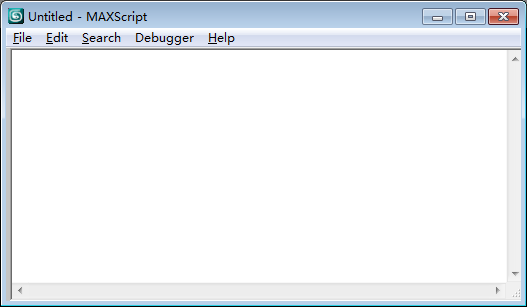
# Max 脚本

## 基本使用

### 使用方法

菜单栏->MAXScript--New Script可以打开编辑器。

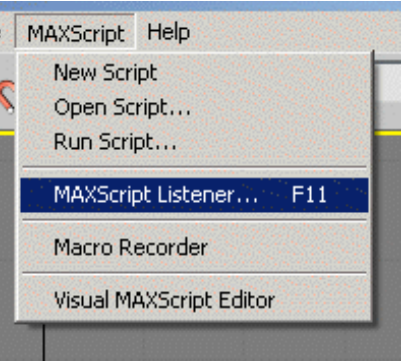


然后编辑器菜单中单击：File->Evaluate All, 执行代码。

执行代码的快捷键是Ctrl+E。（该快捷键经常冲突）

### MAXScript Listener 脚本侦听器

1. 菜单栏



1. 迷你侦听器上右键打开

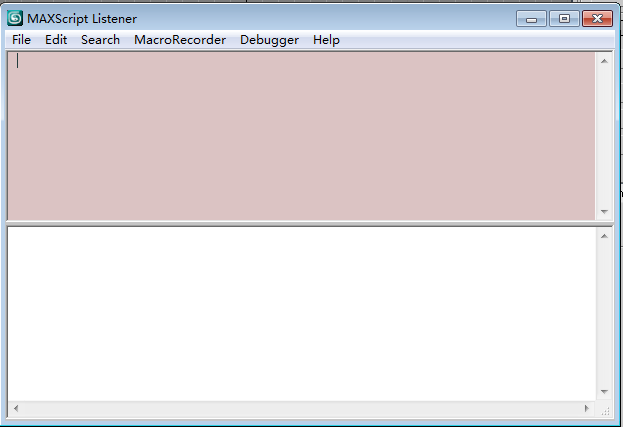


1. 点命令面板那个“锤子”，然后点击MaxScript，就出现



然后单击Open Listener.

**侦听器界面**：



**脚本侦听器**：

可以实时执行MaxScript，可以输入命令后用小键盘上的Enter马上执行，大键盘是shift+enter。

清除区域： Ctarl+D.

**迷你侦听器：**

Max的界面的左下角，是MaxScript的微缩版，每次只显示光标在MaxScript Listener中的那一行。

### Macro Recorder 宏记录器

路径：MaxScript->Macro Recorder.

Macro Recorder的功能是把我们的鼠标和键盘操作转化为脚本形式记录下来，但是有些操作是Macro Recorder无法记录的，还有的时候他会给你一些错误的信息，并且他生成的代码总是包含大量不必要的默认参数。

#### 变换的相对，绝对值选项

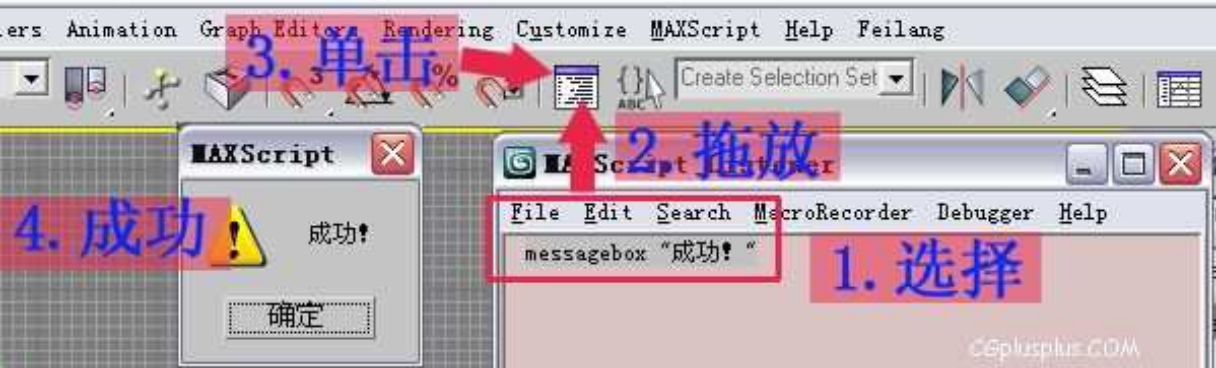


### Macroscript宏脚本

macroscript是一种生成工具栏按钮的脚本，当然也可以设置快捷键，右键菜单等。

#### 在工具栏上生成按钮

可以将maxscript listener中的代码段，选中后直接拖到工具栏上。



右键刚才的按钮，选择Edit Macro Script可以看到Macro Script的写法：



代码如下：

|  |
| --- |
| macroScript Macro1  category:"DragAndDrop"  toolTip:""  (  messagebox "helo"  ) |

简单修改版：

|  |
| --- |
| macroScript Macro1  category:"DragAndDrop"  toolTip:"这是我的按钮"  buttontext:"我的按钮"  (  messagebox "helo"  ) |

执行该脚本，然后在customize user interface->toolbars面板中，在category中找到DragAndDraop，就可以找到该按钮。



此时，可以把它拖至工具栏。



#### 制作按钮的图标

在刚才制作的按钮上，右键选择Edit Button Appearance, 可以修改按钮的外观。



max的图标目录:Autodesk\3ds max 2009\ui\icons。

具体图标特性，见Max软件特性。

使用的代码：

|  |
| --- |
| macroScript Macro1  category:"DragAndDrop"  toolTip:"这是我的按钮"  buttontext:"我的按钮"  icon:#("FeilangTools", 1)  (  messagebox "helo"  ) |

执行代码，然后到customize user interface 找到toolbars，选择DragAndDrop，可以看到刚才选项已经有图标了，在拖到工具栏上即可。

#### 调用外部代码文件

把代码 messagebox “ok!”存到e盘，格式改为.ms，然后修改macroscript代码：

|  |
| --- |
| macroScript Macro1  category:"DragAndDrop"  toolTip:"这是我的按钮"  buttontext:"我的按钮"  icon:#("ShowIcon", 1)  (  filein "E:\\hello.ms"  ) |

#### on execute do

当macroscript被“召唤“的时候，需要运行的代码。

示例：

|  |
| --- |
| macroScript Macro1  category:"DragAndDrop"  toolTip:"这是我的按钮"  buttontext:"我的按钮"  icon:#("ShowIcon", 1)  (  on execute do  (  s=sphere()  s.radius=100  )    on isChecked do selection.count==1 -- 当选择一个物体时，按钮被按下  ) |

#### 事件

语法: on event do <expr>

isChecked : 是否checked

isEnabled : 是否可用（不可用灰色）， on isEnabled do selection.count != 0

isVisible : 可见

closeDialogs : 关闭时

例子：

|  |
| --- |
| macroScript Macro1  category:"DragAndDrop"  toolTip:"这是我的按钮"  buttontext:"我的按钮"  icon:#("ShowIcon", 1)  (  rollout test "test" width:248 height:169  (  Label lbl "hello,world" pos:[20,40] width:225 height:13  HyperLink hy "www.hello.com" pos:[65,67] width:100 height:15 color:(color 0 255 0) hovercolor:(color 255 127 0) visitedcolor:(color 0 255 0)  address:"www.hello.com"  )    on execute do  (  createdialog test bgcolor:black fgcolor:red  )    on isChecked do test.open -- 这个效果不会马上显现，也许是dialog不会被马上删除吧    on closeDialogs do  (  destroydialog test  )  ) |



## 数据类型

### 常量

### 浮点型常量

### Integer整形常量

### String字符常量

### Time时间常量

### Color颜色常量

### booleanClass布尔常量

### max特殊变量：

### Name

### Point2

### Point3

3D空间中的一个点。表达形式:[数值，数值，数值]，如[0, 0, 0]表示坐标中心。

### Ray

### Quat

### AngleAxis

### EulerAngles

### Matrix3

### BigMatrix

### Box2

### BitArray

### ArrayParameter

### angleaxis

语法：angleaxis <degrees\_float> <axis\_point3>.

<degree\_float>是角度。

<axis\_point3>是方向向量。

如：angleaxis 30 [0,0,1], 就表示z轴相对30度，用到旋转上就是z轴正向转30度，同理，rotate $ (angleaxis 30 [0,0,-1])就是z轴负向转30度。

### eulerangles

语法：eulerAngles <x\_degrees> <y\_degrees> <z\_degrees>.

如：eulerAngles 30 30 0, 表示x轴30度，y轴30度，z轴0度。

### quat

## 作用域

### 全局变量

整个脚本都可以使用。

global

如: global str = “这个是全局变量”

注意：全局变量可以通用，只要你运行过脚本的3dmax没有关闭，那么全局变量还在内存里面，如果别人的脚本跟你用的全局变量名相同，就有可能出现错误。

为了避免这种情况，可以在变量前加个前缀。比如r\_\*.

### 局部变量

local

如： local str=”这是局部变量”

## 错误处理

### try...catch

## 注释

单行注释： --

多行注释： /\*...\*/

## 数组

Array Values 数组

数组中的数据，可以是任何数据类型，甚至表达式。

语法: #(, , …)

空数组 #()

取元素[]，序号从1开始。

.count : Integer 数组的元素个数，只能读取。

as array 把集合转变成数组。如：b = lights as array, 即把场景所有灯光存储为数组并赋值给b。

append 向数组里增加元素。如：append a “cg++”，就是想数组a里增加元素”cg++”.

deleteItem 删除数组某个元素。如：deleteitem a 4, 即把数组a的第四个元素删除。

findItem 寻找数组里面是否有某元素。如： finditem a “cg++”，如果找到，返回第一次找到此元素的需要，如果找不到，返回0.

例子: #(1, 2, “cgplusplus”, 5+6, 123 as float)

## 表达式

### 变量名

不区分大小写。

### 运算符

+

-

\*

/

^

as 强制类型转换(操作数as类型).

classof 判断一个max节点的类别。

语法: classof 节点

例：classof $

### 赋值表达式

## 流程控制

### IF

语法1： if <expr> then <expr> [else <expr>] 或者 if <expr> do <expr>

if then 和 if do的不同点是， if then 后面接表达式后还可以接else执行其他语句，而do后面只能接一个表达式，别无其他选择了。

### Case

语法: case [<expr>] of (<case>)

[<expr>]是可选表达式，<cases>是一系列可能会执行的内容。

<cases>语法：

<factor>:<expr>

default:<expr>

例：

case $.enabled of

(

true: print “打开的”

false:print “关闭的”

)

提供一个default:

|  |
| --- |
| a=1;b=1;  case of  (  (a>b):print "a大于b"  (a<b):print "a小于b"  default:print "其他"  ) |

### for loop

for循环可以逐个计算一列数值，集合，数组等等。

语法：for (in | =) (do | collect)

例1： for I in 1 to 10 do print i -- 输出1到10

例2： for I in 1 to 10 by 2 do print I -- 隔两个数输出1到10

例3： for i in selection do i.name += “new” -- 把所有选择物体的名字后面加一个”new”.

selection 即所有选择的物体。

selection.count 即选择物体的数量。

例4： for i in lights do i.on = false -- 把场景所有灯光关掉（这个lights包含了目标点）。

for i in lights where classof i != targetobject do i.on = false (过滤掉light目标点)

例5: for i in objects do i.name = uniquename “cg” 赋值唯一的名字。

把物体以所给字符串为基础命名，后面跟数字序号。

objects – 场景所有物体。

objects.count – 场景所有物体的数量。

例6:

prefix = “cg”

for i in objects do i.name = prefix + “\_” + i.name

将所有物体的名称价格前缀。

例7: for b in $box\* do b.name = uniquename “myBox” 重命名场景中所有名称以”box”开头的物体。

例8: for vl in lights where classof vl == VRayLight do vl.subdivs = 12 把场景中所有VRayLight的细分调到12.

例9: for g in geometry do g.wirecolor = red 把所有几何体的线框色改成红色。

## 函数

语法：(function|fn) <name> {<parameter>} = <expr>

例子：

fn p = print localtime

调用p()

例：将所有标准mat的diffuse改为一个指定的参数颜色

|  |
| --- |
| fn ranMtlColor col =  (  if classof col == color then  for i in meditmaterials where classof i == standardmaterial do i.diffuse = col  ) |

使用:

ranMtlColor blue

ranMtlColor (color 34 64 123)

### 形参附初值

|  |
| --- |
| fn ranMtlColor col:(color 255 0 0) =  (  if classof col == color then  for i in meditmaterials where classof i == standardmaterial do i.diffuse = col  ) |

### return表达式

例：

|  |
| --- |
| fn myfunction num =  (  if classof num == integer and num >= 1 then  for i in 1 to num do  (  if i == 50 then  return "提前返回"  )    format "完全执行\n"  ) |

### 递归

例：

|  |
| --- |
| fn ranMtlColor mat col =  (  if classof mat == standardmaterial then  mat.diffuse = col  else if calssof mat == Multimaterial then  for j in mat do ranMtlColor j col -- 调用自己  else()  ) |

### 解释性语言导致的函数的先后作用域

max的脚本显然是解释性语言，所以在脚本里面后面的函数可以调用前面的函数，但前面的不能调用后面的。

## 对物体的初步控制

### “.”语法

MaxScript中充当属性控制的作用。

Macro Recorder截出的操作代码：

|  |
| --- |
| select $Box01  $.length = 40  $.lengthsegs = 3 |

### “$”符号

单独出现表示当前选择的物体。

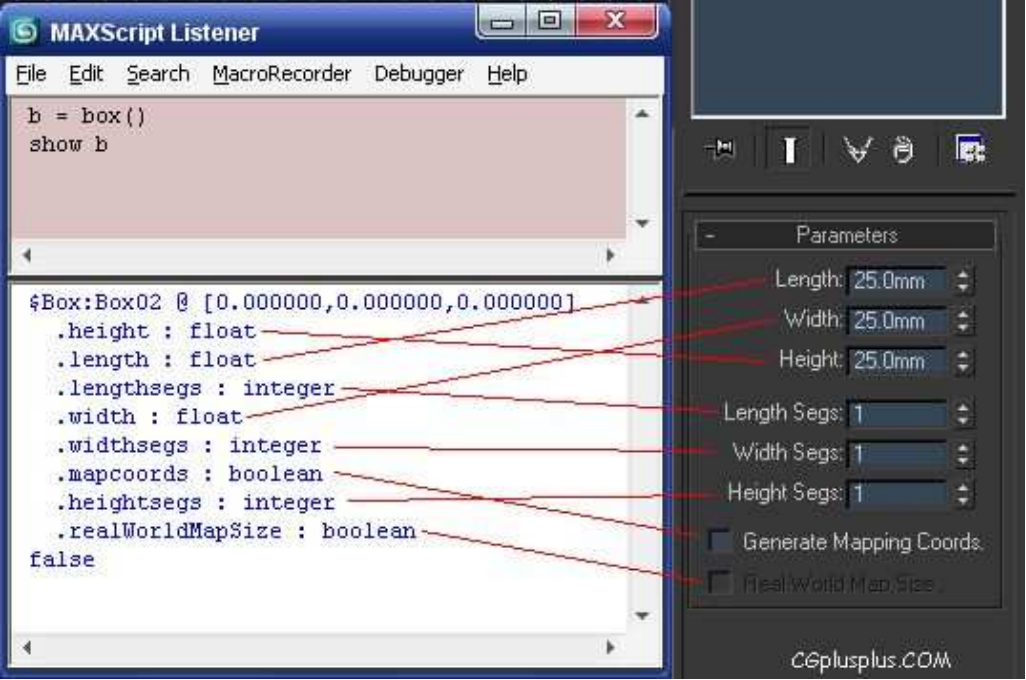
￥

$+物体名称 ： 对此物体的调用。

### Show(showproperties)

Show $ : 显示当前选中物体的属性。

简单例子：



### Move

默认是相对操作。

语法：

move $ [96.6942,240.569,0]

通过修改macroRecorder的变换绝对值选项，变成：

$.pos = [261.736,684.396,0]

操作单独的x,y,z轴：

$.pos.x = 100

$.pos.y = 100

$.pos.z = 100

### Scale

基本语法：

$.scale=[0.5,0.5, 0.5]

三个数值分别是在x,y,z轴上的数值。

### Rotate

语法1：

rotate <node> <angle> <axis\_point3> ,实例：rotate $ (angleaxis 30 [0,0,1]), 将选择物体在z轴旋转30度。

语法2：

rotate <node> <eulerangles>, 实例：rotate $ (eulerangles 0 0 30), 也将物体在z轴上旋转30度。

语法3：

rotate <node> <quat>

同理：

$.rotation.z\_rotation = 30

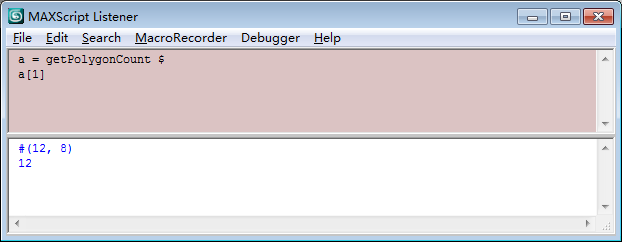
## 常用函数

### getPolygonCount

语法： getPolygonCount<node> 计算节点的面数和顶点数，返回一个数组。

返回值：#(1024, 530)

例：调取物体面数：



### messagebox

messagebox <字符串>

### filein

执行脚本文件

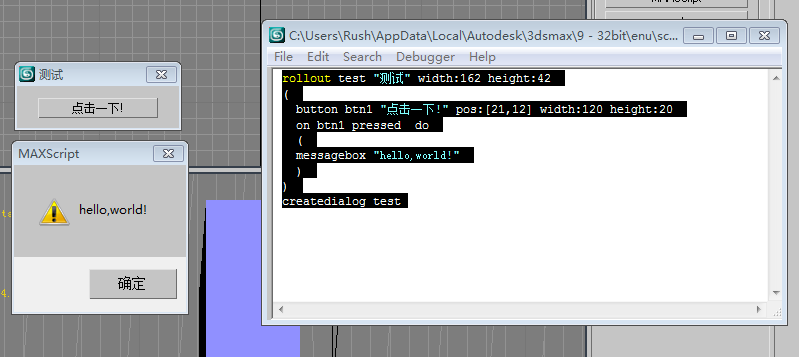
语法：filein “脚本路径”

支持格式: .ms, .mse, mcr (用filein调用.mcr没有意义，这里只是列出来)

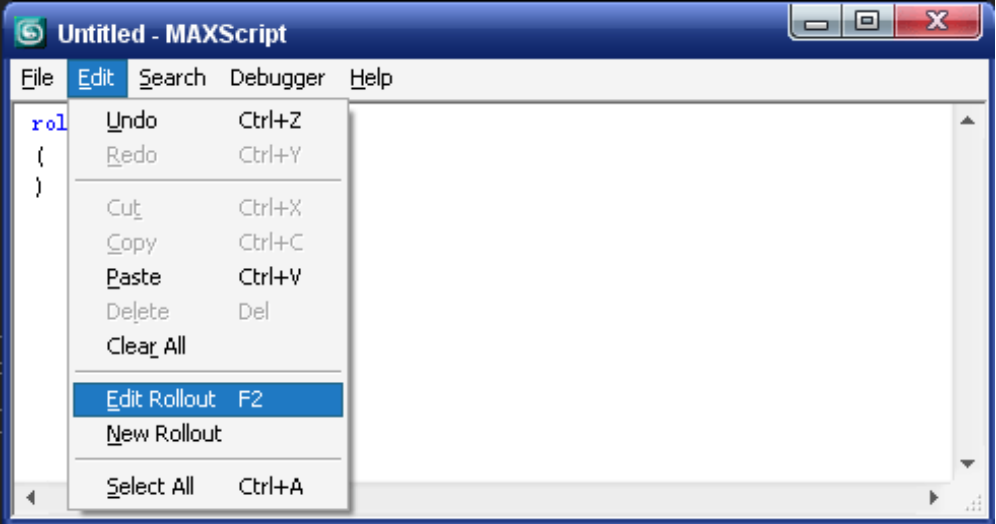
## Rollout（卷展栏） 技术

### Hello world

|  |
| --- |
| rollout test "测试" width:162 height:42  (  button btn1 "点击一下!" pos:[21,12] width:120 height:20  on btn1 pressed do  (  messagebox "hello,world!"  )  )  createdialog test |

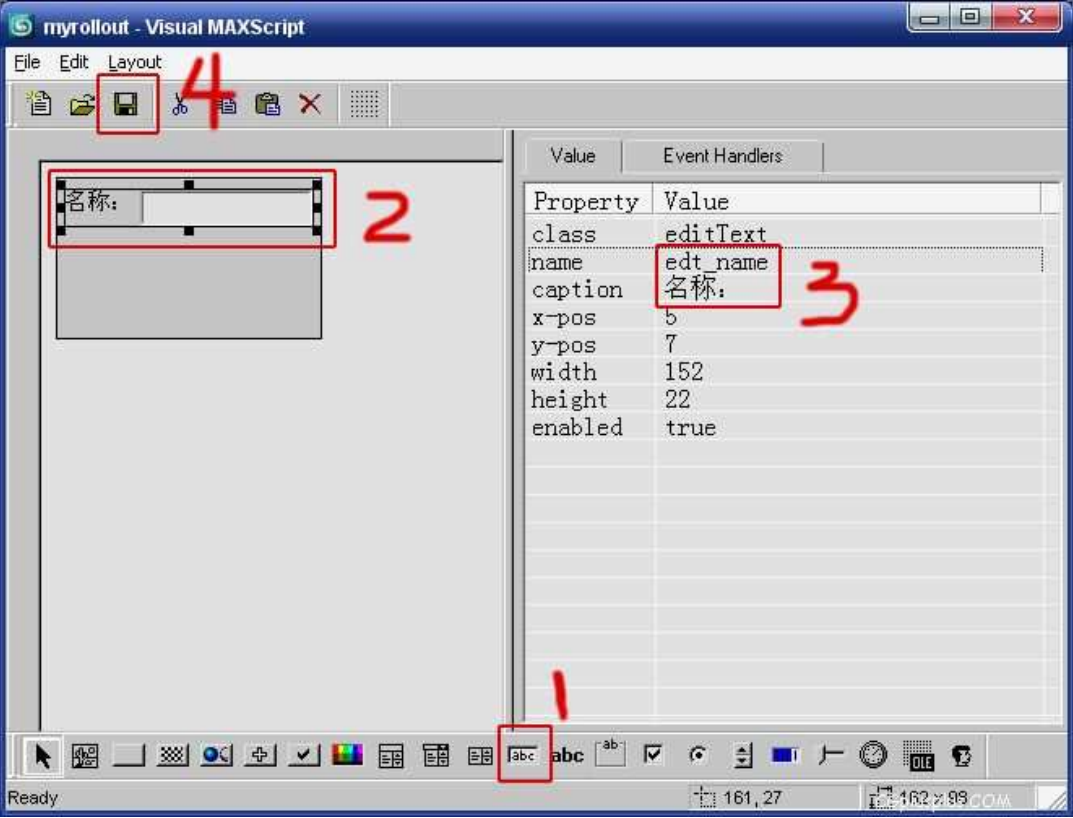


### 可视化编辑

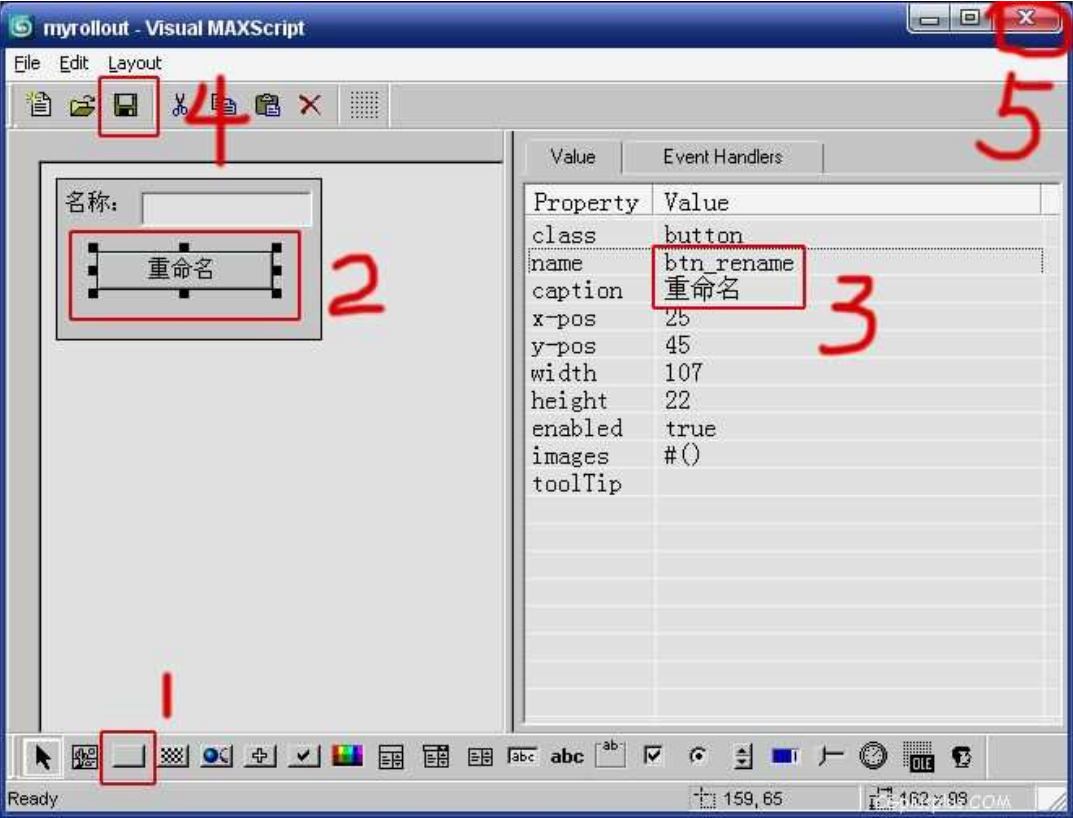




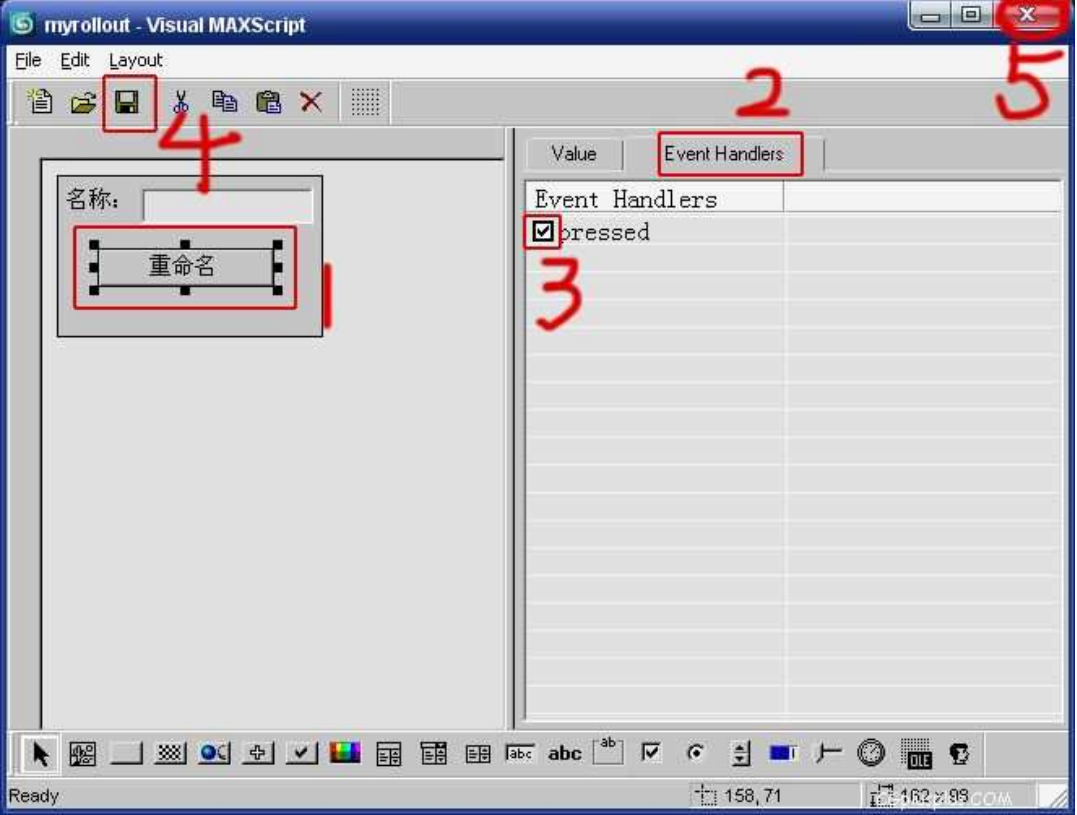
#### 创建一个文本控制器



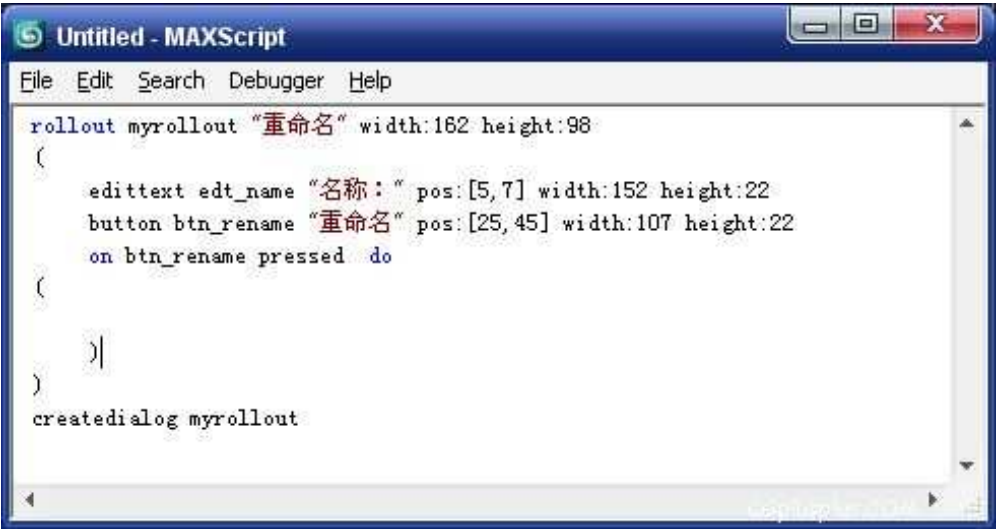
#### 创建一个按钮



### 事件处理



就得到了：



例子：

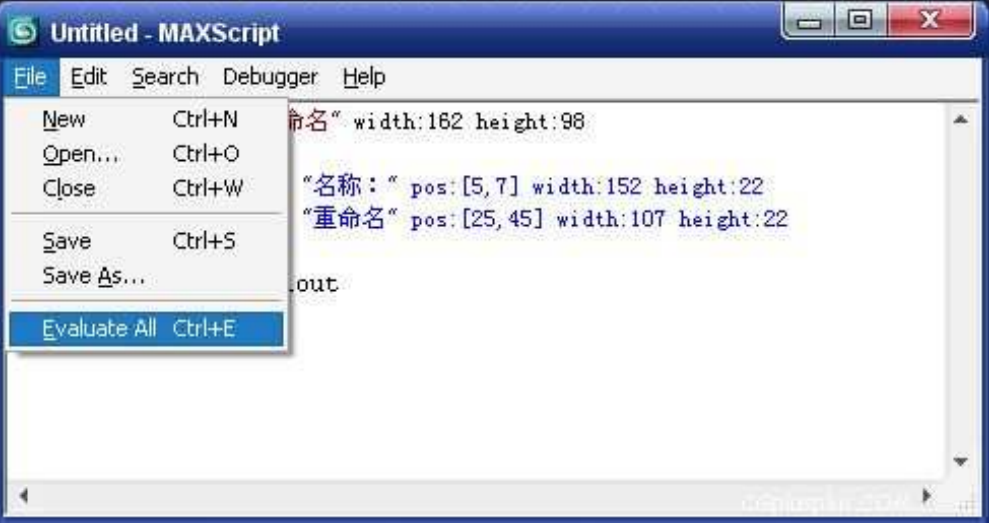
|  |
| --- |
| rollout myrollout "重命名" width:181 height:188  (  edittext edt\_name "名称" pos:[17,20] width:146 height:30  button btn\_rename "重命名" pos:[15,114] width:153 height:35  on btn\_rename pressed do  (  if edt\_name.text != "" then  for i in selection do i.name = uniquename edt\_name.text  )  )  createdialog myrollout |

### 常用接口函数

#### createdialog

例：

|  |
| --- |
| rollout myrollout "重命名" width:181 height:188  (  editText edt\_name "名称" pos:[17,20] width:146 height:30  button btn\_rename "重命名" pos:[15,114] width:153 height:35  )  createdialog myrollout |



结果



#### detroydialog

## Max软件特性

### 图标目录/制作要求

目录：Autodesk\3ds Max 2009\ui\icons

要求:

1. 图标大小分两种，24×24， 16×15, 其中，图标的高度不能变，高度是24的图片，宽度必须是24的整数倍，高度是15的图片，快读必须是16的整数倍。两个图标的倍数要相同。即图标大小应该为：(N\*24)\*24和(N\*16)\*15.
2. 一组图标必须有4个，命名规则是，使用统一的前缀，后缀名字依次是-16a.bmp, -16i.bmp,\_24a.bmp,\_24i.bmp， i结尾的是最终显示在max界面上的图标，a结尾的图标是黑白统统，控制图标需要显示的部分。
3. 图标文件必须放到max根目录的ui\Icons下，max需要重启才能识别。

## 实例

### 1.选择面数为0的物体。

|  |
| --- |
| 1. obj = #() 2. for i in geometry do 3. ( 4. a = getPolygonCount i 5. if a[1] == 0 then append obj i 6. ) 7. if obj.count != 0 then select obj else print "没有面数为0的物体!" |

2. 找所有没有上材质的物体(提示：没材质的判断方法为:$.material == undefined)

3. 收集场景中的所有standard材质(提示：用classof判断材质类型)

4. 调场景中所有的VRayLight的细分到20.

5. 把场景中所有类型为Teapot的线框颜色改为红色。

### 2．幻世的固定cam视角

|  |
| --- |
| macroScript Macro1  category:"HsUtility"  toolTip:"幻世摄像机"  buttontext:"幻世摄像机"  icon:#("Cameras", 1)  (    on execute do  (  Targetcamera fov:45 nearclip:1 farclip:1000 nearrange:0 farrange:1000 mpassEnabled:off mpassRenderPerPass:off pos:[10,-10,12] isSelected:on target:(Targetobject transform:(matrix3 [1,0,0] [0,1,0] [0,0,1] [0,0,0]))  select $  $.name="HsCam"  macros.run "Lights and Cameras" "Camera\_ActivateView"  )    on isChecked do ($HsCam != undefined)    on closeDialogs do delete $HsCam  ) |