馈（Time Callback）：

1. registerTimeCallback <fn>  
建立 Callback。
2. unRegisterTimeCallback <fn>  
取消 Callback。

用户可以注册任意数量的 Callback 程序，当时间发生改变时，每一个函数都会被单独调用。这些函数不能有自己的参数，可以用 MAXScript 系统变量 currentTime 来获取当前的动画时间，例如：

```
fn time_p = print currentTime
registerTimeCallback time_p
```

在上例中，当用户拖动 Time Slider 或播放动画时，注册的函数会在 Listener 窗口打印出当前的时间。

特别注意：

- ◆ 如果运行 Time Callback 时发生运行错误，系统会显示一个错误信息，此后该 Time

Callback 就一直被禁用。

- ◆ 系统在渲染时不会调用 Time Callback，即使在渲染多帧动画时。
- ◆ 注册函数运行在一个特殊的关联语句里，而不是在其创建代码里。在注册函数代码里不可以引用函数作用域以外的局部变量，因为这些变量当调用 Time Callback 时并不在执行堆栈里。但也有一种例外的情况：在 Utility 和 Rollout 里的局部函数、Rollout 变量和嵌套 Rollout，当处在 Rollout 代码作用域里时，用户可以在注册函数代码里引用它们，因为它们直接与其 Rollout 或 Utility 对象相连。
- ◆ 函数 `registerTimeCallback()` 注册的是函数值，而不是函数名或全局变量，这意味着在注册后重新定义函数并不会改变先前定义好的 Callback。用户只有先取消注册，再重新定义函数，然后重新注册函数，或者将注册函数设为对另一函数的调用，如：

```
fn time_cb = print currentTime
fn tcb = time_cb()
registerTimeCallback tcb
```

在上例中，注册函数 `tcb()` 真正调用的 Callback 程序 `time_tcb()`，这样我们可以通过重新定义 `time_tcb()` 来达到修改注册函数的目的。这个方法可在调试程序时使用。

- ◆ Callback 程序在 3ds max 装载一个新文件或重置 3ds max 后仍然有效。

## 22.3 视窗刷新信号反馈机制

用户可以注册一个或几个函数，当 3ds max 视窗刷新时它们会被调用。下面的方法用来注册建立或取消这种 Callback：

1. `registerRedrawViewsCallback <fn>`  
建立注册。
2. `unRegisterRedrawViewsCallback <fn>`  
取消注册。

用户可以注册任意数量的 Callback 程序，当刷新视窗时，每一个函数都会被单独调用。这些函数不能有自己的参数。

```
fn redrawviews_p = print "Viewports Redrawn"
registerRedrawViewsCallback redrawviews_p
```

在上例中，当视窗被刷新时，注册的函数会在 Listener 窗口打印字符串“Viewports Redrawn”。

## 22.4 通用事件反馈机制

MAXScript 允许用户为 3ds max 发出的所有信号事件注册自己的反馈脚本，比如场景文件的 Open、New、Reset、Save（事件之前或之后）、Render、选择集改变（事件之前或

之后)。用户可为每一个信号事件指定任意数量的 Callback 脚本。Callback 脚本可放置在一些 ID 集里,这样就可以用 ID 为组进行 Callback 脚本删除。Callback 脚本还可以被指定为 persistent,这样它们就可以跟场景文件一起存储、重载。

Callback 脚本可用下面的函数创建、维护:

```
1. callbacks.addScript <callback_type_name> \
    (<script_String> | <script_Stringstream> | fileName:<filename_String>) \
    [ id:<name> ] [ persistent:<Boolean> ]
```

本方法用来注册一个新的 Callback 脚本。

参数说明如下:

- ◆ <callback\_type\_name> 指定脚本所属的信号事件类型,有效的类型名如下:
  - <script\_String>
  - <script\_Stringstream>
  - fileName:<filename\_String>

脚本的三种形式,前面两种形式包含了要运行的脚本源代码,第三种形式指定了一个要装载的脚本文件,当信号事件发生时,系统会装载由参数 fileName:参数指定的脚本文件并运行它。用户可指定直接的脚本字符串或脚本文件,但不能同时指定。

- ◆ Id:为可选参数,让用户把一组 Callback 用一个唯一的 ID 进行标识,这样就可以把 ID 码相同的 Callback 一起删除,而不影响别的 Callback;
- ◆ persistent:为可选参数,让用户来控制 Callback 脚本是固定地存储在当前打开的场景文件里,还是作为一个全局的 Callback 脚本,无论注册后执行多少次文件打开或关闭操作,它都会保持有效。如果设为 True,标识该 Callback 脚本存储在当前场景文件里,只要该文件被重新打开,该 Callback 就被装载和注册,而当执行 File | New 或 File | Reset 操作后,该 Callback 在新的场景文件里会被清除;默认值为 False,表示 Callback 脚本为一个全局 Callback 脚本,在关闭 3ds max 之前,对所有打开的场景文件都有效,但不会存储在场景文件里,在下次打开同一文件时,不会自动装载。例如:

```
callbacks.addScript #preRender "setUpRenderGeom()" id:#jbwRender
```

上例注册了一个新的 Callback 脚本,仅在渲染之前被调用。该脚本调用一个函数(它应该已经被定义),注册时为它指定了一个 ID。

2. callbacks.removeScripts [ <callback\_type\_name> ] [ id:<name> ]

本方法用来解除一个或多个 Callback 脚本的注册并删除它们。如果仅指定参数 <callback\_type\_name>,会删除指定类型的所有 Callback 脚本;如果仅指定 id:<name>,会删除所有参数 ID 为<name>的 Callback 脚本;如果同时指定,会删除指定类型里 ID 参数为<name>的 Callback 脚本。

3. callbacks.show()

本方法在 Listener 窗口里列出当前已注册的所有 Callback 脚本。

4. callbacks.broadcastCallback <callback\_type\_name>

本方法发出一个事件信号，让所有的 Callback 脚本都执行一次。在 3ds max 里，`#preRenderFrame` 只能由渲染器调用，而不能用本方法调用。

有效参数 `callback_type_name` 的有效信号值见下表。

Callback_type_name	说明
<b>与 Manipulate 模式按钮有关的消息</b>	
<code>#animateOn</code>	当 3ds max 界面上 Animate 按钮被打开时发送消息
<code>#animationRangeChange</code>	当动画时间范围被改变时发送消息
<code>#manipulateModeOn</code>	当 3ds max 界面上 Manipulate 按钮被按下时发送消息
<code>#manipulateModeOff</code>	当退出 Manipulate 模式时发送消息
<b>与 Modifier 面板有关的消息</b>	
<code>#modPanelObjPreChange</code>	当前编辑对象将在 Modifier 面板下被修改前发送消息
<code>#modPanelSelChanged</code>	当选择一个新选择集后打开 Modify 面板时（可以是 Modify 面板刚刚被选择或在场景里一个新选择集被选择），发送消息。系统在 Modify 面板被刷新之前、选择集建立之后向系统发送信息
<code>#modPanelObjPostChange</code>	当前编辑对象将在 Modifier 面板下被修改后发送消息
<b>与 Modifier 有关的消息</b>	
<code>#preModifierAdded</code>	在将一个 modifier 添加到 Node 对象之前发送消息。如果在 Callback 程序里调用函数 <code>callbacks.notificationParam()</code> ，会返回一个二元数组，分别包含 Node 对象和要添加的 modifier
<code>#postModifierAdded</code>	在将一个 modifier 添加到 Node 对象之后发送消息。如果在 Callback 程序里调用函数 <code>callbacks.notificationParam()</code> ，会返回一个二元数组，分别包含 Node 对象和已添加的 modifier
<code>#preModifierDeleted</code>	在将一个 modifier 从 Node 对象删除之前发送消息。如果在 Callback 程序里调用函数 <code>callbacks.notificationParam()</code> ，会返回一个二元数组，分别包含 Node 对象和要删除的 modifier
<code>#postModifierDeleted</code>	在将一个 modifier 从 Node 对象删除之后发送消息。如果在 Callback 程序里调用函数 <code>callbacks.notificationParam()</code> ，会返回一个二元数组，分别包含 Node 对象和被删除的 modifier
<b>与 材质库 有关的消息</b>	
<code>#mtlLibPreOpen</code>	在将一个材质库装载之前发送消息
<code>#mtlLibPostOpen</code>	在将一个材质库装载之后发送消息。如果在 Callback 程序里调用函数 <code>callbacks.notificationParam()</code> ，会返回已加载的材质库
<code>#mtlLibPreSave</code>	在将一个材质库保存之前发送消息
<code>#mtlLibPostSave</code>	在将一个材质库保存之后发送消息
<code>#mtlLibPreMerge</code>	在将一个材质库合并之前发送消息
<code>#mtlLibPostMerge</code>	在将一个材质库合并之后发送消息
<b>与 Material 有关的消息</b>	
<code>#mtlRefAdded</code>	当一个材质参考被添加到场景里时发送消息。如果在 Callback 程序里调用函数 <code>callbacks.notificationParam()</code> ，会返回该材质

(续表)

Callback_type_name	说明
#mtlRefDeleted	当一个材质参考被从场景里删除时发送消息。如果在 Callback 程序里调用函数 callbacks.notificationParam(), 会返回该材质
<b>与系统有关的消息</b>	
#systemPreReset	在 3ds max 被重置之前发送消息
#systemPostReset	在 3ds max 被重置之后发送消息
#postSystemStartup	在 3ds max 被启动时发送消息
#pluginLoaded	无论何时装载一个插件时发送消息
#systemPreNew	在 3ds max 刚被重置之前发送消息。如果在 Callback 程序里调用函数 callbacks.notificationParam(), 会返回一个整数, 反映了用户在 New Scene 对话框里的选项: 1: 全部刷新; 2: 保留对象; 3: 保留对象和层级
#systemPostNew	在 3ds max 刚被重置之后发送消息
#preSystemShutdown	当 3ds max 被关闭时发送消息
#postSystemShutdown	当 3ds max 完成关闭之前发送消息
#systemPreDirChange	当 3ds max 系统文件路径改变(在 Configure Path 对话框里进行修改)之前发送消息
#systemPostDirChange	当 3ds max 系统文件路径改变(在 Configure Path 对话框里进行修改)之后发送消息
#pluginLoaded	当一个插件被装载进场景时发送消息。如果在 Callback 程序里调用函数 callbacks.notificationParam(), 会返回一个字符串, 包含刚刚加载的.dll 文件名称
<b>与 Radiosity 插件有关的消息</b>	
#radiosityPluginChanged	Radiosity 插件被修改时发送消息
#radiosityProcessDone	Radiosity 插件处理完成时发送消息
#radiosityProcessReset	Radiosity 插件处理被重置时发送消息
#radiosityProcessStart	Radiosity 处理开始后发送消息
#radiosityProcessStopped	Radiosity 处理停止后发送消息
<b>与 Undo 有关的消息</b>	
#sceneUndo	执行一次 Undo 操作后发送消息
#sceneRedo	执行一次 Redo 操作后发送消息
<b>与 Schematic View 有关的消息</b>	
#svSelectionSetChanged	当 Schematic View 对象选择集被改变时发送消息
#svDoubleClickGraphNode	当用户鼠标双击 Schematic View 对象时发送消息
#svPreLayoutChange	当 Schematic View 执行布局算法之前发送消息。如果在 Callback 程序里调用函数 callbacks.notificationParam(), 会返回一个整数, 表示 Schematic View 序号

(续表)

Callback_type_name	说明
#svPostLayoutChange	当 Schematic View 执行布局算法之后发送消息。如果在 Callback 程序里调用函数 callbacks.notificationParam(), 会返回一个整数, 表示 Schematic View 序号
与 Asset Browser 有关的消息	
#assetBrowserPreNavigate	当一个 URL 被装载进 Asset Browser 时发送消息。如果在 Callback 程序里调用函数 callbacks.notificationParam(), 会返回一个字符串, 包含要浏览的 URL
与渲染有关的消息	
#preRender	在渲染开始之前发送消息。在本事件的 Callback 脚本中, 用户还可以向场景中添加对象。如果为多帧渲染, 仅在渲染开始前发送一次消息, 而不是在每帧开始之前都发送消息。在#preRender 消息的 Callback 脚本中, 改变任何渲染参数(如 Height、Width、Aliasing 等)都不会对本次渲染发生作用, 而要等到下次渲染时才生效。返回一个包含 28 个元素的数组
#postRender	在渲染结束时发送消息。在发送消息时, 渲染对话框已关闭, 所以在本事件的 Callback 脚本中, 用户也可以向场景中添加对象。如果为多帧渲染, 仅在渲染全部结束后发送一次消息, 而不是在每帧结束后都发送消息
#preRenderEval	在渲染器开始渲染对象之前发送消息
#preRenderFrame	在每帧开始渲染之前发送消息。本消息在渲染器对场景几何体执行快照操作后发送, 在本 Callback 脚本中, 不能对场景进行修改
#postRenderFrame	在每帧渲染完成之后发送消息, 返回一个包含 22 个元素的数组
#beginRenderingReflectRefractMap	渲染时, 在准备贴图反射和折射之前发送消息
#beginRenderingActualFrame	在设置好渲染参数之后、实际开始渲染之前发送消息
#beginRenderingTonemappingImage	在渲染 Tonemapping 图像之前发送消息
与 Render 对话框和渲染插件有关的消息	
#tabbedDialogCreated	在一个选项卡式 render 对话框被创建之后发送消息
#tabbedDialogDeleted	在一个选项卡式 render 对话框被删除之后发送消息
#preRendererChange	在当前 renderer 被修改, 或某一个 renderer 的类型被修改之前发送消息
#postRendererChange	在当前 renderer 被修改, 或某一个 renderer 的类型被修改之后发送消息
#renderParamsChanged	当通用渲染器参数被修改时发送消息
与文件有关的消息	
#filePreOpen	打开一个新文件之前发送消息
#filePostOpen	成功打开一个新文件之后发送消息
#filePreMerge	执行文件合并之前发送消息
#filePostMerge	文件合并成功之后发送消息
#filePreSave	执行文件保存之前发送消息

(续表)

Callback_type_name	说明
#filePostSave	文件保存之后发送消息
#filePostOpenProcess:	在打开文件操作完成之后发送消息
#filePreSaveOld	旧版文件保存之前发送消息
#preImport	文件导入之前发送消息
#postImport	文件导入之后发送消息
#filePostMergeProcess	在导入文件操作完成之后发送消息
#importFailed	如果文件导入失败，发送消息
#preExport	导出文件之前发送消息
#postExport	文件导出成功之后发送消息
#exportFailed	如果文件导出失败，发送消息
与 Xrefs 有关的消息	
#objectXrefPreMerge	装载 Xref 对象之前发送消息
#objectXrefPostMerge	装载 Xref 对象之后发送消息
#sceneXrefPreMerge	合并 Xref 对象之前发送消息
#sceneXrefPostMerge	合并 Xref 对象之后发送消息
与 Layer 有关的消息	
#layerCreated	在 layer 被创建之后发送消息。在 callbacks.notificationParam() 函数中调用 callbacks.notificationParam() 函数返回一个 layerReferenceTarget 值。注意：#layerCreated 有一个问题，如果一个对象被合并，或执行一个 Xref 对象，一个 Layer 可能会被创建，但这些操作不会导致一个#layerCreated 事件
#layerDeleted:	在 layer 被删除之前发送消息
#nodeLayerChanged	在对象被放在一个新层之后发送消息。在 callbacks.notificationParam() 函数中调用 callbacks.notificationParam() 函数返回一个三元数组：Node 对象、旧 Layer 和新 Layer
与 Node 对象有关的消息	
#nodeCreated	当 Node 对象被创建时发送消息
#nodeRenamed	当 Node 对象被更名时发送消息
#nodeHide	当 Node 对象被隐藏时发送消息
#nodeUnhide	当 Node 对象被解除隐藏时发送消息
#nodeFreeze	当 Node 对象被冻结时发送消息
#nodeUnfreeze	当 Node 对象被解除冻结时发送消息
#nodeLinked	当建立一个新的父-子连接时发送消息
#nodeUnlinked	当断开一个父-子连接时发送消息
#nodePreMaterial	当 Node 对象被赋予新材质之前发送消息
#nodePostMaterial	当 Node 对象被赋予新材质之后发送消息
#sceneNodeAdded	当 Node 对象被添加到场景后发送消息

(续表)

Callback_type_name	说明
#selNodesPreDelete	当选择的 Node 对象被删除之前发送消息
#selNodesPostDelete:	当选择的场景对象被删除之后发送消息
#nodeNameSet	在一个 Node 对象的.name 属性被设置或被修改时发送消息
#preNodeBonePropChanged	在 Node 对象的 Bone 属性被修改之前发送消息
#postNodeBonePropChanged	在 Node 对象的 Bone 属性被修改之后发送消息
#preNodeGeneralPropChanged	在 Node 对象的通用属性被修改之前发送消息
#postNodeGeneralPropChanged	在 Node 对象的通用属性被修改之后发送消息
#preNodeGiPropChanged	在 Node 对象的.advancedLighting 属性被修改之前发送消息
#postNodeGiPropChanged	在 Node 对象的.advancedLighting 属性被修改之后发送消息
#preNodeMentalrayPropChanged	在 Node 对象的.mentalray 属性被修改之前发送消息
#postNodeMentalrayPropChanged	在 Node 对象的.mentalray 属性被修改之后发送消息
#preNodesCloned	在 Node 对象被克隆之前发送消息
#postNodesCloned	在 Node 对象被克隆之后发送消息。在 Callback 程序里调用 callbacks.notificationParam(), 返回一个三元组: 第一个元素为源 Node 对象, 第二个元素为克隆 Node 对象, 第三个元素为一个 Name 类值, 其值可能为#copy、#instance 或#reference, 表示克隆类型
#preNodeUserPropChanged	在 Node 对象的自定义属性被修改之前发送消息
#postNodeUserPropChanged	在 Node 对象的自定义属性被修改之后发送消息
#preMirrorNodes	在 Mirror 操作开始之前发送消息
#postMirrorNodes	在 Mirror 操作完成之后发送消息
#selectedNodesPreDelete	在被选择对象被删除之前发送消息
#selectedNodesPostDelete	在被选择对象被删除之后发送消息
#selectionSetChanged	在对象选择集被修改之后发送消息
与 FileLink 有关的消息	
#fileLinkPostAttach	在附属文件连接之后发送消息
#fileLinkPreBind	在文件被连接绑定之前发送消息
#fileLinkPostBind:	在文件连接绑定之后发送消息
#fileLinkPreDetach	在解除文件连接之前发送消息
#fileLinkPostDetach	在解除文件连接之后发送消息
#fileLinkPreReload	在文件连接重载之前发送消息
#fileLinkPostReload	在文件连接重载之后发送消息
#fileLinkPreAttach	在附属文件连接之前发送消息
其他消息	
#colorChanged	当系统刷新它的颜色时发送消息
#heightMenuChanged	当用户对 Height 菜单进行操作时发送消息

(续表)

Callback_type_name	说明
#selectionSetChanged	在选择集被改变后发送消息
#bitmapChanged	在 BitMap 被重载后发送消息
#byCategoryDisplayFilterChanged	在对象被改变隐藏状态后发送消息
#colorChanged	在系统刷新定制颜色后发送消息
#customDisplayFilterChanged	在用户改变显示过滤器 (activated/deactivated) 后发送消息
#lightingUnitDisplaySystemChange	在用户改变光线显示单位后发送消息
#spacemodeChange	在用户改变参考坐标系时发送消息
#timeunitsChange	如果用户改变时间格式设置, 发送消息
#unitsChange	如果用户改变单位设置, 发送消息
#viewportChange	如果用户改变视窗布局, 发送消息
#mainWindowEnabled	如果主窗口获得焦点, 发送消息。在 Callback 程序里调用 notificationParam() 函数返回一个 Boolean 值: 如果返回 True, 表示 3ds max 主窗口获得焦点; 如果为 False, 其他窗口获得焦点
#preProgress	在进度条显示前发送消息
#postProgress	在进度条完成后发送消息

# 第 23 章 MAXScript 杂项函数

## 23.1 暂停脚本执行

`sleep <time_in_seconds>`

让系统暂停执行脚本。参数`<time_in_seconds>`为浮点数，以秒为单位，表示暂停执行的时间。

**注意** 暂停时间长度并不非常精确。用户可以按 ESC 键来中止暂停（同时也会中止所有脚本执行），也可以用 SPACE 键来中止 `sleep()` 函数而脚本会继续执行。在两种情况下，都需要持续按下键盘半秒时间，MAXScript 才会有反应。

## 23.2 时间计算函数

`timeStamp()`

返回当天从零点（00:00）到当前流逝的时间，以毫秒为单位。主要用于一些与时间有关的操作，如：

```
start = timeStamp()
process_mesh()           --进行一些复杂的处理
end = timeStamp()
--打印处理需要的时间
format "Processing took % seconds\n" ((end - start)/ 1000.0)
```

## 23.3 控制渲染器

用户可以用 `render()` 函数来调用 3ds max 渲染器，并可以用许多可选参数来控制渲染。其调用格式为：

`render [ ] [ ] .....`

其中[ ]为可选参数，详细说明如下：

1. [ `camera: <camera_node>` ]

默认值为当前活动视窗。

2. [ `frame: <number> | #current` ]

渲染单帧，默认值为#current。

3. [ framerange: <interval> | #active ]
  - [ fromframe: <number> ]
  - [ toframe: <number> ]
  - [ nthframe: <number> ]
 控制渲染多帧时渲染哪些帧，默认值为 unsupplied。
4. [ outputwidth: <number> ]
 输出位图宽度，默认值为当前渲染对话框的宽度。
5. [ outputheight: <number> ]
 输出位图高度，默认值为当前渲染对话框的高度。
6. [ outputSize: <Point2> ]
 指定渲染输出位图大小的一种替代方法，<Point2>的格式为[width,height]。
7. [ pixelaspect: <number> ]
 像素高宽比，默认值为 1.0。
8. [ renderhiddenobjects: <Boolean> ]
 控制是否渲染隐藏对象，默认值为当前渲染对话框中复选框 Render Hidden Objects 的设置。
9. [ superblack: <Boolean> ]
 默认值为当前 Render Scene 对话框中复选框 Superblack 的设置。
10. [ force2sided: <Boolean> ]
 默认值为当前 Render Scene 对话框中复选框 Force 2 Sided 的设置。
11. [ renderatmosphericeffects: <Boolean> ]
 默认值为当前 Render Scene 对话框中复选框 Render Atmospheric Effects 的设置。
12. [ renderfields: <Boolean> ]
 默认值为当前 Render Scene 对话框中复选框 Render Fields 的设置。
13. [ fieldorder: #odd | #even ]
 默认值为 Preference setting 对话框里 Rendering 选项卡下 Field Order 栏的设置。
14. [ outputfile: <String> ]
 指定输出文件名，默认值为输出到虚拟帧缓存（VFB）。如果渲染多帧，而指定文件类型为单幅图像（如.bmp、.jpg、.tga 等），帧序号会自动添加到文件名后面。
15. [ vfb: <Boolean> ]
 默认值为当前 Render Scene 对话框中复选框 Virtual Frame Buffer 的设置。
16. [ netrender: <Boolean> ]
 默认值为当前 Render Scene 对话框中复选框 Net Render 的设置。
17. [ renderer: #draft | #production]
 设置渲染器类型：#draft 或#production，默认值为当前渲染器对话框里的设置。
18. [ renderType: #normal | #region | #regionCrop | #blowup | #selection ]
19. [ region: #(left,top,right,bottom)]
 控制渲染类型，对应 3ds max 工具栏里的 Render Type 工具，如果设置为#region 或

#blowUp, 参数 region: 被用来指定活动视窗里的一个渲染区域, 单位为像素, 坐标原点为 VFB 位图的左上角。注意只有在渲染视窗时才可以指定渲染类型为#region 和#blowUp, 如果在进行 Camera 渲染时指定渲染类型为#region 和#blowUp, 会导致运行错误。默认值为#normal。

20. [ to: <bitmap> ]

指定一个已有的位图, 存入渲染结果。Render()函数从该位图里获取图像大小及其他参数设置。如果没有指定 to:参数, 系统会创建一个新位图, 并将它作为 render()函数的返回值。

21. [ channels: <array\_of\_channel\_names> ]

指定在渲染过程中创建哪些 g-buffer 通道。如:

```
bm = render camera:$cam2 channels:#(#zDepth, #coverage, #objectID)
```

将对摄像机 cam2 进行渲染, 将渲染结果放在一个新位图里, 该位图包含了 z-depth, pixel coverage 和对象 g-buffer\_ID 通道。参数 channels: 必须是一个数组, 元素为通道标识符。通道标识符共有下面这些:

#zDepth	#coverage
#matID	#node
#objectID	#shaderColor
#UVCoords	#shaderTransparency
#normal	#velocity
#unClamped	#weight

默认值为没有 g-buffer 通道。

22. [ aperture: <Float> ]

默认值为当前 Render Scene 对话框的微调器 Aperture Width 的值。

23. [ ditherTrueColor: <Boolean> ]

默认值为 Preferencesetting 对话框里 Rendering 选项卡下 Dither True Color 复选框的设置。

24. [ ditherPaletted: <Boolean> ]

默认值为 Preferencesetting 对话框里 Rendering 选项卡下 Dither Paletted 复选框的设置。

25. [ videocolorcheck: <Boolean> ]

默认值为当前 Render Scene 对话框的复选框 Video Color Check 的值。

26. [ renderPAL: <Boolean> ]

默认值为 Preferencesetting 对话框里 Rendering 选项卡下 Video Color Check NTSC/PAL 复选框的设置。如果为 True, 表示视频颜色检测可用, 执行 PAL 制式颜色检测。

27. [ superBlackThreshold: <Integer> ]

默认值为 Preferencesetting 对话框里 Rendering 选项卡下 Super Black Threshold 栏的设置。

28. [ maxPixelSize: <Float> ]

设置渲染图像的像素大小。

当使用 3ds max 标准 Scanline Renderer 时，还有下面这些参数可用：

29. [ antiAliasing: <Boolean> ]

默认值为当前 Renderer 的 MAX Default Scanline A-Buffer 卷展栏下 Anti-Aliasing 复选框的设置。

30. [ antiAliasFilterSize: <Float> ]

默认值为当前 renderer 的 MAX Default Scanline A-Buffer 卷展栏下 Anti-Aliasing Filter Size 复选框的设置。

31. [ antiAliasFilter: <filter> ]

默认值为当前 renderer 的 MAX Default Scanline A-Buffer 卷展栏下 Anti-Aliasing Filter 复选框的设置。

32. [ enablePixelSampler: <Boolean> ]

默认值为当前 renderer 的 MAX Default Scanline A-Buffer 卷展栏下 Global SuperSampling 复选框的设置。

33. [ mapping: <Boolean> ]

默认值为当前 renderer 的 MAX Default Scanline A-Buffer 卷展栏下 Mapping 复选框的设置。

34. [ shadows: <Boolean> ]

默认值为当前 renderer 的 MAX Default Scanline A-Buffer 卷展栏下 Shadows 复选框的设置。

35. [ autoReflect: <Boolean> ]

默认值为当前 renderer 的 MAX Default Scanline A-Buffer 卷展栏下 Auto-Reflect/Refract and Mirrors 复选框的设置。

36. [ forceWireframe: <Boolean> ]

默认值为当前 renderer 的 MAX Default Scanline A-Buffer 卷展栏下 Force Wireframe 复选框的设置。

37. [ wireThickness: <Float> ]

默认值为当前 renderer 的 MAX Default Scanline A-Buffer 卷展栏下 Wire Thickness 栏的设置，默认值为 1.0。

38. [ filterMaps: <Boolean> ]

默认值为当前 renderer 的 MAX Default Scanline A-Buffer 卷展栏下 Anti-Aliasing Filter Maps 复选框的设置。

39. [ objectMotionBlur: <Boolean> ]

默认值为当前 renderer 的 MAX Default Scanline A-Buffer 卷展栏下 Object Motion Blur Apply 复选框的设置。

40. [ objectBlurDuration: <Float> ]

默认值为 0.5。

41. [ objectBlurSamples: <Integer> ]

默认值为 10。

42. [ objectBlurSubdivisions: <Integer> ]

默认值为 10。

43. [ imageMotionBlur: <Boolean> ]

默认值为当前 renderer 的 Image Motion Blur Apply 复选框的设置。

44. [ imageBlurDuration: <Float> ]

默认值为 0.5。

45. [ autoReflectLevels: <Integer> ]

默认值为 1。

用户可以调用 render() 函数，并可以对渲染结果做下面四件事情的一种或几种：

- ◆ 用参数 vfb: 控制是否将渲染结果显示在虚拟帧缓存里；
- ◆ 用参数 outputfile: 指定将渲染结果存储到一个图形文件里，图形文件类型由文件名的后缀决定；
- ◆ 返回一个类型为 Bitmap 的值，用户可以对它执行任何 Bitmap 操作；
- ◆ 用参数 to: 指定将渲染结果存储到一个已有的图形文件里，渲染结果的设置如：height、width、aspect、gamma、文件名都从该图形文件获取。

例如：

```
render camera:$cam01 outputwidth:320 outputheight:240
for c in cameras do render c outputFile:(c.name + ".bmp")vfb:off
Rollout1.image.bitmap = render camera:$cam01 ...
```

## 23.4 执行外部命令或程序

MAXScript 提供了两种方法来执行外部程序：

1. DOSCommand <command\_String>

参数<command\_String>为一个字符串，包含一个 DOS 命令，系统随后会执行该命令。

例如：

```
DOSCommand "delete c:\\temp\\foo.dat"
DOSCommand("copy" + source_file + " " + dest_folder)
```

也可以用这个函数来调用其他函数，如可以收集场景文件里所有贴图和图像文件，并把它们放在一个文件夹里。

DOSCommand() 函数返回一个整数，表示命令执行后的结果。

2. ShellLaunch <filename\_String> <parameters\_String>

本方法模拟用户在 Windows 桌面上双击某一图标的操作，比如：

```
ShellLaunch "E:\\tests\\lookup.html"
```

会启动浏览器 Netscape/Ie4，同时打开超文本文件 lookup.html。

如果参数<filename\_String>没有指定文件扩展名，Windows 会在指定路径下搜索同名的

可执行文件，如果没有指定路径，Windows 会在环境变量 PATH 指定的搜索路径里依次查找可执行文件。如：

```
shellLaunch "e:/program files/ucalc/ucalc" " "
```

会执行指定路径下的文件 ucalc.exe。

参数<parameters\_String>为应用程序指定命令行参数，如：

```
shellLaunch "e:/t.avi" "/play /loop"
```

会运行与.avi 文件类型相关的应用程序，并将 play 和 loop 作为命令行参数。

## 23.5 退出和重置 3ds max 系统

quitMAX()函数用来在脚本里退出 3ds max，本函数经常用于批处理和渲染脚本，其格式为：

1. quitMAX [ #noPrompt ]

如果没有指定可选参数#noPrompt，而在场景里有未保存的修改，3ds max 会显示标准的 Save Changes 对话框，提示用户是否保存。

另外，还可以用下面的函数来控制场景的 Save 和 Undo 状态：

2. setSaveRequired <Boolean>

为场景文件设置 dirty 标记，如果场景文件设置了 dirty 标记或 UNDO 缓存为非空，当用户执行操作 File | New 或 File | Reset 时，系统会提示用户是否保存。

3. getSaveRequired()

如果 dirty 标记为 True 或 Undo 缓存为非空，返回 True；如果 Undo 缓存为非空，本函数总是返回 True，即使刚刚调用函数 setSaveRequired False。

4. clearUndoBuffer()

清空 Undo 缓存。本函数有两个功能，重置场景 Undo 状态以及控制退出 3ds max 时是否显示 Save Changes 对话框。

一般情况下，当关闭场景时，如果 Undo 缓存为非空，3ds max 会提示用户是否保存。

5. resetMaxFile()[ #noPrompt ]

让用户在脚本里重置 3ds max，如果没有指定可选参数#noPrompt，而在场景里有未保存的修改，3ds max 会显示标准的 Save Changes 对话框，提示用户是否保存。

## 23.6 其他函数

1. freeSceneBitmaps()

释放所有由 Image 文件位图缓存占用的内存。当内存被许多位图文件分割成许多碎片时，可以用这个办法来释放内存。

2. rescaleWorldUnits <factor> [ #selOnly ]

本方法可实现 Rescale World Utility 插件的功能。参数<factor>为对象的缩放因子，如果指定了可选参数#selOnly，只有当前选择的对象被缩放。

3. IsNetServer()

如果 3ds max 在网络渲染模式下运行，返回 True；如果处于正常交互模式，返回 False。

4. loadDlIsFromDir <directory\_path\_String> <filename\_wildcard\_String>

装载指定路径下的插件。参数<directory\_path\_String>必须以“\”结束，请参见 3.2.2 节关于特殊字符输入的说明。

如：

```
LoadDlIsFromDir "f:\\maxsdk\\plugin\\\" \"*.dlc"
```

5. maxVersion()

返回一个三元数组，如#(3000, 6, 0)，分别表示 3ds max 版本号、max API 号、SDK 版本号。

6. swap <destination> <destination>

交换两个变量的值。如：

```
swap myMaterial.diffuseMap.map1 myMaterial.diffuseMap.map2
```