TEXAS INSTRUMENTS

TI-nspire™

Lua 脚本语言 API 参考指南

本参考手册适用于 TI-Nspire™软件 3.4 版本。要获得最新版本的文档，请访问 *education.ti.com/nspire/scripting*

重要信息

除非在程序附带的《许可证》中明示声明，否则 Texas Instruments 不对任何程序或书面材料做出任何明示或暗示担保，包括但不限于对某个特定用途的适销性和适用性的暗示担保，并且这些材料均以“ 原样” 提供。任何情况下，Texas Instruments 对因购买或使用这些材料而蒙受特殊、附带、偶然或连带损失的任何人都不承担任何责任。无论采用何种赔偿方式，Texas Instruments 的唯一且排他性义务不得超出本程序许可证规定的数额。此外，对于任何其他方因使用这些材料而提起的任何类型的索赔，Texas Instruments 概不负责。

商标及版权

TI-Nspire™ 软件使用 Lua 作为脚本语言环境。版权和许可信息请访问
http://www.lua.org/license.html
TI-Nspire™软件 使用 Chipmunk Physics 作为仿真环境。 版权和许可信息请访问
http://chipmunk-physics.net/release/ChipmunkLatest-Docs/
Microsoft®和 Windows®是 Microsoft Corporation 在美国和其他国家/地区的注册商标
Mac®， iPad® 和 OS X® 是 Apple Inc. 的注册商标
Unicode®是 Unicode， Inc. 在美国和其他国家/地区的注册商标。

目录

表格列表

第一章

标准库（Standard Libraries）

TI-Nspire™软件整合了大部分 Lua 标准库。本章节概述了支持的 Lua 库函数以及限制。
标准函数的解释及其细节参见《Lua 5.1 Reference Manual》。

1.1 基本库函数

更多细节请点击下面的链接，将会跳转到《Lua 5.1 Reference Manual》的"Basic Functions"章节。

assert	**collectgarbage**	**error**	**_G**	**getfenv**	**getmetatable**
ipairs	**load** [1]	**loadstring** [1]	**next**	**pairs**	**pcall**
print [2]	**rawequal**	**rawget**	**rawset**	**select**	**setfenv**
Setmetatabl	**tonumber**	**tostring**	**type**	**unpack**	**_VERSION**
xpcall					

[1] 请小心使用 **load** 和 **loadstring** 函数，在 TI-Nspire™编辑器中不支持使用这些函数直接执行的 Lua 源代码，这些源代码将无法调试以及正确的返回错误信息，使用 **load** 和 **loadstring** 函数可能会导致错乱的结果。

[2] **print** 函数输出的内容仅被重定向到 TI-Nspire™编辑器的控制台 (console) 中。在任何不包含 TI-Nspire™编辑器的平台中调用 **print** 函数会被忽略。

1.1.1 协同程序子库（Coroutine Sub-Library）

更多细节请点击此链接跳转到《Lua 5.1 Reference Manual》中的"Coroutine Manipulation"一节。下面的函数在 Lua 的协同程序表（**coroutine** table）中定义。大量使用协同程序可能导致在 TI-Nspire™编辑器中难以调试。

create	**resume**	**running**	**status**	**wrap**	**yield**

1.2 模块库（Module Library）

模块库的实现是非常有限的。更多细节请参照第十三章。

1.3 字符串库（String Library）

更多细节请点击此链接跳转到《Lua 5.1 Reference Manual》中的“String Manipulation”一节。

字符串函数 **upper** 和 **lower** 可能不适合当前的区域设置。在字符串中使用这些函数只有 26 个英文字母会进行大小写转换。此限制也同样适用于 **find**， **gmatch**，和 **match** 函数参数中的模式串匹配符（即%a，%l，%u 和%w）。

byte **char** **dump** **find** **format** **gmatch** **gsub** **len**
lower **match** **rep** **reverse** **sub** **upper**

1.4 表库（Table Library）

更多细节请点击此链接跳转到《Lua 5.1 Reference Manual》中的“Table Manipulation” 一节。

create insert maxn remove sort

1.5 数学库（Math Library）

更多细节请点击此链接跳转到《Lua 5.1 Reference Manual》中的“Mathematical Functions”一节。下面的函数在数学表（**math** table）中定义。无限大和未定义的结果将转换到相应的 TI-Nspire™表示并且与 TI-Nspire™ 数学扩展相配合。不支持字符串表达式（无限大和未定义）到数值表示的反向转换。

abs **acos** **asin** **atan** **atan2** **ceil** **cos** **cosh**
deg **exp** **floor** **fmod** **frexp** **huge** **ldexp** **log**
log10 **max** **min** **modf** **pi** **pow** **rad** **random**
randomseed **sin** **sinh** **sqrt** **tan** **tanh**

1.6 未实现的库和函数

以下标准 Lua 库在 TI-Nspire™软件中不可用：

file **io** **os** **debug**

下面的标准函数和标准表条目（standard table entries）在 TI-Nspire™软件中未提供：

dofile **loadfile** **module** **package.cpath** **package.loadlib**
package.path **package.seeall**

第二章

触摸库（Touch Library）

触摸库(Touch Library)是在 **platform.apiLevel = '2.2'**中为 TI-Nspire™平台新增的库。在所有平台上这个库均可用，但是在不支持触摸的平台上相关函数的调用可能会被忽略。

触摸库提供了一个低级别的接口，以允许脚本语言的作者在任何平台下平等的开发脚本语言。如果希望实现平台兼容性，那么脚本语言作者应在全部不同平台上进行设计和测试。

2.1 概述

接下来会概述系统特征和脚本语言行为，脚本语言作者应当了解这些内容以便成功的为支持触摸的平台设计脚本语言和编写跨平台的脚本语言。

2.1.1 屏幕键盘及屏幕尺寸调整行为

TI-Nspire™软件包含两种屏幕键盘—字母键盘 (ABC keyboard) 和函数键盘 (Function keyboard) 。用户可以在这两种键盘之间切换。脚本语言环境的默认键盘是字母键盘。

每种支持触摸的平台上可能有不同的键盘模式 (keyboard modes)—停靠(docked)，不停靠(undocked)和分离 (split)。不论在哪种模式下都不会发生 resize 事件。如果键盘是停靠模式，TI-Nspire™平台会允许用户拖移(pan)屏幕以访问在键盘背后的内容—参见 setScrollHeight()以了解当屏幕显示停靠的键盘时，脚本语言如何控制展开(controlling scrolling)。新的事件 on.keyboardUp() 支持处理键盘高度超过屏幕内容时的情况。支持触摸的平台通常支持手势拖移来不停靠和分离屏幕键盘。因而，脚本语言不需要处理拖移键盘的情况。

2.1.2 事件处理

所有的事件处理在第八章中都有详细的描述。在支持触摸的平台上，即在 platform.apiLevel = '2.2' 中除了加入了两种新的事件处理程序，以处理屏幕键盘显示/隐藏—参见 on.keyboardUp(keyboardOverlapHeight) 和 on.keyboardDown()事件处理程序。请参见表 2.1 来了解每种手势与其对应的事件处理程序。

表 2.1: 手势与其对应的事件处理程序

手势	“on” 事件处理程序	备注
单击（Single Tap）	**on.mouseDown()** **on.mouseUp()**	应当注意该手势识别器会在手指离开屏幕时和触发 **mouseUP** 事件之间添加一个延迟。
双击(Double Tap)	**on.mouseDown()** **on.mouseUp()** **on.mouseDown()** **on.mouseUp()**	事件在第二次轻敲屏幕完成以后触发。但是接下来的 **mouseDown** 和 **mouseUp** 事件会立刻触发。
拖移(Pan)	**on.mouseDown()** **on.mouseMove()**’s ... **on.mouseUp()**	类似一种在桌面平台上先按下鼠标，然后拖动鼠标，最后重新释放鼠标的行为。当在不支持触摸的平台中运行时，即使鼠标没有按下也会触发 **on.mouseMove()。**
长按移动(Long Press Move）	**on.mouseDown()** **on.mouseMove()’s** **on.mouseUp()**	和拖移是一样的行为。脚本语言不能区分二者。
其他手势	**on.mouseDown()** [**on.mouseMove()**’s] **on.mouseUp()**	会可靠的触发一个 **on.mouseDown()**和 **on.mouseUp()**事件。一个或多个 **on.mouseMove()** 可能会被触发。多手指手势报告的坐标将低于或在手指之间。

注意：

表 2.1 中描述的行为与 D2Editor:registerFilter()中注册的鼠标事件处理程序略有不同。在单击与双击的情况下，第一个 **on.mouseDown()** 事件会在手势完全识别及手指离开屏幕以后触发。与此类似的是拖移与长按手势。**on.mouseDown()**事件会在手指开始移动或者保持不动一段时间时触发。

另一个重要的方面，一个事件处理程序的返回值是和事件处理有关联的。这主要用于 **platform.apiLevel = '2.0'**下富文本框中注册的事件过滤器—参见 D2Editor:registerFilter()。每个事件处理程序可能返回一个布尔值(boolean)来表明一个事件是已经被处理了(**true**)还是被忽略(**false**)，如果没有明确的返回值，缺省为 **true。**

在处理触摸操作和屏幕键盘时，如果显示了键盘，那么 **mouseDown** 的返回值是很重要的，而且不正确地使用可能会干扰用户的体验。当屏幕键盘显示时用户可以拖移屏幕以看到在键盘背后的内容。如果 **mouseDown** 返回 **true**，或者没有明确的返回值(即默认为 **true**)，会阻止用户拖移屏幕。

2.2 库函数

2.2.1 ppi

```
touch.ppi()
```

返回屏幕沿对角线每英寸的像素数(pixels per inches， PPI)。这个函数对于确定屏幕上触摸目标的可触摸对象尺寸是很有用的。

限制：
ppi 函数 在 **platform.apiLevel = '2.2'**中引入，当前并不是完全可用。这不能用于区分 iPad®版本—通常得到 iPad®2 的像素密度。

API 级别限制：最低 platform.apiLevel = '2.2'

2.2.2 xppi

```
touch.xppi()
```

返回屏幕沿 x 轴每英寸的像素数(pixels per inches， PPI)。这个函数对于确定屏幕上触摸目标的可触摸对象尺寸是很有用的。

限制：
ppi 函数 在 **platform.apiLevel = '2.2'**中引入，当前并不是完全可用。这不能用于区分 iPad®版本—通常得到 iPad®2 的像素密度。

API 级别限制：最低 platform.apiLevel = '2.2'

2.2.3 yppi

```
touch.yppi()
```

返回屏幕沿 y 轴每英寸的像素数(pixels per inches， PPI)。这个函数对于确定屏幕上触摸目标的可触摸对象尺寸是很有用的。

限制：
ppi 函数 在 **platform.apiLevel = '2.2'**中引入，当前并不是完全可用。这不能用于区分 iPad®版本—通常得到 iPad®2 的像素密度。

API 级别限制：最低 platform.apiLevel = '2.2'

2.2.4 enabled

```
touch.enabled()
```

如果当前平台支持触摸那么返回 **true**，否则返回 **false**。如果平台支持触摸，建议使用 ppi 来计算触摸对象的尺寸。

API 级别限制：最低 platform.apiLevel = '2.2'

2.2.5 isKeyboardAvailable

```
touch.isKeyboardAvailable()
```

如果当前平台支持屏幕键盘那么返回 **true**，否则返回 **false**。

API 级别限制：最低 platform.apiLevel = '2.2'

2.2.6 isKeyboardVisible

```
touch.isKeyboardVisible()
```

如果当前有任何键盘可见（停靠(docked)，不停靠(undocked)或者分离 (split)的键盘）那么返回 **true**。

API 级别限制：最低 platform.apiLevel = '2.2'

2.2.7 showKeyboard

```
touch.showKeyboard(boolean)
```

如果当前屏幕没有可见的键盘那么显示一个停靠的字母键盘（docked ABC keyboard）。默认为 **true**。

API 级别限制：最低 platform.apiLevel = '2.2'

第三章

富文本框库（2D Editor Library）

可以使用 TI-Nspire™产品来创建和操纵 Lua 的 2D 富文本框(2D Editor)。 富文本框可以使用 newRichText()创建。

脚本语言作者应当知道富文本框可以嵌入一种专用的标记语言(markup language)。可以通过调用 createMathBox 或 createChemBox 来在脚本语言中嵌入这种标记。

脚本语言应用的用户也可以从 TI-Nspire™其他应用中复制和粘贴其他带有标记信息的文本（例如记事本(Notes) 应用）。富文本框中使用标记语言的相关信息参见表 3.1。

表 3.1: 富文本框标记语言

描述	标记	备注
数学框	“\0el {...}”	包含一个自然书写形式的表达式。
计算后的数学框	“\0el {...} >’ \0el {...}”	一对数学框—包含一个表达式及其计算结果。
化学框	“\0chem {...}”	包含一个化学方程式。
其它标记	“\1 ...\”	不建议在使用这种标记，因为可能在未来的版本中更改。但是建议脚本语言不带任何 Lua 错误地对其进行处理。

3.1 newRichText

```
D2Editor.newRichText()
```

创建并返回一个新的富文本框。

注意
在富文本框首次绘制之前，必须手工调整富文本框的尺寸。

富文本框的默认设置
一个新富文本框使用以下默认设置创建：

```
:move(0,  0)
:setBorder(0)
:setBorderColor(0x000000)
:setColorable(false)
:setDisable2DinRT(false)
:setFontSize(<系统默认字号>)
:setMainFont(<系统默认字体>)
:setReadOnly(false)
:setSelectable(true)
:setTextColor(0x000000)
:setVisible(true)
```

API 级别限制：最低为 platform.apiLevel = '1.0'

3.2 createChemBox

```
D2Editor:createChemBox()
```

在当前富文本框的光标处插入一个化学框。

返回该富文本框对象。

API 级别限制：最低为 platform.apiLevel = '2.0'

3.3 createMathBox

```
D2Editor:createMathBox()
```

在当前富文本框的光标处插入一个数学框（表达式框）。

返回该富文本框对象。

API 级别限制：最低为 platform.apiLevel = '2.0'

3.4 getExpression

```
D2Editor:getExpression()
```

将当前富文本框内容以一个 UTF-8 编码的字符串返回。

API 级别限制：最低为 platform.apiLevel = '2.0'

3.5 getExpressionSelection

```
D2Editor:getExpressionSelection()
```

该函数返回三个值：
（1）当前富文本框的内容以一个 UTF-8 编码的字符串返回。
（2）当前的光标位置作为一个整数返回。
（3）被高亮选中文本的开始位置作为一个整数返回。

用法
光标和高亮选中文本开始的位置是指字符间的边界，而不是字符在字符串中的位置。

例如以下的两个代码段：

```
str = 'This  is a test string to  see it working.'
d2e,  error = D2Editor.newRichText()
result, error = d2e:setText(str, 16,  28)
str, pos,  sel, error = d2e:getExpressionSelection()
```

以上代码执行后的结果：
str = 'This is a test string to see it working.'
pos = 16 (在"string"单词中字母"s"的右侧)
sel = 28 (在单词"see"中两个字母"e"的中间)

```
str = 'This  is a test string to  see it working.'
d2e,  error = D2Editor.newRichText()
result, error = d2e:setText(str, 28,  16)
str, pos,  sel, error = d2e:getExpressionSelection()
```

以上代码执行后的结果：
str = 'This is a test string to see it working.'
pos = 28 (在单词"see"中两个字母"e"的中间)
sel = 16 (在"string"单词中字母"s"的右侧)

API 级别限制：最低为 platform.apiLevel = '2.0'

3.6 getText

```
D2Editor:getText()
```

将当前富文本框内容以一个 UTF-8 编码的字符串返回。

API 级别限制：最低为 platform.apiLevel = '1.0'

3.7 hasFocus

```
D2Editor:hasFocus()
```

如果当前富文本框具有焦点，那么返回 true，否则返回 false。

API 级别限制：最低为 platform.apiLevel = '2.0'

3.8 isVisible

```
D2Editor:isVisible()
```

如果当前富文本框是可见的，那么返回 true，否则返回 false。

API 级别限制：最低为 platform.apiLevel = '2.0'

3.9 move

```
D2Editor:move(x, y)
```

设置当前富文本框左上角的坐标。x 和 y 都必须在-32767 和 32767 之间。

返回当前富文本框对象。

API 级别限制：最低为 platform.apiLevel = '1.0'

3.10 registerFilter

```
D2Editor:registerFilter(handlerTable)
```

此函数能够注册一个事件处理函数表(Table)，或者通过传递一个 nil 来注销事件处理函数表。这些处理函数能够在事件被富文本框响应之前进行过滤，并决定是否将事件继续传递给富文本框。

返回当前富文本框对象。

参数 **handlerTable** 是一个事件处理函数的表(table)。任何在事件处理（Event Handling）一节中所描述的事件都可以被处理函数表中对应的函数所处理。

以下示例代码中，如果用户在富文本框 **ed** 中按下 Tab 键，**tabKey** 事件过滤器会将焦点移动到富文本框 **ed2**。而事件 **charIn** 和 **arrowKey** 只是简单地报告哪个键被按下，然后允许事件传递到当前富文本框。

```
-- 创建一个富文本框
ed = D2Editor.newRichText()

-- 注册事件过滤器
ed:registerFilter {
  tabKey = function()
                ed2:setFocus()
                return true
            end,
  charIn  = function(ch)
                print(ch)
                return false
            end,
  arrowKey = function(key)
                print(key)
                return false
            end
}
```

API 级别限制：最低为 platform.apiLevel = '2.0'

3.11 resize

```
D2Editor:resize(width, height)
```

更改文本编辑器的宽度和高度。**width** 和 **height** 必须大于 0 且小于 32768。

返回当前富文本框对象。

API 级别限制：最低为 platform.apiLevel = '1.0'

3.12 setBorder

```
D2Editor:setBorder(thickness)
```

设置富文本框的边框厚度。参数 thickness 必须在 0 到 10 之间。

返回当前富文本框对象。

API 级别限制：最低为 platform.apiLevel = '2.0'

3.13 setBorderColor

```
D2Editor:setBorderColor(color)
```

设置富文本框的边框颜色。 参数 color 必须介于 0 和 16777215（即 0x000000 和 0xFFFFFF）之间。

返回当前富文本框对象。

API 级别限制：最低为 platform.apiLevel = '2.0'

3.14 setColorable

```
D2Editor:setColorable(true or  false)
```

设置富文本框的内容能否被用户设置颜色。

返回富文本框对象。

API 级别限制：最低为 platform.apiLevel = '2.0'

3.15 setDisable2DinRT

```
D2Editor:setDisable2DinRT(true or  false)
```

关闭富文本框中输入的数学表达式的教材显示(2D layout of math)。

返回富文本框对象。

API 级别限制：最低为 platform.apiLevel = '2.0'

3.16 setExpression

```
D2Editor:setExpression(text[, cursor[,  selection[, full-precision]]])
```

将参数 text 的值设置为富文本框的内容。参数 cursor 代表光标位置，它的值可以设置为 1（文本开头），-1（文本末尾）或从 1 到 text 的长度加 1 范围内中的一个值。参数 selection 指定一个所要高亮选中的文本的结束位置。如果 selection 为-1，那么没有文本被选中。如果 cursor<-1 或 selection<-1，则返回一个错误。如果未指定，则 cursor 和 selection 默认为-1。最后一个可选参数 full-precision 如果为 true，将使用全精度，这就是说计算结果的所有位应全部显示。如果为 false，表示计算结果应按照富文本框设置的精度进行舍入。

注意
所有发送到富文本框中的反斜杠必须双写。这一点是附加在特殊符号的标准转义规则之上的。因此，使富文本框显示 **home\stuff\work** 的字符串是"home\\\\stuff\\\\work"。

用法
光标和高亮选中文本的位置是指字符间的边界，而不是字符在字符串中的位置。

下面的代码段高亮显示文本"string to se"，并将光标放在's'之前。

```
str = 'This  is a test string to  see it working.'
d2e,  error = D2Editor.newRichText():resize(100,100)
result, error = d2e:setExpression(str, 16,  28)
```

下面的代码段高亮显示文本"string to se"，并将光标置于"see" 的第二个'e'之前。

```
str = 'This  is a test string to  see it working.'
d2e,  error = D2Editor.newRichText():resize(100,100)
result, error = d2e:setExpression(str, 28,  16)
```

返回富文本框对象。

API 级别限制：最低为 platform.apiLevel = '2.0'

3.17 setFocus

```
D2Editor:setFocus(true or  false)
```

如果输入参数为 true（默认）则将用户输入焦点置于当前富文本框。通常在事件 on.getFocus 处理时调用。

返回当前富文本框对象。

API 级别限制：最低为 platform.apiLevel = '2.0'

3.18 setFontSize

```
D2Editor:setFontSize(size)
```

设置富文本框中的字号。在 TI-Nspire™ CX 和其他旧的手持设备中，字号受限于仅能从以下字号中选择一个：7，9，10，11，12，24。但是任何 Windows®或 Mac OS X®所支持的字号都能够在桌面软件中使用。

返回当前富文本框对象。

API 级别限制：最低为 platform.apiLevel = '2.0'

3.19 setMainFont

```
D2Editor:setMainFont(family, style)
```

设置字体("serif"(有衬线字体) 或"sansserif"(无衬线字体)) 与样式 ("r", "b", "i", "bi")。

样式	描述
r	常规
b	加粗
i	倾斜
bi	加粗并倾斜

返回当前富文本框对象。

注意
此函数只影响当前富文本框中的文本。对于此后使用 setText，setExpression，或 setFormattedExpression 设置的文本仍会使用默认字体。

API 级别限制：最低为 platform.apiLevel = '2.0'

3.20 setReadOnly

```
D2Editor:setReadOnly(true or false)
```

设置富文本框内容可以被用户修改(false)或不能被用户修改(true)。如果未指定一个明确的布尔值(boolean)那么默认为 true。

返回富文本框对象。

API 级别限制：最低为 platform.apiLevel = '1.0'

3.21 setSelectable

```
D2Editor:setSelectable(true or false)
```

设置富文本框的内容可以被用户选择(true)或者不能被用户选择(false)。如果未指定一个明确的布尔值(boolean)那么默认为 true。

返回富文本框对象。

API 级别限制：最低为 platform.apiLevel = '1.0'

3.22 setSizeChangeListener

```
D2Editor:setSizeChangeListener(function(editor, w, h))
```

设置一个回调函数(callback function) 。当富文本框内容超出当前尺寸时，或能够以较少行数容纳内容时，或内容适合在较小的宽度的单独的一行上显示时会调用指定的回调函数。这个回调函数是一个 void（即没有返回值）函数，能够调整编辑器到适当的大小。它接受以下参数传递：

参数	描述
editor	要改变尺寸的富文本框对象
w	优化后可适应显示内容的宽度
h	优化后可适应显示内容的高度

返回富文本框对象。

信息
要移除监听器(listener)，调用 D2Editor:setSizeChangeListener(nil)

API 级别限制：最低为 platform.apiLevel = '2.0'

3.23 setText

```
D2Editor:setText(text[, cursor[,  selection[, full-precision]]])
```

更多细节参见 setExpression()。

返回富文本框对象。

API 级别限制：最低为 platform.apiLevel = '1.0'

3.24 setTextChangeListener

```
D2Editor:setTextChangeListener(function(editor))
```

设置一个回调函数(callback function) 。当富文本框内容的改变时会调用指定的回调函数。该函数会被传递一个当前富文本框对象。这样可以即时处理文字输入。

返回富文本框对象。

信息
要移除监听器(listener)，调用 D2Editor:setTextChangeListener(nil)。

API 级别限制：最低为 platform.apiLevel = '2.0'

3.25 setTextColor

```
D2Editor:setTextColor(color)
```

设置富文本框文本的颜色。参数 color 必须介于 0 和 16777215（即 0x000000 和 0xFFFFFF）之间。

返回富文本框对象。

API 级别限制：最低为 platform.apiLevel = '2.0'

3.26 setVisible

```
D2Editor:setVisible(true or  false)
```

设置富文本框的可见性。

返回富文本框对象。

API 级别限制：最低为 platform.apiLevel = '2.0'

3.27 setWordWrapWidth

```
D2Editor:setWordWrapWidth(width)
```

设置富文本框自动换行(word-wrapping)的宽度。如果编辑器在 2D 模式下，则被忽略。若要指定为富文本框的宽度，参数需设置为 0。若要禁用自动换行，参数需设置为小于 0 的值。 参数 width 必须在 -32767 到 32767 之间。

注意
当自动换行被禁用时，即 width< 0, 会添加一个省略标识来分割文本。负的 width 值指定在省略标识使用之前从富文本框右侧的边缘开始分割文本。

返回富文本框对象。

API 级别限制：最低为 platform.apiLevel = '2.0'

第四章

类库（Class Library）

类库实现了基本的面向对象编程中类的定义。

4.1 class

```
class([parent_class])
```

返回一个新的类。如果指定了一个父类，则新类将会继承父类。

```
Widget = class()
function Widget:init()  ... end

Button  = class(Widget)
function Button:init()  ... end
```

例如上面的例子，当脚本语言调用 Button()时，一个新的 Button 对象将被创建。 The Button:init()函数被调用来进行初始化，新生成的 Button 象会作为函数调用的结果返回。

在这个例子中的 Button 类继承 Widget 类中定义的所有方法和属性。Button 类可以覆盖父类的任何方法。

API 级别限制：最低为 platform.apiLevel = '1.0'

第五章

剪贴板库（Clipboard Library）

5.1 addText

```
clipboard.addText(string)
```

将字符串的内容以纯文本方式复制到剪贴板中，MIME 类型“text/plain”。

API 级别限制：最低为 platform.apiLevel = ‘1.0’

5.2 getText

```
clipboard.getText()
```

这个例程将剪切板中的内容以纯文本字符串的形式返回。如果剪切板不包含任何文本(MIME 类型“text/plain”)，返回 nil。

API 级别限制：最低为 platform.apiLevel = ‘1.0’

第六章

光标库（Cursor Library）

这个光标库可以控制鼠标指针的外观。只能够在手持设备上控制鼠标指针的可见性。

一个好的做法是在 **on.activate()**事件中设置所期望的鼠标指针外观。在脚本语言停止(deactivated)过程中对于鼠标指针库的调用将会被忽略(在 **on.deactivate()**被触发以后)。

6.1 set

```
cursor.set(cursorname)
```

参数 **cursorname** 是一个说明想要使用的鼠标指针形状的字符串。可以是以下字符串之一：“default”，“interrogation”，“crosshair”，“text”，“pointer”，“link select”，“diag resize”，“wait busy”，“hollow pointer”，“rotation”，“pencil”，“zoom box”，“hide”，“arrow”，“zoom out”，“dotted arrow”，“clear”，“animate”，“excel plus”，“mod label”，“writing”，“unavailable”，“resize row”，“resize column”，“drag grab”，“hand open”，“hand closed”，“hand pointer”，“zoom in”，“dilation”，“translation”，“show”。

API 级别限制：最低为 platform.apiLevel = ‘1.0’

6.2 hide

```
cursor.hide()
```

此例程使手持设备上的鼠标指针隐藏。

注意：在非手持设备上调用此函数将被忽略。

API 级别限制：最低为 platform.apiLevel = ‘1.0’

6.3 show

```
cursor.show()
```

此例程使手持设备上的鼠标指针可见。

注意：在非手持设备上调用此函数将被忽略。

API 级别限制：最低为 platform.apiLevel = '1.0'

第七章

文档库（Document Library）

7.1 markChanged

```
document.markChanged()
```

这个例程会标记当前文档已被修改，在关闭文档之前用户将被提示保存 TI-Nspire™文件。

API 级别限制：最低为 platform.apiLevel = ‘1.0’

第八章

事件处理（Event Handling）

脚本语言应用程序通过事件处理程序（event handler）来对外部事件做出反应。所有的事件处理程序都集中在“on”模块。

样例
例如，当需要重绘窗口时， on.paint(gc) 会被调用。on.paint 被传递一个图文环境(graphics context)并利用它在窗口（window）上绘图。

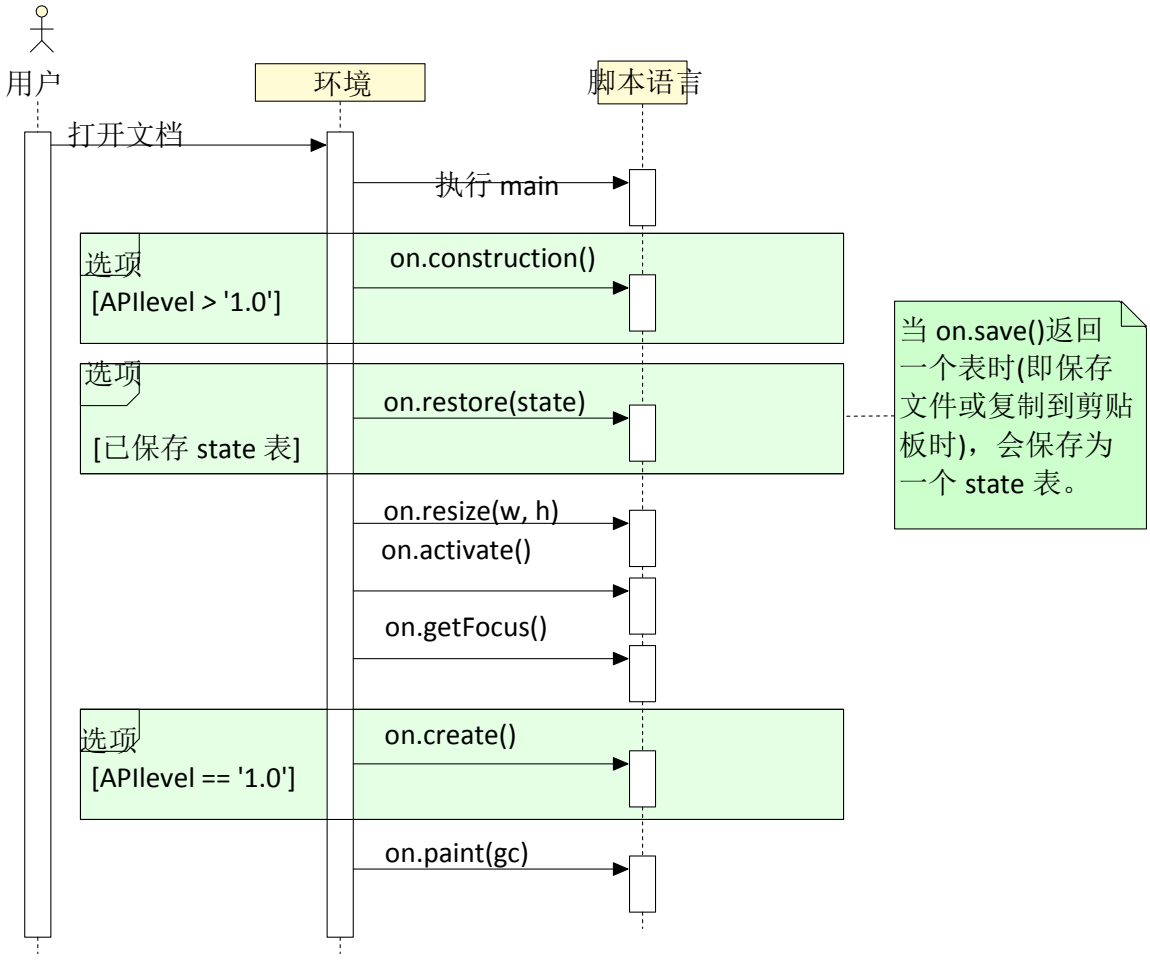
```
function on.paint(gc)
gc:drawLine(...)
:
end
```

简化的文档打开情景

在细节上有很多情景(scenario)可以讨论。所有特殊事件的行为在事件的描述部分中讨论了全部细节。
然而，这里讨论的是打开文档时的情况，以便更好地体现出各选项以及事件的触发顺序。事实上在不同的环境下可能有更多的事件。

根据脚本语言的 API 级别，事件处理程序 **on.construction** 和 **on.create** 中的一个会被调用。**on.construction** 的好处首先是在初始化应用时分离出类变量的定义（在 mian 中完成）；其次从初始化中分离出了布局(layout)（在 **on.resize()**中完成）。**on.create()**的主要问题是失去了无效屏幕(invalidate，参见 14.8.2 小节)的功能以及混合了布局(layout)与初始化。稍后可能在某些情况下失去无效屏幕的能力。另一个选项是在脚本语言已经保存并提供了 state 表(参见 8.40 小节)时调用 **on.restore()**。下述的图表直观的体现了这一点。

此外，一个脚本语言在收到事件 **on.resize()**前可能没有一个尺寸(size)，理解这一点是很重要的。在事件 **on.resize()**前获取平台的窗口(platform window)宽度或者高度(参见 14.8.1 小节)可能会返回 0。


用户
环境
脚本语言
打开文档
执行 main
选项
[APIlevel > '1.0']
on.construction()
选项
[已保存 state 表]
on.restore(state)
当 on.save()返回一个表时(即保存文件或复制到剪贴板时)，会保存为一个 state 表。
on.resize(w, h)
on.activate()
on.getFocus()
选项
[APIlevel == '1.0']
on.create()
on.paint(gc)

8.1 activate

```
on.activate()
```

当脚本语言应用程序被激活时调用此事件处理程序。窗口大小在此处不能够被初始化。因此，如果有依赖窗口大小的图形元件，那么此处不适合对它们进行创建和放置。

API 级别限制：最低为 platform.apiLevel = '1.0'

8.2 arrowDown

```
on.arrowDown()
```

当用户按下向下方向键(down arrow key)时调用此事件处理程序。

API 级别限制：最低为 platform.apiLevel = '1.0'

8.3 arrowKey

```
on.arrowKey(key)
```

当用户按下方向键(arrow key)时调用此事件处理程序。参数 **key** 可以是"up"， "down"，"left"，或者"right"。 如果脚本语言为特定的方向键实现了一个处理程序(例如 on.arrowDown) ，那么此事件处理程序将不会被调用。

API 级别限制：最低为 platform.apiLevel = '1.0'

8.4 arrowLeft

```
on.arrowLeft()
```

当用户按下向左方向键(left arrow key)时调用此事件处理程序。

API 级别限制：最低为 platform.apiLevel = '1.0'

8.5 arrowRight

```
on.arrowRight()
```

当用户按下向右方向键(right arrow key)时调用此事件处理程序。

API 级别限制：最低为 platform.apiLevel = '1.0'

8.6 arrowUp

```
on.arrowUp()
```

当用户按下向上方向键(up arrow key)时调用此事件处理程序。

API 级别限制：最低为 platform.apiLevel = '1.0'

8.7 charIn

```
on.charIn(char)
```

当用户输入一个字母、数字或其他字符时调用此事件处理程序。参数 char 通常是 1 个字节的字符串。但是由于它可以包含一个 UTF-8 编码的字符，它就有可能是 2 个或更多字节的字符串。同时，它也有可能包含从某个快捷键得到的一个函数名称的字母，如从 trig 菜单得到的"sin"。

API 级别限制：最低为 platform.apiLevel = '1.0'

8.8 backspaceKey

```
on.backspaceKey()
```

当用户按下桌面平台键盘上的 Backspace 键时或者按下手持设备上的 Del 键时调用此事件处理程序。

API 级别限制：最低为 platform.apiLevel = '1.0'

8.9 backtabKey

```
on.backtabKey()
```

当用户按下 Shift + Tab 时调用此事件处理程序。

API 级别限制：最低为 platform.apiLevel = '1.0'

8.10 clearKey

```
on.clearKey()
```

当用户按下手持式键盘上的 Clear 键时调用此事件处理程序。

API 级别限制：最低为 platform.apiLevel = '1.0'

8.11 construction

```
on.construction()
```

此函数保证先于任何其它事件调用。

API 级别限制：最低为 platform.apiLevel = '2.0'

8.12 contextMenu

```
on.contextMenu()
```

当用户按下 Ctrl + Menu(context Menu key)时调用此事件处理程序。

API 级别限制：最低为 platform.apiLevel = '1.0'

8.13 copy

```
on.copy()
```

当用户从菜单中选择复制(Copy)命令或按下 Ctrl + C 时调用此事件处理程序。

注意
复制功能可通过 toolpalette.enableCopy(enable)开启/关闭。

API 级别限制：最低为 platform.apiLevel = '1.0'

8.14 create

```
on.create()
```

对于 platform.apiLevel ≥ '2.0'的脚本语言，使用 on.construction()代替。

当脚本语言应用被创建以及窗口调整尺寸之后调用此事件处理程序。窗口尺寸和图文环境(window size and graphics context)此时是可用的。此事件处理程序结束后将会调用 on.paint 事件处理程序。

最好将此函数视为一个自动触发一次的初始化方法。

API 级别限制：
仅 platform.apiLevel = '1.0'中可用
在 platform.apiLevel = '2.0'中被移除

8.15 createMathBox

```
on.createMathBox()
```

当用户按下 Ctrl + M 或插入一个数学框(表达式框，Expression Box)时调用此事件处理程序。如果合适，此事件处理程序的实现将可在相应的富文本框插入一个数学框。

API 级别限制：最低为 platform.apiLevel = '2.0'

8.16 cut

```
on.cut()
```

当用户从菜单中选择剪切(Cut)命令或按下 Ctrl + X 时调用此事件处理程序。

注意
剪切功能可通过 toolpalette.enableCut(enable)开启/关闭。

API 级别限制：最低为 platform.apiLevel = ‘1.0’

8.17 deactivate

```
on.deactivate()
```

当脚本停用(deactivate)时调用此事件处理程。这可能在用户将焦点移动到另一个页面或移动到同一页上的其他应用程序时发生。

API 级别限制：最低为 platform.apiLevel = ‘1.0’

8.18 deleteKey

```
on.deleteKey()
```

当用户按下桌面键盘上的 Delete 键时调用此事件处理程序。它不是手持设备键盘上的 Del 键。

API 级别限制：最低为 platform.apiLevel = ‘1.0’

8.19 destroy

```
on.destroy()
```

在脚本语言应用被删除之前时调用此事件处理程序。当一个脚本语言应用被剪切到剪贴板，并且包含它的文件被关闭时，该脚本语言应用将被删除。

API 级别限制：最低为 platform.apiLevel = ‘1.0’

8.20 enterKey

```
on.enterKey()
```

当用户按下 Enter 键时调用此事件处理程序。

API 级别限制：最低为 platform.apiLevel = '1.0'

8.21 escapeKey

```
on.escapeKey()
```

当用户按下 Esc 键时调用此事件处理程序。

API 级别限制：最低为 platform.apiLevel = '1.0'

8.22 getFocus

```
on.getFocus()
```

当脚本语言收到用户输入焦点时调用此事件处理程序。

API 级别限制：最低为 platform.apiLevel = '2.0'

8.23 getSymbolList

```
on.getSymbolList()
```

当脚本语言应用符号列表(symbol list)被序列化(serialize)到剪贴板时调用此事件处理程序。该脚本语言返回需要拷贝到剪切板的符号表(symbol table)中变量名称的一个列表。TI-Nspire™软件在复制脚本语言应用程序的同时也复制变量名称和值。之后，当用户粘贴脚本语言应用程序到另一个问题中时，系统会将相应的变量添加到该问题的符号表中。

需要注意的是，on.getSymbolList()仅在当一个包含脚本语言应用的页面被复制时，而不是在当一个包含脚本语言应用的问题被复制时被调用。这是因为当问题被复制时，整个符号表也将被复制。

例如，下面的函数表明变量 **f1** 需要与应用程序一同被复制到剪贴板中。当应用程序粘贴到其他的

问题甚至其他的文档 TNS 中时，**f1** 也会被添加到符号表中。

```
function on.getSymbolList()
return {"f1"}
end
```

API 级别限制：最低为 platform.apiLevel = '2.0'

8.24 grabDown

```
on.grabDown(x, y)
```

发生以下情况时调用此事件处理程序：

- 当用户按下并持续按住手持设备上的抓移(Select)键
- 当用户在手持设备上按下 Ctrl + 抓移(Select)
- 当用户在桌面软件中的活动页面(active card)上按下鼠标中键，此时参数 x 和 y 始终为 0

grabDown 和 grabUp 事件在所有情况下阻止 mouseUp 事件的发生。当长按设备上的抓移(Select)键时，他们将先于 mouseDown 事件发生。

API 级别限制：最低为 platform.apiLevel = '1.0'

8.25 grabUp

```
on.grabUp(x, y)
```

当抓取(grab)生效时，释放鼠标按键将调用该事件处理程序。参数 x 和 y 始终为 0。

API 级别限制：最低为 platform.apiLevel = '1.0'

8.26 help

```
on.help()
```

当用户按下帮助键(Help key)时调用此事件处理程序。在桌面软件上，帮助键是 Ctrl + Shift + ?， 在手持设备上是 Ctrl + ?(在 Trig 按钮上) 。

API 级别限制：最低为 platform.apiLevel = ‘1.0’

8.27 keyboardDown

```
on.keyboardDown()
```

仅在支持触摸的平台上调用此事件处理程序。它的调用表示任何停靠的键盘(docked keyboard)被用户隐藏或者被脚本语言通过调用 **touch.keyboardShow(false)**隐藏。

API 级别限制：最低为 platform.apiLevel = ‘2.2’

8.28 keyboardUp

```
on.keyboardUp(keyboardOverlapHeight)
```

仅在支持触摸的平台上调用此事件处理程序。它的调用表示一个停靠的键盘(docked keyboard)已经在屏幕上显示而且可能与脚本语言内容重叠。如果发生了重叠那么参数 **keyboardOverlapHeight** 将提供高度。这个程序的返回值控制用户能否通过拖移(pan)手势进行展开(scroll)。如果返回 **true** 那么允许用户展开，否则(**false**)将由脚本语言在 **on.paint()**中依次绘制内容来实现展开。默认返回值为 **true**。

API 级别限制：最低为 platform.apiLevel = ‘2.2’

8.29 loseFocus

```
on.loseFocus()
```

当脚本语言失去用户输入焦点时调用此事件处理程序。

API 级别限制：最低为 platform.apiLevel = ‘2.0’

8.30 mouseDown

```
on.mouseDown(x, y)
```

用户点击鼠标时调用此事件处理程序。参数 **x** 和 **y** 是相对于窗口的坐标，以像素为单位。

注意
如果鼠标右键被持续按下，该事件将**不会**发生。

API 级别限制：最低为 platform.apiLevel = '1.0'

8.31 mouseMove

```
on.mouseMove(x, y)
```

当用户移动鼠标时调用此事件处理程序。鼠标按钮不需要按下来触发此事件。

API 级别限制：最低为 platform.apiLevel = '1.0'

8.32 mouseUp

```
on.mouseUp(x, y)
```

当用户释放鼠标按钮时调用此事件处理程序。

注意
该事件在下列情况下**不会**发生：
- 鼠标右键已经被按下，导致先前的 mouseDown 被阻塞。
- 先前的 mouseDown 事件没有被处理

API 级别限制：最低为 platform.apiLevel = '1.0'

8.33 paint

```
on.paint(gc)
```

当脚本语言应用窗口需要绘制时调用此事件处理程序。参数 gc (graphics context，图文环境) 被脚本语言代码用于在窗口中绘图。

API 级别限制：最低为 platform.apiLevel = '1.0'

8.34 paste

```
on.paste()
```

当用户从菜单中选择粘贴(Paste)命令或按下 Ctrl + V 时调用此事件处理程序。

注意
粘贴功能可通过 toolpalette.enablePaste(enable)开启/关闭。

API 级别限制：最低为 platform.apiLevel = '1.0'

8.35 resize

```
on.resize(width, height)
```

当脚本语言应用的窗口大小改变时会调用此事件处理程序。这里适合初始化（或重新初始化）基于窗体大小的图形对象(graphical objects)。

API 级别限制：最低为 platform.apiLevel = '1.0'

8.36 restore

```
on.restore(state)
```

当脚本语言应用从一个已经保存过状态(state)的文档中恢复，或者应用被粘贴到一个文档中时，会调用此事件处理程序。仅在之前应用被复制到剪贴板(Clipboard)或保存到一个文档时将状态和应用一同进行了保存，该事件处理程序才会被调用。参见 on.save 事件处理程序。

参数 **state** 是事件处理程序 on.save 返回的表。

警告
在脚本语言初始化过程中无法使用的函数在 on.restore 中同样不可用。不能被调用的函数是 math.eval 和 platform.isDeviceModeRendering。

API 级别限制：最低为 platform.apiLevel = '1.0'

8.37 returnKey

```
on.returnKey()
```

当用户按下手持设备上的回车键 (Return key)时调用该事件处理程序。

注意
回车键在手持设备上回车键为如图所示的按键。
在 iPad®上为长按 enter 键。
在桌面软件中不支持此功能。

API 级别限制：最低为 platform.apiLevel = '1.0'

8.38 rightMouseDown

```
on.rightMouseDown(x, y)
```

当用户点击鼠标右键时调用此事件处理程序。参数 **x** 和 **y** 是相对于窗口的坐标，以像素为单位。

注意
仅在桌面版本中可用。

鼠标事件是唯一的，即在鼠标左键按下时不会触发 rightMouseDown 事件，反之亦然。

API 级别限制：最低为 platform.apiLevel = '1.0'

8.39 rightMouseUp

```
on.rightMouseUp(x, y)
```

当用户释放鼠标右键时调用此事件处理程序。

注意
仅在桌面版本中可用。

该事件在下列情况下**不会**发生:
- 鼠标左键已经被按下而导致阻塞了先前的 rightMouseDown 事件。
- 先前的 rightMouseDown 事件没有被处理。

API 级别限制：最低为 platform.apiLevel = '1.0'

8.40 save

```
on.save()
```

当脚本语言应用被保存到文件或者复制到剪贴板(Clipboard)时调用此事件处理程序。本事件处理程序应当返回一个当事件 on.restore 处理程序调用时用于恢复适当数据的表 (table) 。

API 级别限制：最低为 platform.apiLevel = '1.0'

8.41 tabKey

```
on.tabKey()
```

当用户按下 Tab 键时调用此事件处理程序。

API 级别限制：最低为 platform.apiLevel = '1.0'

8.42 timer

```
on.timer()
```

如果脚本语言应用实现了 on.timer，那么每次计时器(timer)时钟周期到了时会调用此事件处理程序。

API 级别限制：最低为 platform.apiLevel = '1.0'

8.43 varChange

```
on.varChange(varlist)
```

当被监控的变量被另一个应用改变时调用此事件处理程序。参数 **varlist** 是所有被更改的变量名称列表。这个事件处理程序必须返回一个值表明它接受新赋予的值或者不接受更改。

有效的返回值及其含义：

返回值	简介	备注
0	成功	脚本语言应用接受更改
-1	因范围而不接受	新值不满足条件，因为它太小或者太大而超出了范围。
-2	因类型而不接受	新值不满足条件，因为其类型无法被脚本语言应用中使用。
-3	因存在性而不接受	另一个应用删除了变量，而本应用需要它。

API 级别限制：最低为 platform.apiLevel = '1.0'

第九章

图形库(Graphics Library)

图文环境(graphics context)是一个模块，它可以处理脚本语言应用绘图来输出到窗口，以及调用图形库的例程来绘制窗口。每次窗口要重新绘制的时候，会将一个图文环境传递给 **on.paint** 事件处理程序。

图文环境采用了一个基于像素的直角坐标系，这个坐标系的原点在绘制窗口的左上角。

9.1 clipRect

```
gc:clipRect(op[, x, [y,  [width, [height]]]])
```

设置剪辑矩形(clipping rectangle)以用于随后的图形操作。

参数 **op** 为这几个字符串中的一个：“set”，“reset”，“intersect”， 或者“null”。

操作	描述
reset	设置剪辑矩形包含整个窗口。剩余参数被忽略并省去。
set	设置剪辑矩形到 x, y 坐标处，大小为指定的宽度(width) 和高度(height)。未指定参数则默认为系统窗口位置和大小。
intersect	在 platform.apilevel = ‘2.0’中移除。
null	将剪辑矩形置空。所有随后的图形命令会被忽略。

通常“set”操作在画图前被调用，比如绘制一段文字。在绘制完最后一个需要剪辑的图形后调用“reset”操作是非常重要的，这样就不会残留下剪辑矩形进而产生副作用。

API 级别限制：最低为 platform.apiLevel = ‘1.0’

9.2 drawArc

```
gc:drawArc(x, y,  width, height,  startAngle,  arcAngle)
```

根据矩形左上角坐标(x,y)，以及以像素为单位的宽度(width)和高度(height)来绘制一个弧。参数 width 和 height 必须大于等于 0。这个弧从 startAngle 角度开始，到 endAngle 角度处结束。0 角度指向右方，90 度角指向上方。（均采用标准的数学规则。但值得特别指出的是 y 轴在此倒置）。

如果要绘制一个圆，宽度和高度必须相等，起始和结束角度也必须是 0 度和 360 度。如果宽度和高度不相等，此例程将会绘制出一个椭圆。

API 级别限制：最低为 platform.apiLevel = '1.0'

9.3 drawImage

```
gc:drawImage(imageHandle,  x, y)
```

在处(x, y)绘制一张图像(image)。所绘制的图像必须由在之前调用 image.new(...)生成。

API 级别限制：最低为 platform.apiLevel = '1.0'

9.4 drawLine

```
gc:drawLine(x1, y1,  x2,  y2)
```

在(x1,y1)和(x2,y2)两点之间绘制一条线。

API 级别限制：最低为 platform.apiLevel = '1.0'

9.5 drawPolyLine

```
gc:drawPolyLine({x1, y1,  x2,  y2,  ..., xn,  yn})
```

绘制出连接一系列点(x,y)的线。多边形绘制时并不是自动闭合的。为了得到闭合的多边形，第一个点的坐标需要在点的列表末尾再出现一次。

API 级别限制：最低为 platform.apiLevel = '1.0'

9.6 drawRect

```
gc:drawRect(x, y, width, height)
```

根据给定的宽度(width)和高度(height)以点(x, y)为矩形的左上角绘制一个矩形。参数 width 和 height 必须大于等于 0。

API 级别限制：最低为 platform.apiLevel = '1.0'

9.7 drawString

```
gc:drawString("text", x, y [,vertalignment])
```

在窗口中的像素坐标(x,y)处绘制字符串。对齐方式(vertical alignment，即参数 vertalignment)可以是"baseline"，"bottom"，"middle"， 或者 "top"。 这种对齐是通过对字符串中的字符包围一个矩形，以矩形的高度来进行对齐。如果未指定对齐方式那么默认为"none"。

返回所绘制文本末尾像素的 x 坐标。

API 级别限制：最低为 platform.apiLevel = '1.0'

9.8 fillArc

```
gc:fillArc(x, y, width, height,  startAngle,  endAngle)
```

用预设颜色填充一段弧。参数 width 和 height 必须大于等于 0。设置填充颜色请参见 setColorRGB。

API 级别限制：最低为 platform.apiLevel = '1.0'

9.9 fillPolygon

```
gc:fillPolygon({x1, y1,  x2,  y2,  ... xn,  yn})
```

用预设颜色填充一个多边形。点的列表限定了一个多边形。设置填充颜色请参见 setColorRGB。

API 级别限制：最低为 platform.apiLevel = '1.0'

9.10 fillRect

```
gc:fillRect(x,  y, width, height)
```

用预设颜色填充一个矩形。参数 width 和 height 必须大于等于 0。设置填充颜色请参见 setColorRGB。

API 级别限制：最低为 platform.apiLevel = '1.0'

9.11 getStringHeight

```
gc:getStringHeight("text")
```

返回文本的像素宽度。此时像素宽度根据先前调用的 setFont 设定的字体而定。

API 级别限制：最低为 platform.apiLevel = '1.0'

9.12 getStringWidth

```
gc:getStringWidth("text")
```

返回文本的像素高度。此时像素高度根据先前调用的 setFont 设定的字体而定。

API 级别限制：最低为 platform.apiLevel = '1.0'

9.13 setColorRGB

```
gc:setColorRGB(red, green,  blue)
gc:setColorRGB(0xRRGGBB) – 仅 platform.level = '2.0'
```

为随后的绘制和填充例程设定颜色。红、绿、蓝三种分量颜色的取值范围在 0 到 255 之间。例如黑色是 0,0,0，而白色是 255,255,255。此外，单独的一个数值也可以被当做参数，换算公式为 blue + 255 * (green + 255 * red)。

API 级别限制：
最低为 platform.apiLevel = '1.0'

在 platform.apiLevel = ‘2.0’中扩展

9.14 setFont

```
gc:setFont(family, style, size)
```

为绘制和测量文本设定字体。字体(family)可以是“sansserif” 或 “serif”。字形(style)可以是“r” (常规)，“b” (加粗)，“i” (倾斜)，或 “bi” (加粗并倾斜)。

在 TI-Nspire™ CX 和其他旧的手持设备中，字号(size)受到限制，只能从以下字号中选择一个： 7，9，10，11，12，或 24。但是任何在 Windows®或 Mac® OS X®中支持的字号都能够在桌面软件中使用。

返回生效的字体，字形和字号。

API 级别限制：最低为 platform.apiLevel = ‘1.0’

9.15 setPen

```
gc:setPen([thickness[, style]])
```

设定用于绘制直线和边框的线条样式。参数 thickness 可以是“thin”，“medium”，或“thick”。如果没有指定 thickness，默认为 “thin”。参数 style 可以是“smooth”，“dotted”，或 “dashed”。如果没有指定 style，默认为“smooth”。

API 级别限制：最低为 platform.apiLevel = ‘1.0’

第十章

图像库(Image Library)

image 对象是一个包含了图像的容器。一般地，小型 GUI 对象像按钮，箭头以及一些其他的装饰性的图形都是 image 对象。

10.1 new

```
img = image.new(str)
```

这个函数通过输入的字符串来返回一个新的 image 对象。这个字符串包含一个图像数据头(header)和其后的每个像素的二进制表示。

这个图像数据头含有 20 字节的数据。这些数据的编排如下图表格所示，所有的域均为小端字节整数(little endian integer)。

偏移量	宽度 (字节)	备注
0	4	位图的像素宽度
4	4	位图的像素高度
8	1	图像对齐方式(0)
9	1	Flags (0)
10	2	Pad (0)
12	4	每行像素所占的的字节数
16	2	每个像素的位数(16)
18	2	每个位的位面(1)

图像像素数据紧随在数据头之后。像素按行排列。每个像素是一个 16 位(bit)的小端字节整数，红，绿，蓝颜色分量分别对应 5 个位。

最高位决定该像素是否被绘制。如果是 0，表明该像素不被绘制。如果是 1，该像素由剩余的 15 位以 RGB 颜色绘制。

例如：0x8000 是黑色，0x801F 是蓝色，0x83E0 是绿色，0xFC00 是红色 0xFFFF 是白色。

API 级别限制：最低为 platform.apiLevel = '1.0'

10.2 copy

```
cimage = image:copy(width, height)
```

返回图像缩放到指定像素宽度(width)和高度(height)的副本。

宽度和高度默认为源图像的尺寸。

API 级别限制：最低为 platform.apiLevel = '1.0'

10.3 height

```
h = image:height()
```

返回图像的像素高度。

API 级别限制：最低为 platform.apiLevel = '1.0'

10.4 rotate

```
rimage  = image:rotate(angle)
```

返回图像逆时针旋转 **angle** 度后的副本。

API 级别限制：最低为 platform.apiLevel = '2.0'

10.5 width

```
w  = image:width()
```

返回图像的像素宽度。

API 级别限制：最低为 platform.apiLevel = '1.0'

第十一章

区域库(Locale Library)

11.1 name

```
locale.name()
```

返回当前所在区域的名称。区域名称是一组含两个字母的语言代码。在这之后可能会有一条下划线和一组两个字母的国家代码。

API 级别限制：最低为 platform.apiLevel = '1.0'

第十二章

数学库扩展(Math Library Extension)

除了标准 LUA 提供的数学库外，TI-Nspire™ 数学服务器(math server) 还提供了一个接口以允许访问 TI-Nspire™产品的高级数学特性。

注意
TI-Nspire™数学服务器使用了大量的 unicode 字符。例如，数学服务器使用 Unicode 字符 U+F02F，UTF-8 字符“\239\128\175” 来表示虚数 *i*，又比如小写字母”e”是用于科学计数法的特殊字符 。参见 http://en.wikipedia.org/wiki/UTF-8 了解如何在 unicode 字符与 UTF-8 字符之间转换。请参阅《TI-Nspire™参考指南》中 TI-Nspire™软件使用的 unicode 字符列表。

所有从 TI-Nspire™数学服务器返回结果都为全精度(full-precision)的表达式。为了限制返回结果的精度在显示位数范围之内，可通过 math.getEvalSettings()检索当前显示器位数，并且在显示由 TI-Nspire™数学的服务器返回的值前，采用适当的精度。

12.1 eval

```
math.eval(math_expression) — platform.apiLevel = '2.0'
math.eval(math_expression [,exact]) — platform.apiLevel = '1.0'
```

该函数发送一个表达式或命令给。TI-Nspire™数学服务器进行计算。输入的表达式必须是字符串，以便。TI-Nspire™数学服务器解释和计算。

第二个参数 **exact** (仅在 platform.apiLevel = ‘1.0’中使用)只有在计算机代数系统（Computer Algebra System，简称 CAS）中是有意义的。如果为 true，它指示数学服务器计算并返回精确的数值结果。**exact** 默认值是 false，这样数学服务器将试图计算近似结果。

从 platform.apiLevel = ‘2.0’开始，计算将使用当前文档的设置，除非所有计算均在近似模式下进行全精度计算。当前文档设置可以由 **math.setEvalSettings** 覆盖。

如果数学服务器成功计算了表达式，它以基本 Lua 数据类型返回结果。如果由于语法、简化或语义错误，数学服务器无法计算表达式， **eval** 将返回两个结果：nil 和对服务器有意义的错误号。（错误号记录在《TI-Nspire™参考指南》—错误代码和 math.eval 消息中）如果服务器计算出是符号结果，它不能由基本 Lua 类型表示，所以 **eval** 返回 nil 和字符串 “incompatible data 类型”。

样例
给定 x 的值计算 **f1** 的值，参数 x 必须先转化为字符串，然后任何内嵌的“e”必须由 Unicode 字符 U+F000 替换。

```
local mx  = tostring(x):gsub("e", string.uchar(0xF000))
local expr  = "f1(" .. mx  .. ")" return
math.eval(expr)
```

注意
由于 **math.eval** 总是在近似模式进行计算，像布尔逻辑（Boolean logic）和一些转换将抛出错误：

r,e = math.eval('1 and 2') 返回”自变量必须是布尔表达式或整数”

r,e = math.eval(”0@>Base10”) 返回 ”域错误”

math.evalStr 可以在这种情况下正常工作。

警告
在脚本语言初始化过程中不能使用 math.eval。

API 级别限制：最低为 platform.apiLevel = ‘1.0’
在 **platform.apilevel = ’2.0’**中移除了可选参数 **exact**，改为使用当前文档设置，近似模式以及全精度。

12.2 evalStr

```
math.evalStr(math_expression)
```

该函数发送一个表达式或命令给 TI-Nspire™数学服务器进行计算。输入的表达式必须是字符串，以便 TI-Nspire™数学服务器解释和计算。计算使用当前文档设置进行，设置可以由 **math.setEvalSettings** 重设。

注意：
所有的计算都在全精度下进行进行，无论文档如何设置或重设。

如果数学服务器计算表示式成功，返回一个字符串结果，如果服务器没有返回计算的结果，**evalStr** 函数不返回结果。如果数学服务器由于语法、简化或语义错误没有正确返回结果，**evalStr** 返回两个结果： nil 和对服务器有意义的错误代码。

样例

在精确模式下计算“10^19”返回“1. 19”。 仔细的观察结果可以发现字符串内容是“\049\046\239\128\128\49\57”。“\239\128\128” 是 Unicode 字符 U+F000，用于科学计数法的小号的大写字母“E”。

```
result, error = math.evalStr('10^19')
t, u,  v, w,  x, y, z = string.byte(result, 1,  7)
print (result, #result, t, u,  v, w,  x, y, z)

->1.?19  7 49 46 239 128 128 49 57
```

计算 “2-3” 返回 “-1”. 结果字符串被编码为 “\226\136\146\49”. “\226\136\146” 是 Unicode 字符 U+2212，是一个负号。

```
result, error = math.evalStr('2-3')
v, w,  x, y, z = string.byte(result, 1,  5)
print (result, #result, v, w,  x, y, z)

->?1.  5 226 136 146 49 46
```

API 级别限制：最低为 platform.apiLevel = ‘2.0’

12.3 getEvalSettings

```
math.getEvalSettings()
```

返回一个当前 **math.eval** 使用的文档设置的表(table)。这些设置与当前文档设置相同，除非调用了 **setEvalSettings**。

样例
这个例子阐释了由 **getEvalSettings** 返回的表的结构。

```
{
  {'Display Digits', 'Fixed12'},
  {'Angle Mode', 'Gradian'},
  {'Calculation Mode', 'Approximate'},
  {'Real or Complex Format', 'Polar'},
  {'Exponential Format', 'Engineering'},
  {'Vector Format', 'Cylindrical'},
  {'Base', 'Binary'},
  {'Unit System', 'Eng/US'}, }
}
```

API 级别限制：最低为 platform.apiLevel = '2.0'

12.4 setEvalSettings

```
math.setEvalSettings(settingStructure)
```

此函数用来为后续的 **math.eval** 和 **math.evalStr** 数学计算重设一个或多个当前文档设置。它不会改变文档当前设置。参数 settingStructure 是由一些子表构成的表(table)，每个内部的表包含要重设的文档设置名称和要使用的设置值。

样例
调用 math.setEvalSettings()的例子

```
settings = {
      {'Unit System', 'Eng/US'},
      {'Calculation Mode', 'Approximate'},
      {'Real or Complex Format', 'Polar'},
      {'Exponential Format', 'Engineering'}
}

math.setEvalSettings(settings)
```

为了方便用户， **setEvalSettings** 接受使用设置的序号和使用设置值的序号进行设置。使用的序号对应于设置的顺序和它们在 文件-> 设置 -> 文档设置 中的值的顺序。

样例
使用含有序号的表调用 math.setEvalSettings() 的例子

```
settingsTable = {
  {2, 3},
  {4, 3},
  {6, 3},
  {8, 2}
}

math.setEvalSettings(settingsTable)
```

事实上，**setEvalSettings** 接收任意名称和序号的组合。所以下面的样例也是合法的。

样例
使用混有序号和名称的表调用 math.setEvalSettings() 的例子

```
settings = {
  {3, 'Exact'},
  {'Angle  Mode',  2},
  {'Real  or  Complex Format',  'Polar'},
  {8, 2}
}

math.setEvalSettings(settings)
```

math.setEvalSettings 函数可以在脚本语言应用的任何位置调用。被修改的文档设置在脚本语言应用中所有随后的 **math.eval** 使用。（除非在随后的 **setEvalSettings** 调用中修改）。

注意
所有 TI-Nspire™数学服务器的结果均以全精度表达式返回。如果用户需要限制显示位数，必须调用 math.getEvalSettings()math.getEvalSettings()，并且在显示由 TI-Nspire™ 数学服务器返回的值前采用适当的精度。

API 级别限制：最低为 platform.apiLevel = '2.0'

第十三章

模块库(Module Library)

```
require '<library name>'
```

使用 **require** 来载入 TI-Nspire™ 软件中的预定义库，可用的库请见下表。

require 的行为与标准 Lua 中的相同，但可用的库有一定限制。不支持用户自定义的库。

库	描述
color	一个定义了颜色的表(table)，通过颜色拾取器(color picker)使用 TI-Nspire™ 软件中的颜色对象。
physics	载入 physics 模块。

表中定义的颜色：

black	darkgray	gray	mediumgray	lightgray	white
navy	blue	brown	red	magenta	orange
yellow	green	dodgerblue			

API 级别限制：最低为 platform.apiLevel = '2.0'

第十四章

平台库(Platform Library)

通过平台库可以得到平台的专用信息。

14.1 apiLevel

```
platform.apiLevel
```

这是脚本语言环境版本的唯一标识。如果一个脚本语言没有申请所需的 API 级别(API level)，那么默认将采用创建脚本语言时使用的 API 级别。

如果申请一个当前版本所不支持的 API 级别，那么将会导致 TI-Nspire™软件使用当前版本所支持的最高 API 级别。更多细节请参见 A.1 一节。

警告：在未来的软件发行版本中，申请一个不支持的 API 级别时的行为可能有所不同。可能会默认为低于所申请 API 级别的最高级别，而不是最高所支持的 API 级别。

注意

- 如果存在，那么 platform.apiLevel = 'X.X'声明只能放在脚本语言的主体部分，而不可以放在任何函数当中，建议放在脚本语言的第一行。
- 动态载入脚本语言(**load()** 或者 **loadstring()**)会使用与主脚本语言相同的"platform.apiLevel = 'X.X'" 。在动态载入的脚本语言中更改 API 级别会引发错误。

API 级别限制：最低为 platform.apiLevel = '2.0'

14.2 gc

```
platform.gc()
```

返回一个虚拟的图文环境(graphics context)。当正常的图文环境不可用时，它通常被用来测量字符串的像素宽度和高度。这种情况可能在脚本语言应用初始化过程中创建新的文本元素时出现。图文环境只在 paint 事件期间有效，此时再为文本创建和规定容器长度显然太晚了。

这个图文环境不可以被用于绘图，因为它并不一定与窗口关联。

以下是一个使用虚拟图文环境来得到字符串的像素宽度和高度的例子。

```
local gc = platform.gc()
gc:begin()
local width  = gc:getStringWidth(a_string)
local height = gc:getStringHeight(a_string)
gc:finish()
```

在使用图文环境之前，最重要的一点是在使用其中的函数 getString 前用 gc:begin()来启动它，使用完之后记得调用 gc:finish()来废除它。

API 级别限制：
仅 platform.apiLevel = '1.0'中可用
在 platform.apiLevel = '2.0'中被移除

14.3 hw

```
platform.hw()
```

返回一个表示主机 CPU 硬件速度的数值。数字越大，硬件速度越快。

级别	主机硬件
3	黑白屏幕的 TI-Nspire™ 以及 CX 手持设备
7	Microsoft® Windows ®，Mac ®或网络播放器

API 级别限制：最低为 platform.apiLevel = '2.0'

14.4 isColorDisplay

```
platform.isColorDisplay()
```

如果主机平台(host platform)是彩色显示时，返回 **true**。如果是灰度显示则返回 **false**。

API 级别限制：最低为 platform.apiLevel = '1.0'

14.5 isDeviceModeRendering

```
platform.isDeviceModeRendering()
```

当脚本语言在手持设备或桌面软件模拟器视图(emulator)中运行时返回 **true**，当其在桌面软件普通视图(normal view)下运行时返回 **false**。

注意
platform.isDeviceModeRendering 在脚本语言初始化期间或在 **on.restore** 事件中不可用。

API 级别限制：最低为 platform.apiLevel = '1.0'

14.6 isTabletModeRendering

```
platform.isTabletModeRendering()
```

如果脚本语言在支持触摸的平板电脑上运行时返回 **true**，否则返回 **false**。

API 级别限制：最低为 platform.apiLevel = '2.2'

14.7 registerErrorHandler

```
platform.registerErrorHandler(function(lineNumber, errorMessage,
                                       callStack, locals) ... end)
```

这一函数为脚本语言设置了错误处理函数。错误处理函数的设置可以为脚本语言遇到错误时提供控制。返回 **true** 可阻止向用户报告错误，脚本语言将在下一个事件中继续执行。

注意
在脚本语言初始化期间或在 **on.restore** 中发生错误时，不会调用错误处理函数。

API 级别限制：最低为 platform.apiLevel = '2.0'

14.8 window

```
platform.window
```

返回脚本语言应用当前所拥有的窗口对象 (window object)。窗口由分配给脚本语言应用的一部分页面构成。当页面使用分隔布局时，多个应用同时可见。每一个可见的应用拥有它自己的窗口。

窗口对象有几个十分有趣的用法。

API 级别限制：最低为 platform.apiLevel = '1.0'

14.8.1 height and width

```
platform.window:height()
platform.window:width()
```

例程 **height()** 和 **width()** 各自返回显示窗口的像素高度和宽度。

API 级别限制：最低为 platform.apiLevel = '1.0'

14.8.2 invalidate

```
platform.window:invalidate(x, y, width, height)
```

这个函数将窗口的一个区域设为无效并强制其重新绘制。参数 x 和 y 默认为(0, 0)，width 和 height 默认为窗口的像素宽度和高度。

通过调用 **platform.window:invalidate()** 可强制重新绘制整个窗口，这时允许所有参数取默认值。

API 级别限制：最低为 platform.apiLevel = '1.0'

14.8.3 setFocus

```
platform.window:setFocus(true or  false)
```

这一函数将所有焦点设置到主窗口。其他对象的焦点将全部移除(例如 D2Editor)。

API 级别限制：最低为 platform.apiLevel = '2.0'

14.8.4 getScrollHeight

```
platform.window:getScrollHeight()
```

如果有一个停靠的键盘(docked keyboard)已经显示，那么这个函数返回其当前的展开高度(scroll height)，否则返回 0。在不支持触摸的平台上总是返回 0。

关于键盘可见性的细节请参见 **touch.isKeyboardAvailable()**。

API 级别限制：最低为 platform.apiLevel = '2.2'

14.8.5 setScrollHeight

```
platform.window:setScrollHeight()
```

如果有一个停靠的键盘(docked keyboard)已经显示，那么这个函数设置其展开高度(scroll height)，否则予以忽略。有效值的范围是从 **0** 到**<键盘重叠时的高度>**。有关 键盘重叠时的高度(keyboard overlap height)请参见 **on.keyboardUp()**。

API 级别限制：最低为 platform.apiLevel = '2.2'

14.9 withGC

```
platform.withGC(function, args)
```

在一个虚拟的图文环境中执行 **function(args)**并且返回函数 **function()**的所有返回值。虚拟图文环境通常是用于在 paint 事件外测量字符串的像素宽度和高度。将布局(layout)从 paint 事件处理程序中分离是提高脚本语言性能的好方法。可能在 **on.resize()**事件中、用户交互更改数据时或者计时器过期时(timer expiration)，会更改布局。脚本语言不应该假设任何状态，例如字号，会从一次 **platform.withGC** 的调用保留至下一次调用 **platform.withGC**。

这个图文环境不可以被用于绘图。

下面是使用 withGC()来得到一个字符串像素宽度和高度的例子。

```
function getHeightWidth(str, gc)
   gc:setFont('serif', 'b', 12) — 设置字体
   width = gc:getStringWidth(str) — str 的像素宽度
    height = gc:getStringHeight(str) —   str 的像素高度
    return height, width
end

height, width = platform.withGC(getHeightWidth, 'Hello World')
```

API 级别限制：最低为 platform.apiLevel = ‘2.0’

第十五章

字符串库扩展(String Library Extension)

除了标准的 Lua 字符串函数，一些扩展例程用于协助处理 Unicode 字符串。

15.1 split

```
string.split(str  [,delim])
```

基于分隔符 delim 将字符串 str 分成一些子字符串，返回子字符串列表。分隔符默认为白色空格(”%s+”)。

API 级别限制：最低为 platform.apiLevel = ‘1.0’

15.2 uchar

```
string.uchar(chnum, ...)
```

Unicode 字符可以通过 UTF-8 编码而被包含在字符串中。这个例程将一个或多个 Unicode 字符代码转换为一个 UTF-8 字符串。

API 级别限制：最低为 platform.apiLevel = ‘1.0’

15.3 usub

```
string.usub(str, startpos, endpos)
or
str:usub(startpos, endpos)

print(string.usub("abc", 1,  1)) – 输出 "a"
print(string.usub("abc", 2,  2)) – 输出 "b"
print(string.usub("abc", 2,  3)) – 输出 "bc"
```

这个例程返回字符串 str 的一个子字符串。它是 Unicode 版本的 string.sub。它将多字节字符编码成 UTF-8。

警告
这一例程的代价很高。因为在其操作期间，会为其分配一块临时内存作为缓冲区。

API 级别限制：最低为 platform.apiLevel = '1.0'

第十六章

计时器库(Timer Library)

每个脚本语言应用都有一个计时器(timer)。计时器的频率取决于其应用的平台。在手持设备上一般约为 0.02 秒。请在手持设备上谨慎使用较短的时间周期。

脚本语言应用应当实现 **on.timer()**函数来响应计时器过期(timer expiration)。

即便窗口在屏幕上不可见，计时器仍会继续发送计时器终止消息至脚本语言应用。

包含脚本语言应用的文件关闭或脚本语言应用从文件中删除时，计时器会自动停止。

16.1 getMilliSecCounter

```
timer.getMilliSecCounter()
```

返回内部毫秒计数的值。计数器(counter)在计到 2^{32} 后重置为零。

API 级别限制：最低为 platform.apiLevel = '1.0'

16.2 start

```
timer.start(period)
```

以参数 period 为给定的时间周期启动计时器（单位：秒）。这一时间频率必须≥ 0.01 (10 ms)。如果调用这个程序时计时器已经在运行，则计时器会被重置为新的时间周期。

API 级别限制：最低为 platform.apiLevel = '1.0'

注意
timer.start()不可在处理 on.timer()事件期间调用。除非它是 on.timer()事件结束前的最后一条语句。

16.3 stop

```
timer.stop()
```

停止计时器。

API 级别限制：最低为 platform.apiLevel = '1.0'

第十七章

菜单库 (Tool Palette Library)

菜单(tool palette)提供了一种让用户在脚本语言应用中可以选择命令并且调用对应的功能的界面。

17.1 register

```
toolpalette.register(menuStructure)
```

脚本应用调用这个函数来在 TI-Nspire™框架(framework)中注册一个菜单。而菜单数据结构是一个表(table)，其中描述了每一个菜单(tool box，一级菜单)的名称，以及当用户选择了其中的某个子项(item)时应该调用的函数。

toolpalette.register()函数可以在脚本语言的开始部分就被调用一次。一旦被注册，菜单将被 TI-Nspire™框架自动接管，最多只能有 15 个一级菜单，每个一级菜单只能有 30 个子项。

当用户选择了某个一级菜单的某个子项时，会给相关联的函数传入 2 个参数：一级菜单名称和子项名称。

在 paint 事件中调用 **toolpalette.register()** 可能会被忽略并因此而无效。

API 级别限制：
最低为 platform.apiLevel = '1.0'
在 platform.apiLevel = '2.0'中扩展

从 apiLevel '2.0'开始， **toolpalette.register()** 可以在程序中被多次调用以便在运行时动态更改。

通过调用 **toolpalette.register(nil)**来注一个销菜单。

下面的样例展示了菜单表的数据结构以及其显示内容。

```
menu = {
  {"Mode", — 菜单 "Mode"
          {"Decimal", setDec}, — 子项 "Decimal" 会调用 setDec()
          {"Hexadecimal", setHex},
          "-", — 分割线
          {"Signed", setSigned},
          {"Unsigned", setUnsigned},
  },
  {"Boolean",
           {"And", binopAnd},
           {"Or", binopOr},
  },
} toolpalette.register(menu)
```

17.2 enable

```
toolpalette.enable(toolname, itemname, enable)
```

这个例程可以开启/关闭某个子项(item)。参数 **toolname** 是一个包含一级菜单名称(tool box)的字符串。参数 **itemname** 是一个包含子项名称的字符串。参数 **enable** 是一个布尔值(Boolean value)，如果为 true 那么开启一个子项，如果为 false 那么关闭一个子项。

如果成功开启/关闭了一个子项那么这个例程会返回 true。如果没有在已经注册的菜单中找到指定的子项，会返回一个 nil 。

注意
必须在使用 **toolpalette.enable()**前调用 **toolpalette.register()**。

API 级别限制：最低为 platform.apiLevel = '1.0'

17.3 enableCut

```
toolpalette.enableCut(enable)
```

这一例程启用或禁用 编辑->剪切 菜单命令。参数 **enable** 是一个布尔值，其值为 true 时启用命令，为 false 时禁用。

API 级别限制：最低为 platform.apiLevel = '1.0'

17.4 enableCopy

```
toolpalette.enableCopy(enable)
```

这一例程启用或禁用 编辑->复制 菜单命令。参数 **enable** 是一个布尔值，其值为 true 时启用命令，为 false 时禁用。

API 级别限制：最低为 platform.apiLevel = '1.0'

17.5 enablePaste

```
toolpalette.enablePaste(enable)
```

这一例程启用或禁用 编辑->粘贴 菜单命令。参数 **enable** 是一个布尔值，其为 true 时启用命令，为 false 时禁用。

API 级别限制：最低为 platform.apiLevel = '1.0'

第十八章

符号表库(Variable Library)

一个符号表(symbol table)被 TI-Nspire™数学引擎用来计算和存储变量。这个库使脚本语言得以访问存储在符号表中的变量。

并不是所有符号表中的变量都在 Lua 中有兼容的类型。但许多重要的变量类型都支持：实数，整数，字符串，数组和字符串，矩阵(在 Lua 中以列表的列表表示)以及布尔值 true 和 false。

18.1 list

```
var.list()
```

这个函数返回一个当前已在符号表中定义了的变量名称列表。

API 级别限制：最低为 platform.apiLevel = '1.0'

18.2 makeNumericList

```
var.makeNumericList(name)
```

使用给定参数 **name** 在符号表中创建一个数组(list)。这个数组被优化用来保存数值。例程 **storeAt** 和 **recallAt** 作用在用这个函数创建的列表上操作时效率会有很大的提升。

用法说明
这一函数不能被用于创建数字矩阵。下面的例程 var.recallAt 和 var.storeAt 可以处理矩阵，但矩阵必须是被其他方式创建而来。

```
var.store("mat", {{1,2},  {3,4}})  — 创建矩阵 mat
var.storeAt("mat", 13.3, 1,  1)
val  = var.recallAt("mat", 1,  1)
```

API 级别限制：最低为 platform.apiLevel = '2.0'

18.3 monitor

```
var.monitor(name)
```

开启对给定名称的数学变量 **name** 的监视。一旦其他应用改变了这一变量的值，这个脚本语言应用的 on.varChange 事件处理程序就会被调用。参见 on.varChange 的描述。除 0 以外的其他返回值均视为错误。

API 级别限制：最低为 platform.apiLevel = '1.0'

18.4 recall

```
var.recall(name)
```

返回指定名称的数学变量 **name** 的值。如果指定的变量类型没有兼容的 Lua 类型，返回 nil 和一个错误信息。

API 级别限制：最低为 platform.apiLevel = '1.0'

18.5 recallAt

```
var.recallAt(name, col  [,row])
```

返回符号表中一个数组或矩阵某个单元的值。**col** 是以 1 为单位的矩阵或数组列数。**row** 是以 1 为单位的行数。仅在返回矩阵中的某值时，需要参数 **row** 。

这一函数为操作数值而优化，一般返回一个数字。如果要被取回单元的值不是数字，函数返回 nil 和一个错误消息字符串。

API 级别限制：最低为 platform.apiLevel = '2.0'

18.6 recallStr

```
var.recallStr(name)
```

以字符串格式返回指定名称的数学变量 **name** 的值。一些数学类型不具备兼容的 Lua 类型，但所有的数学类型都可以用字符串来表示。若即使用字符串也无法返回，那么返回 nil 和一个错误信息。

API 级别限制：最低为 platform.apiLevel = ‘1.0’

18.7 store

```
var.store(name, value)
```

将参数 value 的值存储为指定名称的数学变量 **name**。如果值无法存储，返回一个错误信息。否则，返回 nil。

API 级别限制：最低为 platform.apiLevel = ‘1.0’

18.8 storeAt

```
var.storeAt(name, numericValue,  col  [, row])
```

将一个数值存入符号表中一个数组或矩阵 **name** 的某个单元。**col** 是以 1 为单位的矩阵或数组列数。**row** 是以 1 为单位的行数。仅在存储矩阵中的某值时，需要参数 **row**。

存储值必须为数字。任何其他类型都会引发错误。

通过存储到数组末尾后边的一列，新值可以被附加到数组中。这个函数作为一种优化非常有用，特别是在添加新值到列表中时。

成功则返 nil，否则，若这一值不能被添加到指定的数组或矩阵中时，返回“cannot store”。

API 级别限制：最低为 platform.apiLevel = ‘2.0’

18.9 unmonitor

```
var.unmonitor(name)
```

关闭对指定名称的数学变量的监视。

API 级别限制：最低为 platform.apiLevel = ‘1.0’

第十九章

物理库(Physics Library)

这是一个 Chipmunk Physics (版本 5.3.4)的接口库。有关这个库的细节可以参见 http://chipmunk-physics.net/release/Chipmunk-5.x/Chipmunk-5.3.4-Docs/。必须载入物理模块以使用这个库(使用“**require ('physics')**”语句)。为了这个库至少需要使用 platform.apiLevel = 2.0。

19.1 Miscellaneous routines

19.1.1 INFINITY

```
infinity = physics.misc.INFINITY()
```

参数	类型	描述
infinity	**输出** number	无穷

从物理引擎返回一个代表无穷的数字。.

API 级别限制：最低为 platform.apiLevel = ‘2.0’

19.1.2 momentForBox

```
inertia = physics.misc.momentForBox(mass,  width, height)
```

参数	类型	描述
mass	输入 number	方形物体的质量
width	输入 number	方形物体的宽
height	输入 number	方形物体的高
inertia	输出 number	方形物体的转动惯量

计算方形物体(solid box)的转动惯量。这对计算转动惯量并作为 physics.Body(...)构造函数(constructor)的输入参数是非常有帮助的。

API 级别限制：最低为 platform.apiLevel = ‘2.0’

19.1.3 momentForCircle

```
inertia = physics.misc.momentForCircle(mass,  innerRadius,  outerRadius, offBody)
```

参数	类型	描述
mass	**输入** number	圆的质量
innerRadius	**输入** number	圆的内径
outerRadius	**输入** number	圆的外径
offset	**输入** physics.Vect	圆距离重心的偏移量
inertia	**输出** number	圆的转动惯量

计算一个圆(circle)的转动惯量。一个圆形刚体的内径为 0。这对计算转动惯量并作为 physics.Body(...) 构造函数 (constructor)的输入参数是非常有帮助的。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.1.4 momentForPoly

```
inertia = physics.misc.momentForPoly(mass,  vertices, offset)
```

参数	类型	描述
mass	**输入** number	多边形的质量
vertices	**输入** *{physics.Vect}*	定义多边形外形的向量列表
offset	**输入** physics.Vect	多边形距离重心的偏移量
inertia	**输出** number	多边形的转动惯量

计算一个多边形 (polygon)的转动惯量。这对计算转动惯量并作为 physics.Body(...)构造函数 (constructor)的输入参数是非常有帮助的。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.1.5 momentForSegment

```
inertia = physics.misc.momentForSegment(mass,  endPointA, endPointB)
```

参数	类型	描述
mass	**输入** number	线段的质量
endPointA	**输入** physics.Vect	定义线段一端的点
endPointB	**输入** physics.Vect	定义线段另一端的点
inertia	**输出** number	线段的转动惯量

计算一条线段(segment)的转动惯量。这对计算转动惯量并作为 physics.Body(...)构造函数(constructor)的输入参数是非常有帮助的。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.2 Vectors

向量(vector)是一个二维的由 x 和 y 坐标组成的对象。它的类型是 TI.cpVect

19.2.1 Vect

```
vector = physics.Vect(x, y)
vector = physics.Vect(angle)
vector = physics.Vect(vect)
```

参数	类型	描述
x	**输入** number	向量的 x 分量
y	**输入** number	向量的 y 分量
angle	**输入** number	角的大小(弧度)
vect	**输入** physics.Vect	一个向量
vector	**输出** physics.Vect	一个向量

第一种形式创建了一个带有初始 x 和 y 分量的向量。第二种形式创建了一个方向是 angle 弧度的角的单位向量。第三种形式创建了一个输入向量的副本。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.2.2 add

```
sum  = physics.Vect:add(vec)
```

参数	类型	描述
self	**输入** physics.Vect	输入向量
vec	**输入** physics.Vect	要和 **self** 相加的向量
sum	**输出** physics.Vect	**self** 和 **vec** 相加得到的向量

返回 **self** 与 **vec** 的向量和。

Vect 类也重载了加法运算符(+)。因而向量 **v1** 与 **v2** 的和可以用表达式 **v1** + **v2** 得到。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.2.3 clamp

```
clamped = physics.Vect:clamp(len)
```

参数	类型	描述
self	**输入** physics.Vect	输入向量
len clamped	**输入** number	向量的最大长度
self	**输出** physics.Vect	一个长度不长于 **len** 的新向量

返回在长度 **len** 之内的 **self** 的副本。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.2.4 cross

```
crossprod  = physics.Vect:cross(vec)
```

参数	类型	描述
self	**输入** physics.Vect	输入向量
vec	**输入** physics.Vect	与 **self** 叉乘的向量
zmag	**输出** number	**self** 和 **vec** 矢量叉乘后的 z 坐标的值

返回 **self** 和 **vec** 矢量叉乘后的 z 坐标的值。

API 级别限制：最低为 platform.apiLevel = ‘2.0’

19.2.5 dist

```
dist = physics.Vect:dist(vec)
```

参数	类型	描述
self	**输入** physics.Vect	输入向量
vec	**输入** physics.Vect	需要得到与 **self** 之间距离的向量
dist	**输出** number	**self** 到 **vec** 的距离

返回 **self** 和 **vec** 之间的距离。

API 级别限制：最低为 platform.apiLevel = ‘2.0’

19.2.6 distsq

```
distsq = physics.Vect:distsq(vec)
```

参数	类型	描述
self	**输入** physics.Vect	输入向量
vec	**输入** physics.Vect	需要得到与 **self** 之间距离的平方的向量
distsq	**输出** number	**self** 到 **vec** 的距离的平方

返回 **self** 和 **vec** 的距离的平方。对于距离间的比较，使用这个程序要比 **physics.Vect:dist** 快。

API 级别限制：最低为 platform.apiLevel = ‘2.0’

19.2.7 dot

```
dotprod = physics.Vect:dot(vec)
```

参数	类型	描述
self	**输入** physics.Vect	输入向量
vec	**输入** physics.Vect	另一个向量
dotprod	**输出** number	**self** 和 **vec** 的点积

返回 **self** 和 **vec** 的点积。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.2.8 eql

```
isequ = physics.Vect:eql(vec)
```

参数	类型	描述
self	**输入** physics.Vect	输入向量
vec	**输入** physics.Vect	与 **self** 比较的向量
isequ	**输出** boolean	如果 **self** 的分量与 **vec** 的分量相等返回 true

如果 **self** 的 **x** 和 **y** 分量与 **vec** 的相等，返回 **true**。当比较浮点数是否相等时采取通常的预防措施以避免浮点数陷阱。

Vect 类也重载了相等运算符（==）。因此向量 **v1** 和 **v2** 可以用表达式 **v1** == **v2** 来比较。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.2.9 length

```
len = physics.Vect:length()
```

参数	类型	描述
self	**输入** physics.Vect	输入向量
len	**输出** number	向量 **self** 的长度

返回向量 **self** 的长度。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.2.10 lengthsq

```
lensq  = physics.Vect:lengthsq()
```

参数	类型	描述
self	**输入** physics.Vect	输入向量
lensq	**输出** number	**self** 长度的平方

返回 **self** 的长度的平方。当仅需要比较长度时，这个程序比 **Vect:length()**快。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.2.11 lerp

```
v = physics.Vect:lerp(vec, f)
```

参数	类型	描述
self	**输入** physics.Vect	输入向量
vec	**输入** physics.Vect	另一个向量
f	**输入** number	**f** 是一个在 0 到 1 之间的分数，代表 **self** 和 **vec** 之间距离的比例。
v	**输出** physics.Vect	**self** 和 **vec** 之间的一个线性插值向量

返回 **self** 和 **vec** 之间的线性插值向量。**f** 是 **self** 和 **vec** 之间距离的比例分数。

注意
当 **f** 大于 1.0 或小于 0 时，效果可能与预期不同。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.2.12 lerpconst

```
v = physics.Vect:lerpconst(vec, d)
```

参数	类型	描述
self	**输入** physics.Vect	输入向量
vec	**输入** physics.Vect	另一个向量
d	**输入** number	用于插入一个新向量的从 **self** 到 **vec** 的距离
v	**输出** physics.Vect	**self** 和 **vec** 之间的一个线性插值向量

返回一个 **self** 和 **vec** 之间长度为 **d** 的线性插值向量

注意
当 **d** 大于 1.0 大或小于 0 时，效果可能与预期不同。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.2.13 mult

```
v = physics.Vect:mult(factor)
```

参数	类型	描述
self	**输入** physics.Vect	输入向量
factor	**输入** number	与 **self** 相乘的因数
v	**输出** physics.Vect	缩放后的向量结果

将一个因数与一个向量相乘。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.2.14 near

```
isnear = physics.Vect:near(vec, distance)
```

参数	类型	描述
self	**输入** physics.Vect	输入向量
vec	**输入** physics.Vect	与 **self** 相乘的向量
distance	**输入** number	与 **vec** 的距离
isnear	**输出** boolean	如果 **self** 在给定的 **vec** 的距离之内返回 **true**

确定 **self** 是否靠近另一个向量 **vec**。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.2.15 neg

```
v = physics.Vect:neg()
```

参数	类型	描述
self	**输入** physics.Vect	输入向量
v	**输出** physics.Vect	向量取反后的结果

返回 **self** 的相反向量。

Vect 类也重载了取反运算符(-**self**)。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.2.16 normalize

```
normvec = physics.Vect:normalize()
```

参数	类型	描述
self	**输入** physics.Vect	输入向量
normvec	**输出** physics.Vect	向量标准化后的结果

返回 **self** 的一个标准化后的副本。标准化向量的长度是 1。

API 级别限制：最低为 platform.apiLevel = ‘2.0’

19.2.17 normalizeSafe

```
normvec = physics.Vect:normalizeSafe()
```

参数	类型	描述
self	**输入** physics.Vect	输入向量
normvec	**输出** physics.Vect	向量标准化后的结果

返回 **self** 的一个标准化后的副本。这个安全版本会防止除以 0 。

API 级别限制：最低为 platform.apiLevel = ‘2.0’

19.2.18 perp

```
perpvec  = physics.Vect:perp()
```

参数	类型	描述
self	**输入** physics.Vect	输入向量
perpvec	**输出** physics.Vect	输出与 **self** 垂直的一个向量

返回一个与 **self** 垂直的向量 (逆时针旋转 90 度) 。

API 级别限制：最低为 platform.apiLevel = ‘2.0’

19.2.19 project

```
pvec = physics.Vect:project(vec)
```

参数	类型	描述
self	**输入** physics.Vect	输入向量
vec	**输入** physics.Vect	另一个向量
pvec	**输出** physics.Vect	**self** 在 **vec** 上的投影

计算 **self** 在另一个向量 **vec** 上的投影。

API 级别限制：最低为 platform.apiLevel = ‘2.0’

19.2.20 rotate

```
rvec = physics.Vect:rotate(vec)
```

参数	类型	描述
self	**输入** physics.Vect	输入向量
vec	**输入** physics.Vect	另一个向量
pvec	**输出** physics.Vect	向量旋转后的结果

使用复数乘法旋转 **self** 来得到 **vec**。如果 **self** 不是单位向量，其大小就会发生变化。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.2.21 rperp

```
perpvec = physics.Vect:rperp()
```

参数	类型	描述
self	**输入** physics.Vect	输入向量
perpvec	**输出** physics.Vect	输出与 **self** 垂直的一个向量

返回一个与 **self** 垂直的向量 (顺时针旋转 90 度) 。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.2.22 setx

```
self = physics.Vect:setx(x)
```

参数	类型	描述
self	**输入** physics.Vect	需要更改的向量
x	**输入** number	向量 **x** 分量的新值
self	**输出** physics.Vect	将更改后的输入向量作为返回值

改变 **self** 的 **x** 分量，返回 **self** 。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.2.23 sety

```
self = physics.Vect:sety(y)
```

参数	类型	描述
self	**输入** physics.Vect	需要更改的向量
y	**输入** number	向量 **y** 分量的新值
self	**输出** physics.Vect	将更改后的输入向量作为返回值

改变 **self** 的 **y** 分量，返回 **self**。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.2.24 slerp

```
v = physics.Vect:slerp(vec, f)
```

参数	类型	描述
self	**输入** physics.Vect	单位向量
vec	**输入** physics.Vect	另一个单位向量
f	**输入** number	**f** 是一个在 0 到 1 之间的分数，代表 **self** 和 **vec** 之间距离的比例。
v	**输出** physics.Vect	**self** 和 **vec** 之间的一个球面线性插值向量

计算单位向量 **self** 和 **vec** 之间的一个球面线性插值向量。有关球面线性插值的含义，值以及用法的讨论参见 http://en.wikipedia.org/wiki/Slerp，

```
local vect1  = physics.Vect(math.pi/3)  — 单位向量
local vect2  = physics.Vect(math.pi/2)  — 单位向量
local result = vect1:slerp(vect2,  0.55)
```

注意
此例程计算得到的结果仅在两个输入向量都是单位向量时有意义。当 **f** 大于 1.0 或小于 0 时，效果可能与预期不同。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.2.25 slerpconst

```
v = physics.Vect:slerpconst(vec,  angle)
```

参数	类型	描述
self	**输入** physics.Vect	一个单位向量
vec	**输入** physics.Vect	另一个单位向量
angle	**输入** number	在 **self** 和 **vec** 之间插入的一个新向量的最大弧度
v	**输出** physics.Vect	**self** 和 **vec** 之间的一个球面线性插值向量

计算单位向量 **self** 和 **vec** 之间的一个不大于指定弧度 **angle** 的球面线性插值向量。

信息
有关球面线性插值的含义，值以及用法的讨论参见 http://en.wikipedia.org/wiki/Slerp

注意
此例程计算得到的结果仅在两个输入向量都是单位向量时有意义。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.2.26 sub

```
diff = physics.Vect:sub(vec)
```

参数	类型	描述
self	**输入** physics.Vect	输入向量
vec	**输入** physics.Vect	要从 **self** 减去的向量
diff	**输出** physics.Vect	**self** 和 **vec** 之间的向量差

返回向量 **self** 和 **vec** 之间的差。

Vect 类也重载了减法运算符 (-)。因此 计算 **v1** 与 **v2** 的向量差也可以用表达式 **v1** - **v2**。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.2.27 toangle

```
angle  = physics.Vect:toangle()
```

参数	类型	描述
self	**输入** physics.Vect	输入向量
angle	**输出** number	**self** 的方向角大小

以弧度为单位返回 **self** 的方向角大小。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.2.28 unrotate

```
uvec = physics.Vect:unrotate(vec)
```

参数	类型	描述
self	**输入** physics.Vect	输入向量
vec	**输入** physics.Vect	另一个向量
uvec	**输出** physics.Vect	输入向量旋转前的结果

physics.Vect:rotate(vec)的反函数。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.2.29 x

```
x = physics.Vect:x()
```

参数	类型	描述
self	**输入** physics.Vect	输入向量
x	**输出** number	向量 **x** 分量的值

返回输入向量的 **x** 分量的值。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.2.30 y

```
y = physics.Vect:y()
```

参数	类型	描述
self	**输入** physics.Vect	输入向量
y	**输出** number	向量 **y** 分量的值

返回输入向量的 **y** 分量的值。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.3 Bounding Boxes

边界框(bounding box)是包含左边(left)、底边(bottom)、右边(right)、上边(top)的一种结构。它的类型是 TI.cpBB 。

19.3.1 BB

```
bb = physics.BB(l, b,  r, t)
```

参数	类型	描述
l	**输入** number	左边的 x 坐标
b	**输入** number	底边的 y 坐标
r	**输入** number	右边的 x 坐标
t	**输入** number	上边的 y 坐标
bb	**输出** physics.BB	根据给定参数创建的边界框

返回带有给定初始边界的新边界框。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.3.2 b

```
bottom  = physics.BB:b()
```

参数	类型	描述
self	**输入** physics.BB	输入边界框
bottom	**输出** number	边界框的底边 y 坐标

返回边界框的底边的 y 坐标。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.3.3 clampVect

```
cvec = physics.BB:clampVect(vec)
```

参数	类型	描述
self	**输入** physics.BB	输入边界框
vec	**输入** physics.Vect	一个向量
cvec	**输出** physics. Vect	向量 **vec** 在边界框内的副本

返回向量 **vec** 在边界框内的副本。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.3.4 containsBB

```
bool  = physics.BB:containsBB(other)
```

参数	类型	描述
self	**输入** physics.BB	输入边界框
other	**输入** physics. BB	另一个边界框
bool	**输出** boolean	如果边界框 **self** 完全包含 **other** 为 true

确定一个边界框是否包含另一个边界框。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.3.5 containsVect

```
bool  = physics.BB:containsVect(vec)
```

参数	类型	描述
self	**输入** physics.BB	输入边界框
vec	**输入** physics. Vect	一个向量
bool	**输出** boolean	如果边界框 **self** 包含 **vec** 时为 true

确定一个边界框是否包含一个向量。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.3.6 expand

```
bb = phyics.BB:expand(vec)
```

参数	类型	描述
self	**输入** physics.BB	输入边界框
vec	**输入** physics. Vect	一个向量
bb	**输出** physics.BB	将 **self** 扩展后的边界框以便可以包含向量 **vec**

返回同时包含 **self 和 vec** 的边界框。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.3.7 intersects

```
bool  = physics.BB:intersects(other)
```

参数	类型	描述
self	**输入** physics.BB	输入边界框
other	**输入** physics. BB	另一个边界框
bool	**输出** Boolean	如果 self 与 other 相交时为 true

确定两个边界框是否相交。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.3.8 l

```
left = physics.BB:l()
```

参数	类型	描述
self	**输入** physics.BB	输入边界框
left	**输出** number	边界框左边的 x 坐标

返回边界框左边的 x 坐标。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.3.9 merge

```
bb = physics.BB:merge(other)
```

参数	类型	描述
self	**输入** physics.BB	输入边界框
other	**输入** physics. BB	另一个边界框
bb	**输出** physics.BB	同时包含 **self** 和 **other** 的边界框

返回同时包含 **self 和 other** 的边界框。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.3.10 setb

```
self = physics.BB:setb(bottom)
```

参数	类型	描述
self	**输入** physics.BB	输入边界框
bottom	**输入** number	边界框底边的 y 坐标的新值
self	**输出** physics.BB	更改后的边界框作为输出的返回值

设置边界框的底边的 y 坐标为 **value**。返回 **self**。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.3.11 r

```
right = physics.BB:r()
```

参数	类型	描述
self	**输入** physics.BB	输入边界框
right	**输出** number	边界框右边的 x 坐标

返回边界框右边的 x 坐标。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.3.12 setl

```
self = physics.BB:setl(left)
```

参数	类型	描述
self	**输入** physics.BB	输入边界框
left	**输入** number	边界框左边的 x 坐标的新值
self	**输出** physics.BB	更改后的边界框作为输出的返回值

设置边界框的左边的 x 坐标为 **value**。返回 **self**。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.3.13 setr

```
self = physics.BB:setr(right)
```

参数	类型	描述
self	**输入** physics.BB	输入边界框
right	**输入** number	边界框右边的 x 坐标的新值
self	**输出** physics.BB	更改后的边界框作为输出的返回值

设置边界框的右边的 x 坐标为 **value**。返回 **self**。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.3.14 sett

```
self = physics.BB:sett(top)
```

参数	类型	描述
self	**输入** physics.BB	输入边界框
top	**输入** number	边界框上边的 y 坐标的新值
self	**输出** physics.BB	更改后的边界框作为输出的返回值

设置边界框的上边的 y 坐标为 **value**。返回 **self**。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.3.15 t

```
top  = physics.BB:t()
```

参数	类型	描述
self	**输入** physics.BB	输入边界框
top	**输出** physics.BB	边界框的上边的 y 坐标

返回边界框的上边的 y 坐标。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.3.16 wrapVect

```
wvec = physics.BB:wrapVect(vec)
```

参数	类型	描述
self	**输入** physics.BB	输入边界框
vec	**输入** physics.Vect	一个向量
wvec	**输出** physics.Vect	边界框覆盖后的向量

返回向量 **vec** 在边界框覆盖(wrap)后的副本。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.4 Bodies

物体(Body)包含一个对象的物理性质（质量、位置、旋转、速度等）。在没有对它赋予一个（或多个）外形(shape)之前，它不具有任何外形。它的类型是 TI.cpBody 。

19.4.1 Body

```
body = physics.Body(mass, inertia)
```

参数	类型	描述
mass	**输入** number	物体的质量
inertia	**输入** number	物体的转动惯量
body	**输出** physics.Body	带有给定的质量和惯性的新的物体

返回一个具有给定质量和转动惯量的物体。

使用提供的函数(provided helper functions)来帮助计算转动惯量。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.4.2 activate

```
self = physics.Body:activate()
```

参数	类型	描述
self	**输入** physics.Body	输入物体
self	**输出** physics.Body	更改后的物体作为输出返回

激活一个休眠物体。

信息

此例程的解释可参见 http://chipmunk-physics.net/release/Chipmunk-5.x/Chipmunk-5.3.4-Docs/

API 级别限制：最低为 platform.apiLevel = '2.0'

19.4.3 angle

```
angle  = physics.Body:angle()
```

参数	类型	描述
self	**输入** physics.Body	输入物体
angle	**输出** number	物体的方向角，用弧度表示

返回物体的方向角，用弧度表示。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.4.4 angVel

```
avel  = physics.Body:angVel()
```

参数	类型	描述
self	**输入** physics.Body	输入物体
avel	**输出** number	物体的角速度（单位：弧度/单位时间）

返回物体的角速度（单位：弧度/单位时间）。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.4.5 applyForce

```
self = physics.Body:applyForce(forceVect,  rOffset)
```

参数	类型	描述
self	**输入** physics.Body	输入物体
forceVect	**输入** physics.Vect	一个力向量
rOffset	**输入** physics.Vect	力向量相对于物体重心的偏移量
self	**输出** physics.Body	更改后的物体作为输出返回

在距物体 **self** 重心的一个相对偏移 **rOffset** 处施加一个力。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.4.6 applyImpulse

```
self = physics.Body:applyImpulse(impulseVect, rOffset)
```

参数	类型	描述
self	**输入** physics.Body	输入物体
impulseVect	**输入** physics.Vect	要加在物体上的冲量向量
rOffset	**输入** physics.Vect	冲量向量相对于物体重心的偏移
self	**输出** physics.Body	更改后的物体作为输出返回

在距物体 **self** 重心的一个相对偏移 **rOffset** 处施加一个冲量。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.4.7 data

```
obj  = physics.Body:data()
```

参数	类型	描述
self	**输入** physics.Body	输入物体
obj	**输出** Lua 对象	编程者已在物体上存储的 Lua 对象

返回编程者已经在物体数据域中存储的 Lua 对象。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.4.8 force

```
fvec  = physics.Body:force()
```

参数	类型	描述
self	**输入** physics.Body	输入物体
fvec	**输出** physics.Vect	物体上的力向量

返回物体上的力向量。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.4.9 isRogue

```
bool = physics.Body:isRogue()
```

参数	类型	描述
self	**输入** physics.Body	输入物体
bool	**输出** boolean	如果是 rogue 物体则为 true

如果是 rogue 物体则返回 true，它从不被加入到仿真空间(simulation Space)中。

信息
rogue 物体的解释可参见 http://chipmunk-physics.net/release/Chipmunk-5.x/Chipmunk-5.3.4-Docs/

API 级别限制：最低为 platform.apiLevel = '2.0'

19.4.10 isSleeping

```
bool = physics.Body:isSleeping()
```

参数	类型	描述
self	**输入** physics.Body	输入物体
bool	**输出** boolean	如果是休眠物体则为 true

如果物体是休眠物体返回 true。

信息
休眠物体的解释可参见 http://chipmunk-physics.net/release/Chipmunk-5.x/Chipmunk-5.3.4-Docs/

API 级别限制：最低为 platform.apiLevel = '2.0'

19.4.11 local2World

```
wvec = physics.Body:local2World(lvec)
```

参数	类型	描述
self	**输入** physics.Body	输入物体
lvec	**输入** physics.Vect	相对于物体位置的向量
vec	**输出** physics.Vect	世界坐标下的向量

将 **lvec** 从相对于物体的坐标转化为世界坐标(world coordinates)。返回转换后的向量。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.4.12 kineticEnergy

```
ke = physics.Body:kineticEnergy()
```

参数	类型	描述
self	**输入** physics.Body	输入物体
ke	**输出** number	物体的动能

返回物体的动能。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.4.13 mass

```
m  = physics.Body:mass()
```

参数	类型	描述
self	**输入** physics.Body	输入物体
m	**输出** number	物体的质量

返回物体的质量。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.4.14 moment

```
m  = physics.Body:moment()
```

参数	类型	描述
self	**输入** physics.Body	输入物体
m	**输出** number	物体的转动惯量

返回物体的转动惯量。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.4.15 pos

```
p = physics.Body:pos()
```

参数	类型	描述
self	**输入** physics.Body	输入物体
p	**输出** physics.Vect	物体的位置

返回物体的位置向量。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.4.16 resetForces

```
self = physics.Body:resetForces()
```

参数	类型	描述
self	**输入** physics.Body	输入物体
self	**输出** physics.Body	更改后的物体作为输出返回

将 **self** 上累积的力和转矩清零。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.4.17 rot

```
rvec  = physics.Body:rot()
```

参数	类型	描述
self	**输入** physics.Body	输入物体
rvec	**输出** physics.Vect	与物体同一方向的单位向量

返回物体的一个方向向量。这是根据计算最终的物体方向角而得到的一个单位向量。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.4.18 setAngle

```
self = physics.Body:setAngle(angle)
```

参数	类型	描述
self	**输入** physics.Body	输入物体
angle	**输入** number	物体的旋转角(单位：弧度)
self	**输出** physics.Body	更改后的物体作为输出返回

更新物体的旋转角(单位：弧度)。

返回更改后的物体。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.4.19 setAngVel

```
self = physics.Body:setAngVel(vel)
```

参数	类型	描述
self	**输入** physics.Body	输入物体
vel	**输入** number	物体的角速度（单位：弧度/单位时间）
self	**输出** physics.Body	更改后的物体作为输出返回

更新物体的角速度（单位：弧度/单位时间）。

返回更改后的物体。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.4.20 setData

```
self = physics.Body:setData(value)
```

参数	类型	描述
self	**输入** physics.Body	输入物体
value	**输入** Lua 对象	编程者添加的 Lua 对象
self	**输出** physics.Body	更改后的物体作为输出返回

更新物体的数据域。可以在此区域存储任意 Lua 对象。用这个位置存储仿真对象的引用是很方便的。

返回更改后的物体。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.4.21 setForce

```
self = physics.Body:setForce(vector)
```

参数	类型	描述
self	**输入** physics.Body	输入物体
vector	**输入** physics.Vect	物体的力向量
self	**输出** physics.Body	更改后的物体作为输出返回

更新物体上的力。

返回更改后的物体。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.4.22 setMass

```
self = physics.Body:setMass(mass)
```

参数	类型	描述
self	**输入** physics.Body	输入物体
mass	**输入** number	物体的质量
self	**输出** physics.Body	更改后的物体作为输出返回

更新物体的质量。

返回更改后的物体。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.4.23 setMoment

```
self = physics.Body:setMoment(moment)
```

参数	类型	描述
self	**输入** physics.Body	输入物体
moment	**输入** number	物体的转动惯量
self	**输出** physics.Body	更改后的物体作为输出返回

更新物体的转动惯量。

使用提供的函数(provided helper functions)来帮助计算转动惯量。

返回更改后的物体。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.4.24 setPos

```
self = physics.Body:setPos(vector)
```

参数	类型	描述
self	**输入** physics.Body	输入物体
vector	**输入** physics.Vect	物体的位置
self	**输出** physics.Body	更改后的物体作为输出返回

更新物体的位置。

返回更改后的物体。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.4.25 setPositionFunc

```
self = physics.Body:setPositionFunc(func)
```

参数	类型	描述
self	**输入** physics.Body	输入物体
func	**输入** function(body, dt)	在每个时间间隔内更新物体位置的函数
self	**输出** physics.Body	更改后的物体作为输出返回

设置一个物体位置函数(position function)。此位置函数必须是一个能够接收物体(body)和时间步长值(dt)的函数，并且此函数要在某些时刻调用 body:updatePosition 来对物体的位置进行更新。

返回更改后的物体。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.4.26 setTorque

```
self = physics.Body:setTorque(torque)
```

参数	类型	描述
self	**输入** physics.Body	输入物体
torque	**输入** number	物体的转矩
self	**输出** physics.Body	更改后的物体作为输出返回

更新物体的转矩。转矩是一个标量(numeric magnitude)。

返回更改后的物体。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.4.27 setVel

```
self = physics.Body:setVel(vector)
```

参数	类型	描述
self	**输入** physics.Body	输入物体
vector	**输入** physics.Vect	物体的矢量速度
wvec	**输出** physics.Body	更改后的物体作为输出返回

更新物体的速度。

返回更改后的物体。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.4.28 setVelocityFunc

```
self = physics.Body:setVelocityFunc(func)
```

参数	类型	描述
self	**输入** physics.Body	输入物体
func	**输入** function(body, grav, damping, dt)	在每个时间间隔内更新物体的速度的函数
wvec	**输出** physics.Body	更改后的物体作为输出返回

设置一个物体的速度函数(velocity function)。此度函数必须是一个能够接受物体(Body)，重力向量(grav)，阻尼系数(damping)和时间步长值(dt)的函数。此函数应该调用 body:updateVelocity 来调整物体的速度。

返回更改后的物体。

样例

```
function  sampleVelocityFunc(body, gravity, damping,  dt)
  local pos = body:pos()
  local sqdist = pos:lengthsq()
  local g = pos:mult(-GravityStrength / (sqdist *  math.sqrt(sqdist)))
  body:updateVelocity(g, damping,  dt)
end

body:setVelocityFunc(sampleVelocityFunc)
```

API 级别限制：最低为 platform.apiLevel = ‘2.0’

19.4.29 setVLimit

```
self = physics.Body:setVLimit(limit)
```

参数	类型	描述
self	**输入** physics.Body	输入物体
limit	**输入** number	物体的最大标量速度
self	**输出** physics.Body	更改后的物体作为输出返回

设置物体的最大速度限制。

返回更改后的物体。

API 级别限制：最低为 platform.apiLevel = ‘2.0’

19.4.30 setWLimit

```
self = physics.Body:setWLimit(limit)
```

参数	类型	描述
self	**输入** physics.Body	输入物体
limit	**输入** number	物体的最大角速度
self	**输出** physics.Body	更改后的物体作为输出返回

更新物体角速度的限制。角速度的单位是弧度/单位时间。

返回更改后的物体。

API 级别限制：最低为 platform.apiLevel = ‘2.0’

19.4.31 sleep

```
self = physics.Body:sleep()
```

参数	类型	描述
self	**输入** physics.Body	输入物体
self	**输出** physics.Body	更改后的物体作为输出返回

使物体进入休眠。

信息
关于休眠物体的解释参见 http://chipmunk-physics.net/release/Chipmunk-5.x/Chipmunk-5.3.4-Docs/

注意
在使物体进入休眠前必须将其加入一个空间(space)。不允许在队列(query)或回调函数（callback）中调用此函数。

API 级别限制：最低为 platform.apiLevel = ‘2.0’

19.4.32 sleepWithGroup

```
self = physics.Body:sleepWithGroup( [group] )
```

参数	类型	描述
self	**输入** physics.Body	输入物体
group	**输入** physics.Body	休眠物体，如果该参数没有提供，则创建一个新组
self	**输出** physics.Body	更改后的物体作为输出返回

使物体进入休眠并将其加入一个包含其它休眠物体的组(group)。

信息
此例程的解释参见 http://chipmunk-physics.net/release/Chipmunk-5.x/Chipmunk-5.3.4-Docs/

注意
在使物体进入休眠前必须将其加入一个空间(space)。不允许在队列(query)或回调函数（callback）中调用此函数。

如果 group 不是休眠状态，会引发一个异常。

API 级别限制：最低为 platform.apiLevel = ‘2.0’

19.4.33 torque

```
t = physics.Body:torque()
```

参数	类型	描述
self	**输入** physics.Body	输入物体
torque	**输出** number	物体的转矩

返回物体的转矩。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.4.34 updatePosition

```
physics.Body:updatePosition(dt)
```

参数	类型	描述
self	**输入** physics.Body	输入物体
dt	**输出** number	时间间隔（单位：秒）

使用欧拉积分更新物体的位置。

信息
此例程的解释参见 http://chipmunk-physics.net/release/Chipmunk-5.x/Chipmunk-5.3.4-Docs/

API 级别限制：最低为 platform.apiLevel = '2.0'

19.4.35 updateVelocity

```
physics.Body:updateVelocity(grav, damp, dt)
```

参数	类型	描述
self	**输入** physics.Body	输入物体
grav	**输入** physics.Vect	重力
damp	**输入** physics.Vect	阻尼系数
dt	**输入** physics.Vect	时间间隔（单位：秒

使用欧拉积分更新物体的矢量速度。

信息
此例程的解释参见 http://chipmunk-physics.net/release/Chipmunk-5.x/Chipmunk-5.3.4-Docs/

API 级别限制：最低为 platform.apiLevel = ‘2.0’

19.4.36 vel

```
vvel  = physics.Body:vel()
```

参数	类型	描述
self	**输入** physics.Body	输入物体
vvel	**输出** physics.Vect	物体的矢量速度

返回物体的矢量速度。

API 级别限制：最低为 platform.apiLevel = ‘2.0’

19.4.37 vLimit

```
vmax = physics.Body:vLimit()
```

参数	类型	描述
self	**输入** physics.Body	输入物体
vmax	**输出** number	物体的最大速度

返回物体的最大速度。

API 级别限制：最低为 platform.apiLevel = ‘2.0’

19.4.38 wLimit

```
wmax  = physics.Body:wLimit()
```

参数	类型	描述
self	**输入** physics.Body	输入物体
wmax	**输出** number	物体的的最大角速度

返回物体的最大角速度。角速度的单位是弧度/单位时间。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.4.39 world2Local

```
lvec  = physics.Body:world2Local(wvec)
```

参数	类型	描述
self	**输入** physics.Body	输入物体
wvec	**输入** physics.Vect	世界坐标系下的向量
lvec	**输出** physics.Vect	相对应与物体位置的向量

将参数 **wvec** 从世界坐标系(world coordinates)转换为相对于物体的坐标系。返回转换后的向量。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.5 Shapes

外形（Shapes）包含对象的表面性质，例如它的摩擦力和弹力大小。所有的碰撞外形(collision shape)均实现了下述的访问例程。

19.5.1 BB

```
bb = physics.Shape:BB()
```

参数	类型	描述
Self	**输入** physics.Shape	输入外形
Bb	**输出** physics.BB	外形的边界框

返回指定外形的边界框。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.5.2 body

```
body = physics.Shape:body()
```

参数	类型	描述
Self	**输入** physics.Shape	输入外形
Body	**输出** physics.Body	与外形相关联的物体

返回附加在外形上的物体。如果外形是静态的，那么它将返回 nil 。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.5.3 collisionType

```
coll = physics.Shape:collisionType()
```

参数	类型	描述
self	**输入** physics.Shape	输入外形
Coll	**输出** number	编程者指派的碰撞类型代号

返回碰撞类型(collision type)所对应的代号。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.5.4 data

```
obj  = physics.Shape:data()
```

参数	类型	描述
Self	**输入** physics.Shape	输入外形
Obj	**输出** Lua object	编程者已在外形上存储的 Lua 对象

返回编程者已经在外形数据域中存储的 Lua 对象。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.5.5 friction

```
f = physics.Shape:friction()
```

参数	类型	描述
Self	**输入** physics.Shape	输入外形
F	**输出** number	外形的摩擦系数

返回外形的摩擦系数。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.5.6 group

```
g = physics.Shape:group()
```

参数	类型	描述
Self	**输入** physics.Shape	输入外形
G	**输出** number	外形的组号

返回外形的组号。

注意
在存储时组号将被转换为正整数。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.5.7 layers

```
layers = physics.Shape:layers()
```

参数	类型	描述
Self	**输入** physics.Shape	输入外形
Layers	**输出** number	外形所占的层号

返回外形所占的层号。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.5.8 rawBB

```
bb = physics.Shape:rawBB()
```

参数	类型	描述
Self	**输入** physics.Shape	输入外形
Bb	**输出** physics.BB	外形的外接边界框

返回外形的外接边界框。仅在调用 physics.Shape:BB() 或 physics.Space:step()之后有效。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.5.9 restitution

```
r = physics.Shape:restitution()
```

参数	类型	描述
Self	**输入** physics.Shape	输入外形
R	**输出** number	外形的可恢复度

返回外形的可恢复度（弹性）。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.5.10 sensor

```
s = physics.Shape:sensor()
```

参数	类型	描述
Self	**输入** physics.Shape	输入外形
S	**输出** boolean	如果外形是一个传感器时为 true

如果外形是一个传感器时返回 true。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.5.11 setCollisionType

```
self = physics.Shape:setCollisionType(collisionType)
```

参数	类型	描述
Self	**输入** physics.Shape	输入外形
collisionType	**输入** number	编程者定义的碰撞类型代号
Self	**输出** physics.Shape	更改后的外形作为输出返回

给外形指派一个碰撞类型代号（一个你定义的整数值）。它用来决定当碰撞发生时调用哪个处理函数。返回 **self** 。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.5.12 setData

```
self = physics.Shape:setData(obj)
```

参数	类型	描述
Self	**输入** physics.Shape	输入外形
Obj	**输入** Lua object	编程者添加的 Lua 对象
Self	**输出** physics.Shape	更改后的外形作为输出返回

设置外形的数据域。编程者可以在这个域中存储任意 Lua 对象。返回 **self** 。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.5.13 setFriction

```
self = physics.Shape:setFriction(f)
```

参数	类型	描述
Self	**输入** physics.Shape	输入外形
F	**输入** number	外形表面的摩擦系数
Self	**输出** physics.Shape	更改后的外形作为输出返回

设置外形的摩擦系数。返回 self。

注意
当 f 大于 1 或小于 0 时，效果可能与预期不同。

API 级别限制：最低为 platform.apiLevel = ‘2.0’

19.5.14 setGroup

```
self = physics.Shape:setGroup(group)
```

参数	类型	描述
Self	**输入** physics.Shape	输入外形
Group	**输入** number	组号
Self	**输出** physics.Shape	更改后的外形作为输出返回

设置外形的组号（编程者定义的数字）。在同一组中的外形不会发生碰撞。

返回 **self**。

注意
在存储时组号将被转换为正整数

API 级别限制：最低为 platform.apiLevel = ‘2.0’

19.5.15 setLayers

```
self = physics.Shape:setLayers(layers)
```

参数	类型	描述
Self	**输入** physics.Shape	输入外形
Layers	**输入** number	一个表示层号的整数，最多允许 32 层
Self	**输出** physics.Shape	更改后的外形作为输出返回

设置外形所在的层。外形只会与同一层中的外形碰撞。参数 **layers** 是一个外形所占层的整数编号。
返回 **self**。

API 级别限制：最低为 platform.apiLevel = ‘2.0’

19.5.16 setRestitution

```
self = physics.Shape:setRestitution(r)
```

参数	类型	描述
Self	**输入** physics.Shape	输入外形
R	**输入** number	外形恢复度的新值
Self	**输出** physics.Shape	更改后的外形作为输出返回

设置外形的恢复度（弹性）。参数 **r** 为时 0.0 代表没有弹性而为 1.0 时代表完全弹性。返回 **self**。

注意
当 r 大于 1 或小于 0 时，效果可能与预期不同。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.5.17 setSensor

```
self = physics.Shape:setSensor(bool)
```

参数	类型	描述
Self	**输入** physics.Shape	输入外形
Bool	**输入** boolean	如果外形是传感器时为 true
Self	**输出** physics.Shape	更改后的外形作为输出返回

决定外形是传感器（true）或不是传感器（false）。传感器会调用碰撞处理函数(collision handlers)但并不发生碰撞。

返回 **self**。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.5.18 setSurfaceV

```
self = physics.Shape:setSurfaceV(vel)
```

参数	类型	描述
Self	**输入** physics.Shape	输入外形
Vec	**输入** physics.Vect	新的矢量表面速度
Self	**输出** physics.Shape	更改后的外形作为输出返回

设置外形的矢量表面速度。返回 **self**。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.5.19 surfaceV

```
sv  = physics.Shape:surfaceV()
```

参数	类型	描述
Self	**输入** physics.Shape	输入外形
Sv	**输出** physics.Vect	外形的矢量表面速度

返回外形的矢量表面速度。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.6 Circle Shapes

圆形(CircleShape)是外形的子类.它的类型是 TI.cpCircleShape。

19.6.1 CircleShape

```
cs  = physics.CircleShape(body, radius, offset)
```

参数	类型	描述
Body	**输入** physics.Body	一个物体或 nil
Radius	**输入** number	圆的半径
Offset	**输入** physics.Vect	圆相对于物体的偏移量(offset)
Cs	**输出** physics.CircleShape	一个新的圆形

根据给定的物体，半径和在基于物体的坐标系下到物体重心的偏移量，返回一个新的圆形。在使用空间(space)中的静态物体(static body)时要特别指定 nil 。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.6.2 offset

```
ovec = physics.CircleShape:offset()
```

参数	类型	描述
Self	**输入** physics.CircleShape	输入圆形
Ovec	**输出** physics.Vect	相对于物体的偏移量(offset)

返回圆形相对于物体重心的偏移向量。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.6.3 radius

```
r = physics.CircleShape:radius()
```

参数	类型	描述
Self	**输入** physics.CircleShape	输入圆形
R	**输出** number	圆形的半径

返回圆形的半径。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.7 Polygon Shapes

多边形(Polygon Shapes)是由一组线段组成的边框。多边形必须是凸多边形，且顶点必须以顺时针顺序定义。多边形的类型是 TI.cpPolyShape。

19.7.1 PolyShape

```
ps = physics.PolyShape(body, vertices, offset)
```

参数	类型	描述
Body	**输入** physics.Body	一个物体或 nil
Vertices	**输入** *{physics.Vect}*	以顺时针方向定义的多边形顶点列表
Offset	**输入** physics.Vect	多边形相对于物体的偏移量(offset)
Ps	**输出** physics.PolyShape	一个新的多边形

根据给定的物体，顶点列表，距离物体重心的偏移向量返回一个新的的多边形。在使用空间(space)中的静态物体(static body)时要特别指定 nil 。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.7.2 numVerts

```
nv = physics.PolyShape:numVerts()
```

参数	类型	描述
Self	**输入** physics.PolyShape	输入多边形
Nv	**输出** number	多边形的顶点个数

返回多边形顶点的个数。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.7.3 points

```
points = physics.PolyShape:points()
```

参数	类型	描述
Self	**输入** physics.PolyShape	输入多边形
points	**输出** *{physics.Vect}*	定义多边形边框的顶点列表，顶点被转换为多边形当前的世界坐标(world coordinates)

返回定义多边形边框的顶点表的副本。顶点被转换成多边形当前的世界坐标(world coordinates)

注意
当多边形没有被加入空间中时，它没有世界坐标。此时，physics.PolyShape:points()返回时每个顶点的 x 和 y 坐标都是 0。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.7.4 vert

```
v = physics.PolyShape:vert(n)
```

参数	类型	描述
self	**输入** physics.PolyShape	输入多边形
v	**输出** physics.Vect	多边形的第 n 个顶点，坐标为相对于外形物体的坐标

返回顶点表中第 **n** 个顶点。如果外形是静态的那么顶点坐标值是世界坐标(world coordinates)，否则是相对于外形物体的坐标。如果 n 比 1 小或大于多边形的顶点总数，返回 nil 。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.8 Segment Shapes
线段是用两个端点和一个厚度定义的。它的类型是 TI.cpSegmentShape。

19.8.1 SegmentShape

```
ss = physics.SegmentShape(body, a, b, radius)
```

参数	类型	描述
Body	**输入** physics.Body	一个物体或 nil
A	**输入** physics.Vect	线段的第一个端点，其坐标是相对于物体的。
B	**输入** physics.Vect	线段的第二个端点，其坐标是相对于物体的。
Radius	**输入** number	线段两端点之间的线与线段边界的距离
Ss	**输出** physics.SegmentShape	一个新的线段外形

返回一个给定端点 **a** 和 **b** 的新的线段外形，参数 **radius** 定义线段的厚度。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.8.2 a

```
avec = physics.SegmentShape:a()
```

参数	类型	描述
Self	**输入** physics.SegmentShape	输入线段外形
Avec	**输出** physics.Vect	线段的第一个端点

返回定义了线段第一个端点的向量 **a** 。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.8.3 b

```
bvec = physics.SegmentShape:b()
```

参数	类型	描述
Self	**输入** physics.SegmentShape	输入线段外形
Bvec	**输出** physics.Vect	线段的第二个端点

返回定义了线段第二个端点的向量 **b** 。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.8.4 normal

```
nvec = physics.SegmentShape:normal()
```

参数	类型	描述
Self	**输入** physics.SegmentShape	输入线段外形
Nvec	**输出** physics.Vect	线段的单位法向量

返回计算得到的线段的单位法向量(unit normal vector)。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.8.5 radius

```
r = physics.SegmentShape:radius()
```

参数	类型	描述
Self	**输入** physics.SegmentShape	输入线段外形
R	**输出** number	线段的厚度

返回线段的厚度。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.9 Spaces

物理空间(physics Space)是仿真的基本单位。

19.9.1 Space

```
s = physics.Space()
```

参数	类型	描述
S	**输出** physics.Space	一个新的仿真空间

返回一个新的物理仿真空间。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.9.2 addBody

```
self = physics.Space:addBody(body)
```

参数	类型	描述
Self	**输入** physics.Space	输入仿真空间
Body	**输入** physics.Body	要添加到仿真空间中的物体
Self	**输出** physics.Space	更改后的空间作为输出返回

向空间中添加一个物体。返回 **self**。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.9.3 addConstraint

```
self = physics.Space:addConstraint(constraint)
```

参数	类型	描述
Self	**输入** physics.Space	输入仿真空间
Constraint	**输入** physics.Constraint	要添加到仿真空间中的约束
Self	**输出** physics.Space	更改后的空间作为输出返回

向空间中添加一个约束（Constraint）。返回 **self**。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.9.4 addCollisionHandler

```
self = physics.Space:addCollisionHandler(collisionTypeA, collisionTypeB, callbacksTable)
```

参数	类型	描述
Self	**输入** physics.Space	输入仿真空间 collisionTypeA
collisionTypeA	**输入** number	第一个碰撞类型的代号
collisionTypeB	**输入** number	第二个碰撞类型的代号
callbacksTable	**输入** table of functions	碰撞检测和处理时调用的函数表
Self	**输出** physics.Space	更改后的空间作为输出返回

注册处理 collisionTypeA 外形和 collisionTypeB 外形间碰撞(collision)的处理函数表。参数 **callbacksTable** 是以如下格式提供的函数表：

```
{
  begin  = function(arbiter, space, callbacksTable) ... end,
  preSolve  = function(arbiter, space, callbacksTable) ... end,
  postSolve  = function(arbiter, space, callbacksTable) ... end,
  separate  = function(arbiter, space, callbacksTable) ... end
}
```

如果 **begin** 处理函数或 **preSolve** 处理函数返回 false，则略过进一步的碰撞计算。如果返回 true，将继续正常进行碰撞处理。

没有必要为处理表中所有条目提供处理函数。对未指定的条目将会提供默认处理函数。

返回 **self**。

有关碰撞处理和调用碰撞处理函数（collision handler callbacks）的说明请参见 http://chipmunk-physics.net/release/Chipmunk-5.x/Chipmunk-5.3.4-Docs/

需要特别注意的一点是这些处理函数不能够从空间中添加或移除物体(Bodies)，外形(Shapes)，或约束(Constraints)。

对于产生碰撞后，正确的移除(或添加)对象的方法参见 post-step callback 函数。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.9.5 addPostStepCallback

```
self = physics.Space:addPostStepCallback(body|shape|constraint,
                                         function(space, object)
                                         ... end )
```

参数	类型	描述
self	**输入** physics.Space	输入仿真空间
body or shape or constraint	**输入** physics.Body 或 physics.Shape 或 physics.Constraint	仿真步骤之后将要被关注的仿真对象
function	**输入** function(space, object)	仿真步骤结束时触发的处理函数
self	**输出** physics.Space	更改后的空间作为输出返回

添加一个处理函数，这个函数会在当前步骤完成时调用。每个处理函数可能与一个物体(Body)，外形(Shape)，或 约束(Constraint)关联。对于一个指定的对象，仅第一个处理函数会被登记。任何对同一个对象注册更多的处理函数都会被忽略。

返回 **self**。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.9.6 addShape

```
self = physics.Space:addShape(shape)
```

参数	类型	描述
self	**输入** physics.Space	输入仿真空间
shape	**输入** physics.Space	要添加到仿真空间中的外形
self	**输出** physics.Shape	更改后的空间作为输出返回

向空间中添加一个外形。返回 **self**。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.9.7 addStaticShape

```
self = physics.Space:addStaticShape(staticShape)
```

参数	类型	描述
self	**输入** physics.Space	输入仿真空间
staticShape	**输入** physics.Space	要添加到仿真空间中的静态外形
self	**输出** physics.Shape	更改后的空间作为输出返回

向空间中添加一个静态外形。返回 **self**。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.9.8 damping

```
d = physics.Space:damping()
```

参数	类型	描述
self	**输入** physics.Space	输入仿真空间
d	**输出** number	仿真空间的阻尼值

API 级别限制：最低为 platform.apiLevel = '2.0'

19.9.9 data

```
obj  = physics.Space:data()
```

参数	类型	描述
self	**输入** physics.Space	输入仿真空间
obj	**输入** Lua 对象	由编程者指定的与空间相关联的对象
self	**输出** physics.Shape	更改后的空间作为输出返回

API 级别限制：最低为 platform.apiLevel = '2.0'

19.9.10 elasticIterations

```
iters = physics.Space:elasticIterations()
```

参数	类型	描述
self	**输入** physics.Space	输入仿真空间
iters	**输出** number	在使用冲量求解器(impulse solver)解决弹性碰撞(elastic collisions)时的迭代次数

API 级别限制：最低为 platform.apiLevel = '2.0'

19.9.11 gravity

```
grav  = physics.Space:gravity()
```

参数	类型	描述
self	**输入** physics.Space	输入仿真空间
grav	**输出** physics.Vect	仿真空间中作用于所有物体上的重力向量(gravity force vector)

API 级别限制：最低为 platform.apiLevel = '2.0'

19.9.12 idleSpeedThreshold

```
speed = physics.Space:idleSpeedThreshold()
```

参数	类型	描述
self	**输入** physics.Space	输入仿真空间
speed	**输出** number	速度的临界值

API 级别限制：最低为 platform.apiLevel = '2.0'

19.9.13 iterations

```
iters = physics.Space:iterations()
```

参数	类型	描述
self	**输入** physics.Space	输入仿真空间
iters	**输出** number	求解器（solver）更新一步仿真所用的迭代次数

API 级别限制：最低为 platform.apiLevel = '2.0'

19.9.14 rehashShape

```
self = physics.Space:rehashShape(shape)
```

参数	类型	描述
self	**输入** physics.Space	输入仿真空间
shape	**输入** shape	重新处理后的外形
self	**输出** physics.Shape	更改后的空间作为输出返回

更新一个已被移动的独立静态外形。

返回 **self**。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.9.15 rehashStatic

```
self = physics.Space:rehashStatic()
```

参数	类型	描述
self	**输入** physics.Space	输入仿真空间
self	**输出** physics.Shape	更改后的空间作为输出返回

重新处理静态空间散列(static spatial hash)中的外形。如果移动任何静态外形，必须调用这个函数，否则将不会更新他们的碰撞检测数据。

返回 **self**。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.9.16 removeBody

```
self = physics.Space:removeBody(body)
```

参数	类型	描述
self	**输入** physics.Space	输入仿真空间
body	**输入** physics.Body	想要从仿真空间中移除的物体
self	**输出** physics.Shape	更改后的空间作为输出返回

从空间中将一个物体移除。 返回 **self**。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.9.17 removeConstraint

```
self = physics.Space:removeConstraint(constraint)
```

参数	类型	描述
self	**输入** physics.Space	输入仿真空间
constraint	**输入** physics.Constraint	想要从仿真空间中移除的约束
self	**输出** physics.Shape	更改后的空间作为输出返回

从空间中移除一个约束。 返回 **self**。

API 级别限制：最低为 platform.apiLevel = ‘2.0’

19.9.18 removeShape

```
self = physics.Space:removeShape(shape)
```

参数	类型	描述
self	**输入** physics.Space	输入仿真空间
shape	**输入** physics.Space	想要从仿真空间中移除的外形
self	**输出** physics.Shape	更改后的空间作为输出返回

从空间中移除一个外形。 返回 **self。**

API 级别限制：最低为 platform.apiLevel = ‘2.0’

19.9.19 removeStaticShape

```
physics.Space:removeStaticShape(staticShape)
```

参数	类型	描述
self	**输入** physics.Space	输入仿真空间
staticShape	**输入** physics.Space	想要从仿真空间中移除的外形
self	**输出** physics.Shape	更改后的空间作为输出返回

从空间中移除一个静态外形。 返回 **self**。

API 级别限制：最低为 platform.apiLevel = ‘2.0’

19.9.20 resizeActiveHash

```
self = physics.Space:resizeActiveHash(dim, count)
```

参数	类型	描述
self	**输入** physics.Space	输入仿真空间
dim	**输入** number	每个散列单元(hash cell)一边的长度，缺省为 100
count	**输入** number	散列表中单元(cell)的个数，缺省为 1000
self	**输出** physics.Shape	更改后的空间作为输出返回

通过调整活动外形的空间散列表可以改善碰撞检测。参数 **dim** 用于设置散列单元(hash cell)的大小(默认 100)，参数 **count** 设置散列单元的数量(默认 1000)。参数 **dim** 应该接近于一个典型外形的边长。根据经验将参数设置 **count** 为输入空间中外形 数量的 10 倍是比较好的。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.9.21 resizeStaticHash

```
self = physics.Space:resizeStaticHash(dim, count)
```

参数	类型	描述
self	**输入** physics.Space	输入仿真空间
dim	**输入** number	每个散列单元(hash cell)一边的长度，缺省为 100
count	**输入** number	散列表中单元(cell)的个数，缺省为 1000
self	**输出** physics.Shape	更改后的空间作为输出返回

配置静态外形的空间散列表. 除作用于静态外形外，配置方法与 resizeActiveHash 类似。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.9.22 setDamping

仿真空间中的阻尼会降低物体的速度。例如取值为 0.9 时，表示每一个物体速度每秒会减少 10%。阻尼默认为 1。这个值会作用在每一个物体上。

```
self = physics.Space:setDamping(d)
```

参数	类型	描述
self	**输入** physics.Space	输入仿真空间
d	**输入** number	仿真空间中新的阻尼值
self	**输出** physics.Shape	更改后的空间作为输出返回

设置空间中的粘滞阻尼值。

注意
当 d 大于 1.0 或小于 0 时，效果可能与预期不同。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.9.23 setData

```
self = physics.Space:setData(obj)
```

参数	类型	描述
self	**输入** physics.Space	输入仿真空间
obj	**输入** Lua 对象	编程者指定的 Lua 对象
self	**输出** physics.Shape	更改后的空间作为输出返回

编程者可以在此区域存储任意 Lua 对象。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.9.24 setElasticIterations

```
self = physics.Space:setElasticIterations(iters)
```

参数	类型	描述
self	**输入** physics.Space	输入仿真空间
iters	**输入** number	在使用冲量求解器(impulse solver)解决弹性碰撞(elastic collisions)时的迭代次数，默认为 0
self	**输出** physics.Shape	更改后的空间作为输出返回

API 级别限制：最低为 platform.apiLevel = '2.0'

19.9.25 setGravity

```
self = physics.Space:setGravity(grav)
```

参数	类型	描述
self	**输入** physics.Space	输入仿真空间
grav	**输入** physics.Vect	仿真空间中作用于所有物体的重力向量，默认为 physics.Vect(0, 0)
self	**输出** physics.Shape	更改后的空间作为输出返回

设置空间中作用于全局的重力。可以通过自定义积分函数在基于每个物体重写。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.9.26 setIdleSpeedThreshold

```
self = physics.Space:setIdleSpeedThreshold(speed)
```

参数	类型	描述
self	**输入** physics.Space	输入仿真空间
speed	**输入** number	速度的临界值
self	**输出** physics.Shape	更改后的空间作为输出返回

设置一个临界速度，在此速度之下的物体被认为是空闲的。 这个值用于决定一个物体何时可以被置于休眠状态。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.9.27 setIterations

```
self = physics.Space:setIterations(iters)
```

参数	类型	描述
self	**输入** physics.Space	输入仿真空间
iters	**输入** number	改进求解器（solver）精度所需要的迭代次数，默认为 10
self	**输出** physics.Shape	更改后的空间作为输出返回

这个值允许编程者控制求解器的精度。默认为 10。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.9.28 setSleepTimeThreshold

```
self = physics.Space:setSleepTimeThreshold(sleep)
```

参数	类型	描述
self	**输入** physics.Space	输入仿真空间
sleep	**输入** number	时间长度（单位：秒），一个外形静止的时间低于此值时，则进入休眠状态。
self	**输出** physics.Shape	更改后的空间作为输出返回

休眠时间限制(Sleep time threshold)用于计算一个物体何时可被置于休眠状态。

API 级别限制：最低为 platform.apiLevel = ‘2.0’

19.9.29 sleepTimeThreshold

```
sleep  = physics.Space:sleepTimeThreshold()
```

参数	类型	描述
self	**输入** physics.Space	输入仿真空间
sleep	**输出** number	用于判断何时一个外形可被置于休眠状态的时间限制值

API 级别限制：最低为 platform.apiLevel = ‘2.0’

19.9.30 step

```
self = physics.Space:step(dt)
```

参数	类型	描述
self	**输入** physics.Space	输入仿真空间
dt	**输入** number	一步仿真所需要的时间长度（单位:秒）
self	**输出** physics.Shape	更改后的空间作为输出返回

按照给定的时间间隔 **dt** 更新空间。建议使用固定的时间间隔以增加接触持久性(contact persistence)的效率，这样只需要更低数量级的迭代次数，和更少的 CPU 使用。

返回 **self**。

API 级别限制：最低为 platform.apiLevel = ‘2.0’

19.10 Constraints

所有约束(Constraints)共用相同的接口(accessor)。

接口	类型	描述
bodyA	physics.Body	约束作用的第一个物体
bodyB	physics.Body	约束作用的第二个物体
setBiasCoef, biasCoef	number	仿真中每一步修正的误差的小数，默认为 0.1，大于 1 或小于 0 时行为可能不像预期的那样
setData, data	Lua object	编程者定义的对象
impulse	number	在最后的仿真步骤中由约束计算得到的冲量。为了把这一冲量转化为力的大小，需要除以传递给 physics.Space:step()的时间间隔
setMaxBias, maxBias	number	约束能够用于误差校正的最大速度。缺省为 INFINITY
setMaxForce, maxForce	number	约束可以作用于两个物体上的最大力度，缺省为 INFINITY

19.10.1 Damped Rotary Spring

```
spring = physics.DampedRotarySpring(a, b, restAngle, stiffness, damping)
```

参数	类型	描述
a	**输入** physics.Body	第一个物体
b	**输入** physics.Body	第二个物体
restAngle	**输入** number	物体想要保持的相对弧度
stiffness	**输入** number	弹性常数
damping	**输入** number	弹簧阻尼的柔软程度
spring	**输出** physics.DampedRotarySpring	新的阻尼旋转弹簧

与阻尼弹簧(damped spring)相似，但在有角度的方式下工作。参数 **restAngle** 是物体想要保持的相对弧度。参数 **stiffness** 和 **damping** 与在阻尼弹簧中的作用相同。

接口	类型
setRestAngle, restAngle	number
setStiffness, stiffness	number
setDamping, damping	number

API 级别限制：最低为 platform.apiLevel = '2.0'

19.10.2 Damped Spring

```
spring = physics.DampedSpring(a, b,  anchr1,  anchr2,  restLength, stiffness, damping)
```

参数	类型	描述
a	**输入** physics.Body	第一个物体
b	**输入** physics.Body	第二个物体
anchr1	**输入** physics.Vect	第一个物体的定位点
Anchr2	**输入** physics.Vect	第二个物体的定位点
restLength	**输入** number	弹簧原长
stiffness	**输入** number	弹性常数
damping	**输入** number	弹簧阻尼的柔软程度
spring	**输出** physics.DampedSpring	新的阻尼弹簧(DampedSpring)

与的 SlideJoint 定义非常相似。参数 **restLength** 弹簧的原长，**stiffness** 是弹性常数， **damping** 表示弹簧阻尼的柔软程度。

接口	类型
setAnchr1, anchr1	physics.Vect
setAnchr2, anchr2	physics.Vect
setRestLength, restLength	number
setStiffness, stiffness	number
setDamping, damping	number

API 级别限制：最低为 platform.apiLevel = '2.0'

19.10.3 Gear Joint

```
joint = physics.GearJoint(a, b,  phase,  ratio)
```

参数	类型	描述
a	**输入** physics.Body	第一个物体
b	**输入** physics.Body	第二个物体
phase	**输入** number	两个物体的初始弧度角偏移量
ratio	**输入** number	两个物体速度的比值
joint	**输出** physics.GearJoint	新的齿轮连接(GearJoint)

保持一对物体的角速度比值恒定。参数 **ratio** 一般按照绝对值计算。参数 **phase** 是两个物体的初始弧度角偏移量。

接口	类型
setPhase, phase	number
setRatio, ratio	number

API 级别限制：最低为 platform.apiLevel = '2.0'

19.10.4 Groove Joint

```
joint = physics.GrooveJoint(a, b, grooveA, grooveB, anchr2)
```

参数	类型	描述
a	**输入** physics.Body	第一个物体
b	**输入** physics.Body	第二个物体
grooveA	**输入** physics.Vect	槽(groove)的一个端点
grooveB	**输入** physics.Vect	槽的另一个端点
anchr2	**输入** physics.Vect	物体 b 的支点
joint	**输出** physics.GlooveJoint	新的槽式连接(GlooveJoint)

凹槽是在物体 **a** 上从 **grooveA** 到 **grooveB**，支点附着于物体 **b** 上的 **anchr2** 。

所有的坐标为相对于物体的坐标。

接口	类型
setAnchr2, anchr2	physics.Vect
setGrooveA, grooveA	physics.Vect
setGrooveB, grooveB	physics.Vect
grooveN	physics.Vect

API 级别限制：最低为 platform.apiLevel = '2.0'

19.10.5 Pin Joint

```
joint = physics.PinJoint(a, b,  anchr1,  anchr2)
```

参数	类型	描述
a	**输入** physics.Body	第一个物体
b	**输入** physics.Body	第二个物体
anchr1	**输入** physics.Vect	物体 a 上的定位点
anchr2	**输入** physics.Vect	物体 b 上的定位点
joint	**输出** physics.PinJoint	新的铰链连接(PinJoint)

a 和 **b** 是需要连接的两个物体，参数 **anchr1** 和 **anchr2** 是这两个物体上的定位点。当连接建立时会自动计算计算两个定位点的距离。如果想设置一个具体的距离，使用 setter 函数来覆盖这个距离。

接口	类型
setAnchr1, anchr1	physics.Vect
setAnchr2, anchr2	physics.Vect
setDist, dist	number

API 级别限制：最低为 platform.apiLevel = '2.0'

19.10.6 Pivot Joint

```
joint = physics.PivotJoint(a, b,  pivot)
joint = physics.PivotJoint(a, b,  anchr1,  anchr2)
```

参数	类型	描述
a	**输入** physics.Body	第一个物体
b	**输入** physics.Body	第二个物体
pivot	**输入** physics.Vect	两个物体间的支点
anchr1	**输入** physics.Vect	物体 a 上的定位点
anchr2	**输入** physics.Vect	物体 b 上的定位点
joint	**输出** physics.PivotJoint	新的枢接(PivotJoint)

a 和 **b** 是需要连接的两个物体，参数 **pivot** 是支点的世界坐标。由于支点位置是以世界坐标给出，就必须要求两个物体已经被移动到正确的位置。或者可以根据一对定位点指定连接，但要确保物体在正确位置，因为连接会在空间开始仿真的同时固定。

接口	类型
setAnchr1, anchr1	physics.Vect
setAnchr2, anchr2	physics.Vect

API 级别限制：最低为 platform.apiLevel = '2.0'

19.10.7 Ratchet Joint

```
joint = physics.RatchetJoint(a, b, phase, ratchet)
```

参数	类型	描述
a	**输入** physics.Body	第一个物体
b	**输入** physics.Body	第二个物体
phase	**输入** number	初始偏移量（单位：弧度）
ratchet	**输入** number	棘轮每次动作(click)间的距离（单位：弧度）
joint	**输出** physics.RatchetJoint	新的棘轮链接(RatchetJoint)

工作起来类似于套筒扳手。参数 **ratchet** 是棘轮每次动作间的距离， 参数 **phase** 是在决定棘齿位置时所使用的初始偏移量。

接口	类型
setAngle, angle	number
setPhase, phase	number
setRatchet, ratchet	number

API 级别限制：最低为 platform.apiLevel = '2.0'

19.10.8 Rotary Limit Joint

```
joint = physics.RotaryLimitJoint(a, b, min, max)
```

参数	类型	描述
a	**输入** physics.Body	第一个物体
b	**输入** physics.Body	第二个物体
min	**输入** number	最小的角距（单位：弧度）
max	**输入** number	最大的角距（单位：弧度
joint	**输出** physics.RotaryLimitJoint	新的旋转限制连接(RotaryLimitJoint)

限制了两个物体的相对转动。参数 **min** 和 **max** 是以弧度为单位的限制。它使得转动角度可以超过一次完全的旋转。

接口	类型
setMin, min	number
setMax, max	number

API 级别限制：最低为 platform.apiLevel = '2.0'

19.10.9 Simple Motor

```
motor = physics.SimpleMotor(a, b,  rate)
```

参数	类型	描述
a	**输入** physics.Body	第一个物体
b	**输入** physics.Body	第二个物体
rate	**输入** number	相对角速度
motor	**输出** physics.SimpleMotor	新的(简单马达) SimpleMotor

使一对物体的相对角速度保持恒定。参数 **rate** 是想要保持的相对角速度大小。

接口	类型
setRate, rate	number

API 级别限制：最低为 platform.apiLevel = '2.0'

19.10.10 Slide Joints

```
joint = physics.SlideJoint(a, b, anchr1, anchr2, min, max)
```

参数	类型	描述
a	**输入** physics.Body	第一个物体
b	**输入** physics.Body	第二个物体
anchr1	**输入** physics.Vect	物体 a 上的定位点
anchr2	**输入** physics.Vect	物体 b 上的定位点
min	**输入** number	物体间的最小距离
max	**输入** number	物体间的最大距离
joint	**输出** physics.SlideJoint	新的滑动连接(SlideJoint)

a 和 **b** 是需要连接的两个物体，参数 **anchr1** 和 **anchr2** 是这两个物体上的定位点，参数 **min** 和 **max** 定义两个定位点间所允许的距离范围。

接口	类型
setAnchr1, anchr1	physics.Vect
setAnchr2, anchr2	physics.Vect
setMin, min	number
setMax, max	number

API 级别限制：最低为 platform.apiLevel = '2.0'

19.11 Arbiters and Collision Pairs

仲裁器(Arbiter)类封装了每一对碰撞的信息。

19.11.1

```
count = #physics.Arbiter
```

返回 Arbiter 中接触点的数量。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.11.2 a

```
shape = physics.Arbiter:a()
```

参数	类型	描述
self	**输入** physics.Arbiter	输入仲裁器
shape	**输出** physics.Shape	碰撞对里边的第一个外形

返回碰撞对(collision pair)中的外形 **a** (第一个 外形)。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.11.3 b

```
shape = physics.Arbiter:b()
```

参数	类型	描述
self	**输入** physics.Arbiter	输入仲裁器
shape	**输出** physics.Shape	碰撞对里边的第二个外形

返回碰撞对中的外形 **b** (第二个 外形)。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.11.4 bodies

```
bodyA, bodyB = physics.Arbiter:bodies()
```

参数	类型	描述
self	**输入** physics.Arbiter	输入仲裁器
bodyA	**输入** physics.Body	碰撞对中的第一个物体
bodyB	**输出** physics.Body	碰撞对中的第二个物体

返回碰撞对中的 bodyA 和 bodyB。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.11.5 depth

```
d = physics.Arbiter:depth(i)
```

参数	类型	描述
self	**输入** physics.Arbiter	输入仲裁器
i	**输入** number	接触点编号
d	**输出** number	第 i 个接触点的穿透深度(penetration depth)

返回第 i 个接触点的穿透深度，当参数 **i** 超出接触点的数量时返回 nil 。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.11.6 elasticity

```
e = physics.Arbiter:elasticity()
```

参数	类型	描述
self	**输入** physics.Arbiter	输入仲裁器
e	**输出** number	计算得到的碰撞对的弹性系数

返回碰撞对的弹性系数。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.11.7 friction

```
f = physics.Arbiter:friction()
```

参数	类型	描述
self	**输入** physics.Arbiter	输入仲裁器
f	**输出** number	计算得到的碰撞对的摩擦系数

返回碰撞对的摩擦系数。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.11.8 impulse

```
ivec = physics.Arbiter:impulse([friction])
```

参数	类型	描述
self	**输入** physics.Arbiter	输入仲裁器
friction	**输入** boolean	值为 true 时，运算时将包含计算出的摩擦力
ivec	**输出** physics.Vect	用于碰撞求解的冲量

返回应用于这一步骤中来求解碰撞的冲量(vector impulse)。若参数 **friction** 为 true(默认为 false)时，计算得到的摩擦力也会被考虑进来。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.11.9 isFirstContact

```
bool = physics.Arbiter:isFirstContact()
```

参数	类型	描述
self	**输入** physics.Arbiter	输入仲裁器
bool	**输出** boolean	如果当前步骤是外形首次接触，则值为 true

如果这是外形接触的第一步，则返回 true。这一信息仅持续到外形不再接触。一旦他们不再接触，这一标志将被重置。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.11.10 normal

```
nvec = physics.Arbiter:normal(i)
```

参数	类型	描述
self	**输入** physics.Arbiter	输入仲裁器
i	**输入** number	接触点编号
nvec	**输出** physics.Vect	垂直于第 i 个接触点的向量

返回第 i 个接触点的碰撞法向量(normal vector)。当参数 **i** 超出接触点数量时返回 nil 。

API 级别限制：最低为 platform.apiLevel = ‘2.0’

19.11.11 point

```
pvec = physics.Arbiter:point(i)
```

参数	类型	描述
self	**输入** physics.Arbiter	输入仲裁器
i	**输入** number	接触点编号
pvec	**输出** physics.Vect	第 i 个接触点的位置

返回第 i 个接触点的位置。当参数 **i** 超出接触点数量时返回 nil。

API 级别限制：最低为 platform.apiLevel = ‘2.0’

19.11.12 setElasticity

```
self = physics.Arbiter:setElasticity(e)
```

参数	类型	描述
self	**输入** physics.Arbiter	输入仲裁器
e	**输入** number	碰撞的弹性系数
self	**输出** physics.Arbier	更改后的仲裁器作为输出返回

重设碰撞的弹性系数。

注意
当 e 大于 1.0 或小于 0 时，效果可能与预期不同。

API 级别限制：最低为 platform.apiLevel = ‘2.0’

19.11.13 setFriction

```
self = physics.Arbiter:setFriction(friction)
```

参数	类型	描述
self	**输入** physics.Arbiter	输入仲裁器
f	**输入** number	碰撞的摩擦系数
self	**输出** physics.Arbier	更改后的仲裁器作为输出返回

重设碰撞的摩擦系数。

注意
当 f 大于 1.0 或小于 0 时，效果可能与预期不同。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.11.14 shapes

```
shapeA, shapeB = physics.Arbiter:shapes()
```

参数	类型	描述
self	**输入** physics.Arbiter	输入仲裁器
shapeA	**输入** physics.Shape	碰撞中的第一个外形
shapeB	**输出** physics.Shape	碰撞中的第二个外形

按照碰撞处理函数(collision handler)中定义的顺序和与当前仲裁器关联的顺序返回 shapeA 和 shapeB 。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.11.15 totalImpulse

```
ivec  = physics.Arbiter:totalImpulse()
```

参数	类型	描述
self	**输入** physics.Arbiter	输入仲裁器
ivec	**输出** physics.Vect	用于求解碰撞的冲量

返回当前步骤中用来求解碰撞的冲量。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.11.16 totalImpulseWithFriction

```
ivec  = physics.Arbiter:totalImpulseWithFriction()
```

参数	类型	描述
self	**输入** physics.Arbiter	输入仲裁器
ivec	**输出** physics.Vect	用于求解碰撞的冲量

返回当前步骤中用来求解碰撞的冲量，此处考虑了摩擦。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.12 Shape Queries

19.12.1 pointQuery

```
bool  = physics.Shape:pointQuery(point)
```

参数	类型	描述
self	**输入** physics.Shape	输入外形
point	**输入** physics.Vect	一个点
bool	**输出** boolean	如果 **point** 位于外形边界范围内，返回 true

如果 **point** 位于外形内部，返回 true 。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.12.2 segmentQuery

```
info = physics.Shape:segmentQuery(vecta, vectb)
```

参数	类型	描述
self	**输入** physics.Shape	输入外形
vecta	**输入** physics.Vect	线段的一个端点
vectb	**输入** physics.Vect	线段的另一个端点
info	**输出** physics.SegmentQueryInfo	线段与外形交点的信息若不相交，返回 nil

检查从 **vecta** 到 **vectb** 的线段是否与 外形相交。在查询结果中返回一个 SegmentQueryInfo 对象，如果没有相交则返回 nil 。

如果一次查询开始于一个外形的内部，则结果是不确定的。圆形和多边形不会报告与这个外形存在碰撞，线段会报告一个错误点，或检测到碰撞后报告正常。为了避免这一缺陷，请先使用分开的点查询(point query)来确定线段查询(segment query)是否开始于一个外形内部。

将结果转换到世界坐标系或绝对距离的方法请参见 SegmentQueryInfo。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.13 Space Queries

19.13.1 pointQuery

```
physics.Space:pointQuery(point, layers,  group,
                         function(shape) ... end)
```

参数	类型	描述
self	**输入** physics.Space	输入空间
point	**输入** physics.Vect	一个点
layers	**输入** number	layers 的层编号。若 shape.layers 与 **layers** 相交，则匹配。
group	**输入** number	要检查的组(group)编号，若外形不在 **group** 中，则匹配。
function	**输出** physics.Vect	提供每个外形轮流匹配的标准函数

查询空间中所有包含 **point**，与 **layers** 匹配但不在 **group** 中的外形。**function** 调用时的参数 shape 为每个外形，包括传感器外形(Sensor Shapes)。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.13.2 pointQueryFirst

```
shape = physics.Space:pointQueryFirst(point, layers, group)
```

参数	类型	描述
self	**输入** physics.Space	输入空间
point	**输入** physics.Vect	一个点
layers	**输入** number	layers 的层编号。若 shape.layers 与 **layers** 相交，则匹配。
group	**输入** number	要检查的组(group)编号，若外形不在 **group** 中，则匹配。

查询空间中包含 **point**，与 **layers** 匹配但不在 **group** 中的第一个外形。若没有找到符合的外形，返回 nil。忽略传感器外形(Sensor Shapes)。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.13.3 segmentQuery

```
physics.Space:segmentQuery(startvect, stopvect, layers, group,
                           function(shape, t, normal) ... end)
```

参数	类型	描述
self	**输入** physics.Space	输入空间
startvect	**输入** physics.Vect	线段的一个端点
stopvect	**输入** physics.Vect	线段的另一个端点
layers	**输入** number	layers 的层编号。若 shape.layers 与 **layers** 相交，则匹配。
group	**输入** number	要检查的组(group)编号，若外形不在 **group** 中，则匹配。
function	function(shape, t, normal)	提供每个外形轮流匹配的标准函数

查询空间中所有与从 **startvect** 到 **stopvect** 的线段相交，与 **layers** 匹配且在 **group** 中的外形。function 的参数包括每个外形，沿线段的距离比例（0 到 1 之间的小数）和外形交点的表面法向量，包括传感器外形。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.13.4 segmentQueryFirst

```
info = physics.Space:segmentQueryFirst(startvect, stopvect, layers, group)
```

参数	类型	描述
self	**输入** physics.Space	输入空间
startvect	**输入** physics.Vect	线段的一个端点
stopvect	**输入** physics.Vect	线段的另一个端点
layers	**输入** number	layers 的层编号。若 shape.layers 与 **layers** 相交，则匹配。
group	**输入** number	要检查的组(group)编号，若外形不在 **group** 中，则匹配。
info	**输出** physics.SegmentQueryInfo	线段与外形交点的信息若不相交，返回 nil。

沿着线段从 **startvect** 到 **layers** 查询空间，与 **layers** 匹配且在 **group** 中的相交外形。返回第一个匹配的外形的 SegmentQueryInfo 对象，不相交则返回 nil。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.14 SegmentQueryInfo

SegmentQueryInfo 对象是一个带有三个域的 Lua 数据表(dictionary table)。

键	值
shape	查询所找到的外形对象
t	从线段起始点到交点的距离比(0 .. 1)
n	外形在交点处的表面法向量

外形在交点处的表面法向量

这一对象有以下例程来将 SegmentQueryInfo 对象中的信息转换为世界坐标系或沿线段的绝对距离。

19.14.1 hitDist

```
d = SegmentQueryInfo:hitDist(startvect, stopvect)
```

参数	类型	描述
self	**输入** physics.SegmentQueryInfo	输入 SegmentQueryInfo
startvect	**输入** physics.Vect	线段的一个端点
stopvect	**输入** physics.Vect	线段的另一个端点
d	**输出** number	相交距离

返回线段首次与外形相交的绝对距离(absolute distance)。

API 级别限制：最低为 platform.apiLevel = '2.0'

19.14.2 hitPoint

```
p = SegmentQueryInfo:hitPoint(startvect, stopvect)
```

参数	类型	描述
self	**输入** physics.SegmentQueryInfo	输入 SegmentQueryInfo
startvect	**输入** physics.Vect	线段的一个端点
stopvect	**输入** physics.Vect	线段的另一个端点
p	**输出** physics.Vect	交点

返回满足从 **startvect** 到 **stopvect** 的线段与外形相交的世界坐标系(world coordinates)下的第一个交点。

API 级别限制：最低为 platform.apiLevel = '2.0'

附录 A

脚本语言兼容性(Script Compatibility)

此附录概括了 Lua 脚本语言在 TI-Nspire™平台上存在的各种兼容性问题以及其概念。这将支持为不同的 TI-Nspire™软件版本以及不同平台编写文档。参见 A.2 节以了解如何在脚本语言开发环境中为比当前支持级别更高的 API 级别编写文档。

A.1 前向及后向兼容性

在 TI-Nspire™平台中有两种兼容性概念。接下来的章节中描述了这些概念以及它们之间的相互作用。有必要理解这些以便可以让文档在不同的 TI-Nspire™软件版本上运行。如果不需要这些特性，你可以直接略过 A.1 节直接阅读 A.2 节。

A.1.1 文档兼容性

这是 TI-Nspire™平台中一个旧的概念。在每个文档中保存了两个不同的 TI-Nspire™发行版本值—文档“最后一次保存”(“last saved”)所使用的版本以及一个“最低需求”(“minimum requested”)版本。任何 TI-Nspire™发行版本如果比”最低需求”版本低，那么会阻止文档的打开。如果 TI-Nspire™发行版本至少是“最后一次保存”版本级别的，那么文档打开时不会有任何警告。

这一概念最近已经被增强了。“最低需求”版本现在动态地取决于内容。这将允许一个更低的“最低需求”版本。然而，改变文档的内容可能会导致“最低需求”版本升高。

对后向兼容性感兴趣的脚本语言作者需要知道，更改同一个文档中其他非脚本语言内容同样可能会更改“最低需求”版本。当前，只有在各种不同版本的 TI-Nspire™发行版本中打开文档，才能让脚本语言作者理解什么是“最低需求”版本。

如果文档仅包含脚本语言，那么规则很简单。文档会被打开，但是如果不支持脚本语言使用的 API 级别，那么脚本语言可能会执行失败。TI-Nspire™软件 3.1 版本是能打开包含脚本语言的文档的最早版本。除此之外，3.1 发行版本仅在文档所包含脚本语言的 API 级别全部是 **platform.apiLevel = '1.0'**时才可以打开文档。

A.1.2 脚本语言编写兼容性

为平台编写的脚本语言默认都具有前向兼容性，尽管脚本语言是在特定平台上设计并测试的(平台兼容性将在 A.3 节中讨论)。API 级别是确保前向兼容性的关键。API 级别指明了在特定 TI-Nspire™软件版本中编写脚本语言所能使用的接口。在表 A.1 中指明了软件版本与其支持的最高 API 级别。

最高支持的 API 级别，或者说当前软件版本所使用的 API 级别，是创建脚本语言时所采用的最初的 API 级别。作者可以在任何时候手工更改 API 级别。

脚本语言的后向兼容性可以通过申请能够运行脚本语言的 TI-Nspire™软件的最旧版本所使用的 API 级别实现。脚本语言应当申请 **platform.apiLevel = '1.0'**以支持 3.1 发行版本。

脚本语言不一定能申请到所需的 API 级别，例如脚本语言运行在旧版本的 TI-Nspire™软件中，而这个版本并不支持所需的 API 级别。此外，在已经存在的 TI-Nspire™软件版本中申请一个不存在或者不支持的 API 级别，将会使用当前版本所支持的最高 API 级别。一个极端的例子是，在 TI-Nspire™软件 3.4 版本中申请 API 级别 0.1 会导致最终使用 API 级别 2.2。更多信息请参见表 A.1。参见 A.2 节的样例以了解如何申请一个不支持的 API 级别。

警告：在未来的软件发行版本中，申请一个不支持的 API 级别时的行为可能有所不同。可能会默认为低于所申请 API 级别的最高级别，而不是最高所支持的 API 级别。

表 A.1: API 级别与其对应的 TI-Nspire™软件版本

API 级别	软件版本	备注
'1.0'	3.1	最初支持 Lua 脚本语言的版本
'2.0'	3.2	重要更新，添加了物理库和其他新特性
'2.2'	3.4	为触摸平台提供了较低级别的支持

A.2 为未来的软件发行版本创建脚本语言

可能未来新发布的 TI-Nspire™软件采用更高的 API 级别，并且这个版本并不包含开发环境。在这种情况下，必须有条件的在运行脚本语言时使用更高 API 级别中的新函数。以下样例示范了如何在 OS 版本 3.2 下开发脚本语言时使用触摸库(touch library)。样例中的 Lua 代码片段必须在脚本语言的开头使用。在 **platform.apiLevel = '2.0'**中并没有定义触摸库，但是在所有未来发布的版本中都有。

```
platform.apiLevel = '2.2'
if touch  then
    if not  touch.enabled then
        function touch.enabled() return true end
        function touch.isKeyboardAvailable() return true end
     end
else
    touch  = {}
    function touch.enable() return false end
end
```

A.3 平台兼容性

脚本语言作者通常更喜欢在其他平台上为独立平台编写脚本语言。不幸的是，这种做法并不正确，因为每一种特性并不一定被所有平台支持。表 A.2 展示了主要的区别。脚本语言作者应当避免使用它们或仅在特定的平台上使用这些特性。也可以尝试实现全平台的用户无缝体验。在最后一种情况下，脚本语言作者应当在所有平台上测试脚本语言。

表 A.2: 平台兼容性概览

特性	桌面软件	手持设备	触摸平台
on.grabDown **on.grabUp**	支持，(x, y) == (0, 0)	支持，没有鼠标指针显示时 与桌面平台相同	不支持
on.returnKey()	不支持	支持	支持
右键菜单 (Context Menu）	**on.contextMenu()** **on.rightMouseDown()** **on.rightMouseUp()**	**on.contextMenu()**	不支持