Logitech G-series Lua API 参考文档

V8.50

预览与概述

目录

目录 2
OnEvent 3
GetMKeyState 5
SetMKeyState 6
Sleep 7
OutputLogMessage 8
ClearLog 9
GetRunningTime 10
GetDate 11
PressKey 12
ReleaseKey 13
PressAndReleaseKey 14
PressMouseButton 15
ReleaseMouseButton 16
PressAndReleaseMouseButton 17
IsMouseButtonPressed 18
IsModifierPressed 19
IsKeyLockOn 20
MoveMouseTo 21
MoveMouseWheel 22
MoveMouseRelative 23
MoveMouseToVirtual 24
GetMousePosition 25
OutputLCDMessage 26
ClearLCD 27
PlayMacro 28
AbortMacro 29
SetBacklightColor 30
OutputDebugMessage 31
SetMouseDPITable 32
SetMouseDPITableIndex 33
EnablePrimaryMouseButtonEvents 34
G13 可编程按键 35
标准 Lua 5.1 库 38
附录 A 39

OnEvent

OnEvent() 方法为脚本提供了一系列事件句柄以方便用户对触发的事件进行操作，您可以使用以下代码实现此方法。

```
function OnEvent(event, arg [, family])

end
```

参数列表

event

该字符串包含了用户所触发的事件名称。

arg

与事件标识符相对应的参数值。

family

触发硬件事件的设备族，如果您确定您的脚本不针对于特定硬件则将其留空即可。仅当您确定需要区分多个输入时使用此参数。

设备族	特定设备
“kb”	键盘设备 (G15, G11, G19, etc)
“lhc”	左手用控制器 (G13, etc)
“mouse”	受支持的鼠标 (G300, G400, etc)

返回值

无返回值

备注信息

下方列表是事件标识符以及其对应参数值：

事件名	参数值	描述信息
"PROFILE_ACTIVATED"	无	配置文件被激活时，此事件为脚本的第一个事件。
"PROFILE_DEACTIVATED"	无	配置文件被反激活时，此事件为脚本的最后一个事件。
"G_PRESSED"	1=G1 18=G18 n = G_n	G Key 按下时
"G_RELEASED"	1=G1 18=G18 n = G_n	G Key 释放时
"M_PRESSED"	1=M1 2=M2 3=M3	M Key 按下时
"M_RELEASED"	1=M1	M Key 释放时

	2=M2 3=M3	
“MOUSE_BUTTON_PRESSED”	2=鼠标按键 2 3=鼠标按键 3 4=鼠标按键 4 …	鼠标按键被按下时 注意：鼠标左键 (1) 默认将不会对此事件进行回应。您可以使用 ‘EnablePrimaryMouseButtonEvents’ 方法将其覆盖。
“MOUSE_BUTTON_RELEASED”	2=鼠标按键 2 3=鼠标按键 3 4=鼠标按键 4 ..	注意：鼠标左键 (1) 默认将不会对此事件进行回应。您可以使用 ‘EnablePrimaryMouseButtonEvents’ 方法将其覆盖。

代码示范

```
-- 此事件作为主事件句柄
-- 您必须首先实现该方法

function OnEvent(event, arg)
	if (event == "PROFILE_ACTIVATED") then
		-- 配置文件已被激活
	end

	if (event == "PROFILE_DEACTIVATED") then
		-- 配置文件已被反激活
	end

	if (event == "G_PRESSED" and arg == 1) then
		-- G1 已被按下
	end

	if (event == "G_RELEASED" and arg == 1) then
		-- G1 已被释放
	end

	if (event == "M_PRESSED" and arg == 1) then
		-- M1 已被按下
	end

	if (event == "M_RELEASED" and arg == 1) then
		-- M1 已被释放
	end

	if (event == "MOUSE_BUTTON_PRESSED" and arg == 6) then
		-- 鼠标按键 6 已被按下
	End

	if (event == "MOUSE_BUTTON_RELEASED" and arg == 6) then
		-- 鼠标按键 6 已被释放
	end
end
```

GetMKeyState

GetMKeyState() 返回当前 M Key 状态值。

```
mkey GetMKeyState  ([family]);
```

参数列表

family

当您希望区分多个输入设备时请使用该可选设备族属性，默认为 “kb”。

设备族	特定设备
“kb”	键盘设备 (G15, G11, G19, etc)
“lhc”	左手用控制器 (G13, etc)

返回值

mkey

1 = M1, 2 = M2, 3 = M3

备注信息

代码示范

```
-- 获取当前 M Key 状态

current_mkey = GetMKeyState()
```

SetMKeyState

SetMKeyState() 可以设置当前 M keys 激活状态。请注意：如果您紧接着该方法调用 **GetMKeyState** 方法将返回上次状态码。使用 OnEvent 句柄以确定当前操作是否完成。

```
mkey SetMKeyState  (mkey, [family]);
```

参数列表

mkey

1 = M1, 2 = M2, 3 = M3

family

当您希望区分多个输入设备时请使用该可选设备族属性，默认为 “kb”。

设备族	特定设备
“kb”	键盘设备 (G15, G11, G19, etc)
“lhc”	左手用控制器 (G13, etc)

返回值

无返回值

备注信息

代码示范

```
-- 当 G1 被按下时设置当前 M Key 状态为 M1

function OnEvent(event, arg)
      if (event == "G_PRESSED" and arg == 1) then
            SetMkeyState(1);
      end
end
```

Sleep

Sleep() 方法将导致脚本暂停并等待所设置的时间后再继续执行。

```
Sleep ( timeout );
```

参数列表

timeout

以毫秒为单位的睡眠时间。

返回值

nil

备注信息

脚本运行于主程序性能分析器不同的线程，所以脚本暂停时不会对其产生影响。

您可以使用此功能模拟操作延时。

性能分析器将等待一秒以使脚本执行完毕，超过该时间脚本将被强制中断。请务必注意不要进行长时间的延时。

代码示范

```
-- 睡眠 20 毫秒

Sleep(20)
```

OutputLogMessage

OutputLogMessage() 将输出日志消息至脚本编辑器的控制台操作窗中。

```
OutputLogMessage    ( ... );
```

参数列表

message

输出样式、包含格式化字符串的消息文本。

返回值

nil

备注信息

string.format() 方法镜像。

您需要手动在末尾插入回车符 "\n" 以结束该段文本。

代码示范

```
-- 输出文本 "Hello World"

OutputLogMessage("Hello World %d\n", 2007)
```

ClearLog

ClearLog() 方法将清空脚本编辑器控制台中的输出内容。

```
ClearLog ()
```

参数列表

无

返回值

nil

备注信息

无

代码示范

```
-- 清空脚本编辑器日志

OutputLogMessage("This message will self destruct in 2 seconds\n")
Sleep(2000)
ClearLog()
```

GetRunningTime

GetRunningTime() 方法将返回以毫秒为单位的执行脚本总时间。

```
elapsed GetRunningTime   ();
```

参数列表

无

返回值

elapsed

一个包含运行总毫秒时间的整数值。

备注信息

您可以使用此方法计算您脚本的运行时间。.

代码示范

```
-- 显示脚本运行时间

OutputLogMessage("This script has been running for: %d ms",
GetRunningTime())
```

GetDate

GetDate() 方法将获取已格式化的当前时间

```
date GetDate  ([format [, time]])
```

参数列表

format

可选的日期字符串格式。

time

可选的时间表。

返回值

date

返回一个包含用户计算机当前日期以及时间（或参考时间）的字符串或时间表，根据所给的字符串格式就行字符串格式化，如果您希望提供自有格式化字符串，该方法将使用与 strftime() 相同的解析规则。特殊字符串 *t 将使 date() 方法返回一个时间表。

备注信息

os.date() 方法镜像。

代码示范

```
-- 显示当前日期与时间

OutputLogMessage("Today’ s date/time is: %s       \n", GetDate())
```

PressKey

PressKey() 方法可被用于模拟键盘按键按下动作。请注意：如果您紧接着该方法**调用** **IsModifierPressed** 方法或 **IsKeyLockOn** 方法将返回上次状态码。您需要等待几毫秒使其结束操作以确定当前操作是否完成。

```
PressKey( scancode [,scancode] );

PressKey( keyname [,keyname] );
```

参数列表

scancode

特定数值扫描码对应的按键将被按下。

keyname

特定预定义键值对应的按键将被按下。

返回值

nil

备注信息

如果提供了多个按键作为实际参数，所有按键将被模拟为同一时间按下。关于扫描码或键值对应列表您可以参考附录 A。

代码示范

```
-- 使用扫描码模拟键值 "a" 被按下
PressKey(30)

-- 使用键值模拟键值 "a" 被按下
PressKey("a")

-- 模拟 "a" 与 "b" 同时按下
PressKey("a", "b")
```

ReleaseKey

ReleaseKey() 方法可被用于模拟键盘按键释放动作。

```
ReleaseKey( scancode [,scancode] );

ReleaseKey( keyname [,keyname] );
```

参数列表

scancode

特定数值扫描码对应的按键将被释放。

keyname

特定数值键值对应的按键将被释放。

返回值

nil

备注信息

如果提供了多个按键作为实际参数，所有按键将被模拟为同一时间释放。关于扫描码或键值对应列表您可以参考附录 A。

代码示范

```
-- 使用扫描码模拟键值 "a" 被释放
ReleaseKey(30)

-- 使用键值模拟键值 "a" 被释放
ReleaseKey("a")

-- 模拟 "a" 与 "b" 同时释放
ReleaseKey("a", "b")
```

PressAndReleaseKey

PressAndReleaseKey() 方法可被用于模拟键盘按键按下动作并跟随按键释放动作。请注意：如果您紧接着该方法调用 **IsModifierPressed** 方法或 **IsKeyLockOn** 方法将返回上次状态码。您需要等待几毫秒使其结束操作以确定当前操作是否完成。

```
PressAndReleaseKey ( scancode [,scancode] );

PressAndReleaseKey ( keyname [,keyname] );
```

Parameters

scancode

特定数值扫描码对应的按键将被按下并随后被释放。

keyname

特定数值键值对应的按键将被按下并随后被释放。

返回值

nil

备注信息

如果提供了多个按键作为实际参数，所有按键将被模拟为同一时间按下并随后释放。关于扫描码或键值对应列表您可以参考附录 A。

代码示范

```
-- 使用扫描码模拟键值 "a" 被按下并随后被释放
PressAndReleaseKey(30)

-- 使用键值模拟键值 "a" 被按下并随后被释放
PressAndReleaseKey("a")

-- 模拟 "a" 与 "b" 同时按下并随后释放
PressAndReleaseKey("a", "b")
```

PressMouseButton

PressMouseButton() 方法可被用于模拟鼠标按键被按下。请注意： **如果您紧接着该方法调用 IsMouseButtonPressed** 方法将返回上次状态码。您需要等待几毫秒使其结束操作以确定当前操作是否完成。

```
PressMouseButton    ( button )
```

参数列表

button

按键标识符，您可以使用下表中列出的值：

按键值	对应操作
1	鼠标左键
2	鼠标中键
3	鼠标右键
4	鼠标按键 X1
5	鼠标按键 X2

返回值

nil

备注信息

无

代码示范

```
-- 模拟鼠标左键被按下
PressMouseButton(1)

-- 模拟鼠标右键被按下
PressMouseButton(3)
```

ReleaseMouseButton

ReleaseMouseButton() 方法可被用于模拟鼠标按键被释放。

```
ReleaseMouseButton    ( button )
```

参数列表

button

按键标识符，您可以使用下表中列出的值：

按键值	对应操作
1	鼠标左键
2	鼠标中键
3	鼠标右键
4	鼠标按键 X1
5	鼠标按键 X2

返回值

nil

备注信息

无

代码示范

```
-- 模拟鼠标左键单击（按下并释放）

PressMouseButton(1)
ReleaseMouseButton(1)
```

PressAndReleaseMouseButton

PressAndReleaseMouseButton()方法可被用于模拟鼠标按键按下动作并跟随按键释放动作。请注意：如果您紧接着该方法**调用 IsMouseButtonPressed** 方法将返回上次状态码。您需要等待几毫秒使其结束操作以确定当前操作是否完成。

```
PressAndReleaseMouseButton      ( button )
```

参数列表

button

按键标识符，您可以使用下表中列出的值：

按键值	对应操作
1	鼠标左键
2	鼠标中键
3	鼠标右键
4	鼠标按键 X1
5	鼠标按键 X2

返回值

nil

备注信息

无

代码示范

```
-- 模拟鼠标左键单击（按下并释放）

PressAndReleaseMouseButton(1)
```

IsMouseButtonPressed

IsMouseButtonPressed() 方法可用于确定某鼠标按键是否被按下。

```
boolean IsMouseButtonPressed    ( button )
```

参数列表

button

按键标识符，您可以使用下表中列出的值：

按键值	对应操作
1	鼠标左键
2	鼠标中键
3	鼠标右键
4	鼠标按键 X1
5	鼠标按键 X2

返回值

当修饰键被按下时将返回 true 反之则返回 false。

备注信息

无

代码示范

```
-- 按下鼠标按键
PressMouseButton(1)

if IsMouseButtonPressed(1) then
    OutputLogMessage("Left mouse button is pressed.\n");
end

-- 释放该鼠标按键
ReleaseMouseButton(1)

if not IsMouseButtonPressed(1) then
    OutputLogMessage("Left mouse button is not pressed.\n");
end
```

IsModifierPressed

IsModifierPressed() 方法可用于确定某修饰键是否被按下。

```
boolean IsModifierPressed    ( keyname );
```

参数列表

keyname

特定预定义的**修饰键**键值是否被按下，该参数必须为下列表值之一：

修饰键	描述信息
"lalt", "ralt", "alt"	左侧、右侧或两侧 Alt 键
"lshift", "rshift", "shift"	左侧、右侧或两侧 Shift 键
"lctrl", "rctrl", "ctrl"	左侧、右侧或两侧 Ctrl 键

返回值

当修饰键被按下时将返回 true 反之则返回 false。

备注信息

无

代码示范

```
-- 模拟按下特定修饰键
PressKey("lshift")

if IsModifierPressed("shift") then
    OutputLogMessage("shift is pressed.\n");
end

-- 释放该修饰键
ReleaseKey("lshift")

if not IsModifierPressed("shift") then
    OutputLogMessage("shift is not pressed.\n");
end
```

IsKeyLockOn

IsKeyLockOn() 方法可用于确定锁定键是否处于启用状态。

```
IsKeyLockOn  ( key )
```

参数列表

key

键值名可参考下表：

键值	对应操作
"scrolllock"	滚动锁定
"capslock"	大小写锁定
"numlock"	数字键锁定

返回值

当锁定键启用时将返回 true 反之则返回 false。

备注信息

无

代码示范

```
-- 检查数字键锁定是否被启用，如果启用则将其关闭
if ( IsKeyLockOn("numlock" ) then
      PressAndReleaseKey("numlock");
end
```

MoveMouseTo

MoveMouseTo() 方法可被用于移动鼠标指针至屏幕中的目标绝对坐标位置。请注意：如果您紧接着该方法**调用 GetMousePosition** 方法将返回上次状态码。您需要等待几毫秒使其结束操作以确定当前操作是否完成。

```
MoveMouseTo  ( x, y, )
```

参数列表

x

标准 x 极坐标介于 0（极左）至 65535（极右）。

Y

标准 y 极坐标介于 0（极左）至 65535（极右）。

返回值

nil

备注信息

如果您接入了多个显示器请使用 **MoveMouseToVirtual** 方法代替。

代码示范

```
-- 移动鼠标至左上角
MoveMouseTo(0, 0)

-- 移动鼠标至屏幕中央
MoveMouseTo(32767, 32767)

-- 移动鼠标至右下角
MoveMouseTo(65535, 65535)
```

MoveMouseWheel

MoveMouseWheel() 方法可被用于模拟鼠标滚轮滚动。

```
MoveMouseWheel   ( click )
```

Parameters

click

鼠标滚轮滚动次数。

返回值

nil

备注信息

如果您输入的为正数，那么将模拟向上滚动（远离用户），如果您输入

的为负数，那么将模拟向下滚动（面向用户）。

代码示范

```
-- 模拟鼠标滚轮向上滚动 3 次
MoveMouseWheel(3)

-- 模拟鼠标滚轮向下滚动 1 次
MoveMouseWheel(-1)
```

MoveMouseRelative

MoveMouseRelative() 方法可被用于模拟鼠标相对当前坐标的偏移量。请注意：如果您紧接着该方法调用 **GetMousePosition** 方法将返回上次状态码。您需要等待几毫秒使其结束操作以确定当前操作是否完成。

```
MoveMouseRelative    ( x, y, )
```

参数列表

x

沿 x 轴方向移动

Y

沿 y 轴方向移动

返回值

nil

备注信息

如果您输入的 x 轴偏移量为正数，那么将模拟向右移动，如果您输入的 x 轴偏移量为负数，那么将模拟向左移动。如果您输入的 y 轴偏移量为正数，那么将模拟向下移动，如果您输入的 y 轴偏移量为负数，那么将模拟向上移动。

代码示范

```
-- 模拟鼠标相对当前坐标移动偏移量为 1 像素并重复 50 次
for i = 0, 50 do
    MoveMouseRelative(0, -1)
    Sleep(8)
end
```

MoveMouseToVirtual

MoveMouseToVirtual() 方法可被用于在多个屏幕中移动鼠标指针至当前屏幕中的目标绝对坐标位置。请注意：如果您紧接着该方法调用 **GetMousePosition** 方法将返回上次状态码。您需要等待几毫秒使其结束操作以确定当前操作是否完成。

```
MoveMouseToVirtual    ( x, y, )
```

Parameters

x

标准 x 极坐标介于 0（极左）至 65535（极右）。

Y

标准 y 极坐标介于 0（极左）至 65535（极右）。

返回值

nil

备注信息

如果您接入了多个显示器请使用 **MoveMouseToVirtual** 方法代替。

代码示范

```
-- 移动鼠标至虚拟桌面中的左上角
MoveMouseToVirtual(0, 0)

-- 移动鼠标至虚拟桌面中的中央
MoveMouseToVirtual (32767, 32767)

-- 移动鼠标至虚拟桌面中的右下角
MoveMouseToVirtual (65535, 65535)
```

GetMousePosition

GetMousePosition() 方法可被用于获取鼠标指针当前相对标准坐标。

```
x,y GetMousePosition    ( )
```

参数列表

无

返回值

x

标准 x 极坐标介于 0（极左）至 65535（极右）。

Y

标准 y 极坐标介于 0（极左）至 65535（极右）。

备注信息

代码示例

```
-- 获取当前鼠标指针坐标
x, y = GetMousePosition();

OutputLogMessage("Mouse is at %d, %d\n", x, y);
```

OutputLCDMessage

OutputLCDMessage() 方法可被用于向 LCD 添加单行文本。

```
OutputLCDMessage    ( text [,timeout] )
```

参数列表

text

待显示字符串

timeout

超时时间为毫秒，等待超时后该消息将被删除。

返回值

nil

备注信息

您可以一次向设备添加并显示至多四条文本信息，默认超时时间为 1 秒。

代码示范

```
-- 使用默认超时显示文本
OutputLCDMessage("Hello world")

-- 显示文本并设置 2 秒超时时间
OutputLCDMessage("Hello world", 2000)
```

ClearLCD

ClearLCD() 方法可用于清除由脚本输出在 LED 中的字符串。

```
ClearLCD （ ）
```

参数列表

无

返回值

nil

备注信息

代码示范

```
-- 清理 LED 已显示内容并输出两条文本信息
ClearLCD ()
OutputLCDMessage("Hello world1")
OutputLCDMessage("Hello world2")
```

PlayMacro

PlayMacro() 方法可被用于播放已存在宏脚本。

```
PlayMacro  ( macroname )
```

参数列表

macroname

您需要指定在当前配置文件中已存在的宏脚本名称。

返回值

nil

备注信息

如果该方法在其他宏脚本播放时被调用则不会采取任何操作。换而言之，在同一段时间内仅可播放一个宏脚本而无法同时播放数个宏脚本。

然而如果调用并播放相同的一个宏脚本那么该宏脚本将被排入队列并在稍后播放。

代码示范

```
-- 播放已存在宏脚本
PlayMacro("my macro");
```

AbortMacro

AbortMacro() 方法可被用于中断当前任何已在播放的宏脚本。

```
AbortMacro （ ）
```

参数列表

无

返回值

nil

备注信息

在调用 PlayMacro 方法时播放宏脚本过程中按下的所有按键将被释放，然而由外部播放的宏脚本仍将继续运行而不受影响。

代码示范

```
-- 播放宏脚本
PlayMacro("my macro")

-- 等待 100 毫秒并将正在播放的宏脚本中断
AbortMacro()
```

SetBacklightColor

SetBacklightColor() 方法被用于设置自定义设备背光灯颜色（如果您的设备支持自定义背光灯）。

```
SetBacklightColor   (red, green, blue, [family])
```

参数列表

red

红色阈值范围 (0 – 255)。

green

绿色阈值范围 (0 – 255)。

blue

蓝色阈值范围 (0 – 255)。

family

当您希望区分多个输入设备时请使用该可选设备族属性，默认为 “kb”。

设备族	特定设备
“kb”	键盘设备 (G15, G11, G19, etc)
“lhc”	左手用控制器 (G13, etc)

返回值

nil

备注信息

无

代码示范

```
-- 设置背光灯为红色
SetBacklightColor(255, 0, 0);

-- 为所有左手用控制器设备设置背光灯为蓝色
SetBacklightColor(0, 0, 255, “lhc”);
```

OutputDebugMessage

OutputDebugMessage() 方法将发送日志消息至 Windows 脚本调试器。

```
OutputDebugMessage    ( ... );
```

参数列表

Message

输出样式、包含格式化字符串的消息文本。

返回值

nil

备注信息

string.format() 方法镜像。

您需要手动在末尾插入回车符 "\n" 以结束该段文本。您可以使用类似

于 Dbg 查看器等工具查看这些输出的消息。

代码示范

```
-- 输出文本 "Hello World"

OutputDebugMessage("Hello World %d\n", 2007)
```

SetMouseDPITable

SetMouseDPITable() 方法可向已支持的游戏鼠标设置当前 DPI 表中的数值。

```
SetMouseDPITable    ({value1, value2, value3}, [index]       );
```

参数列表

DPI Array

DPI 数值数组

DPI Index

可选并从 1 开始的 DPI 索引值以直接应用该 DPI 数值。

返回值

nil

备注信息

如果您不指定索引值，那么将使用首条 DPI 数值设为当前值，您最多

可以添加 16 条数值。

当您激活新配置文件时将使用该 DPI 设置覆盖上次状态。

代码示范

```
-- 设置 DPI 数值为 {500, 1000, 1500, 2000, 2500}
-- 默认状态下，500 DPI 将被设为当前 DPI 数值
SetMouseDPITable({500, 1000, 1500, 2000, 2500})

-- 设置 DPI 数值为 {500, 2500} 并设置第二个数值为当前 DPI 数值
SetMouseDPITable({500, 2500}, 2)
```

SetMouseDPITableIndex

SetMouseDPITableIndex() 方法可向已支持的游戏鼠标通过 DPI 表索引值设置 DPI。

```
SetMouseDPITableIndex    (index);
```

参数列表

Index

基于 1 开始的 DPI 表索引值。

返回值

nil

备注信息

如果未曾调用 **SetMouseDPITable** 方法则将使用鼠标中现有 DPI 表。您最多可以添加 16 条数值。

当您激活新配置文件时将使用该 DPI 设置覆盖上次状态。

代码示范

```
-- 设置初始 DPI数值为 {500, 1000, 1500, 2000, 2500}
SetMouseDPITable({500, 1000, 1500, 2000, 2500})

-- 设置当前 DPI 为表中第三项 (1500 DPI)
SetMouseDPITableIndex(3);
```

EnablePrimaryMouseButtonEvents

EnablePrimaryMouseButtonEvents() 方法将启用鼠标按键 1 的事件报告。

```
EnablePrimaryMouseButtonEvents      (enable);
```

参数列表

enable

1 或 true 以启用鼠标按键 1 事件报告

0 或 false 以禁用鼠标按键 1 事件报告

返回值

nil

备注信息

出于性能上的原因考虑，鼠标左键按键默认将不对触发事件进行报告。

代码示范

```
-- 启用鼠标按键 1 事件报告
EnablePrimaryMouseButtonEvents(true);

-- 禁用鼠标按键 1 事件报告
EnablePrimaryMouseButtonEvents(false);
```

G13 可编程按键

G13 游戏面板具有一个可以模拟鼠标的摇杆，您可以为其分配一些鼠标功能。 您可以通过在设置窗口中的配置文件选项面板或通过 Lua 脚本语言控制由摇杆所模拟鼠标的移动速度。以下实例可帮助您使用 Lua 方法控制鼠标运行状态：

SetMouseSpeed ()

参数列表

New mouse speed

鼠标绝对移动速度由 32 至 255。

返回值

nil

备注信息

无

代码示范

```
-- 设置鼠标移动速度至 128

SetMouseSpeed(128);
```

GetMouseSpeed()

参数列表

Current mouse speed

当前鼠标的绝对移动速度。

返回值

当前模拟鼠标的移动速度

备注信息

无

代码示范

```
-- 获取鼠标移动速度

OutputLogMessage("The Mouse Speeed is: %d\n”, GetMouseSpeed());
```

IncrementMouseSpeed()

参数列表

Mouse speed increment

返回值

nil

备注信息

加快鼠标速度最大值将被限制至 255。

代码示范

```
-- 增加鼠标移动速度是 10

IncrementMouseSpeed(10);
```

DecrementMouseSpeed()

参数列表

Mouse speed decrement

返回值

nil

备注信息

降低鼠标速度最小值将被限制至 32。

代码示范

```
-- 降低鼠标移动速度至 10

DecrementMouseSpeed(10);
```

G13 鼠标不具备任何原生按键，例如左键、中键以及右键等。鼠标按键必须通过 Lua 进行编程。您可以通过这个示例了解如何使用通用 Lua 代码实现鼠标按键触发操作：

```
if event=="G_PRESSED" and arg==x then
    PressMouseButton( y );
end

if event=="G_RELEASED" and arg==x then
    ReleaseMouseButton( y );
```

标准 Lua 5.1 库

以下函数为受支持的标准库函数：

- string.byte
- string.char
- string.dump
- string.find
- string.format
- string.gmatch
- string.gsub
- string.len
- string.lower
- string.match
- string.rep
- string.reverse
- string.sub
- string.upper
- table.concat
- table.insert
- table.maxn
- table.remove
- table.sort
- math.abs
- math.acos
- math.asin
- math.atan
- math.atan2
- math.ceil
- math.cos
- math.deg
- math.exp
- math.floor
- math.fmod
- math.frexp
- math.huge
- math.ldexp
- math.log
- math.log10
- math.max
- math.min
- math.modf
- math.pi
- math.pow
- math.rad
- math.random
- math.randomseed
- math.sin
- math.sinh
- math.sqrt
- math.tan
- math.tanh

附录 A

该表中所有键值以及扫描值可被用于 **PressKey()**, **ReleaseKey()**, **IsModifierPressed()** 代码。

键值	扫描码（十六进制）	扫描码（十进制）
"escape"	0x01	1
"f1"	0x3b	59
"f2"	0x3c	60
"f3"	0x3d	61
"f4"	0x3e	62
"f5"	0x3f	63
"f6"	0x40	64
"f7"	0x41	65
"f8"	0x42	66
"f9"	0x43	67
"f10"	0x44	68
"f11"	0x57	87
"f12"	0x58	88
"f13"	0x64	100
"f14"	0x65	101
"f15"	0x66	102
"f16"	0x67	103
"f17"	0x68	104
"f18"	0x69	105
"f19"	0x6a	106
"f20"	0x6b	107
"f21"	0x6c	108
"f22"	0x6d	109
"f23"	0x6e	110
"f24"	0x76	118
"printscreen"	0x137	311
"scrolllock"	0x46	70
"pause"	0x146	326
"tilde"	0x29	41
"1"	0x02	2
"2"	0x03	3
"3"	0x04	4
"4"	0x05	5
"5"	0x06	6
"6"	0x07	7
"7"	0x08	8
"8"	0x09	9
"9"	0x0a	10
"0"	0x0b	11
"minus"	0x0c	12
"equal"	0x0d	13
"backspace"	0x0e	14
"tab"	0x0f	15
"q"	0x10	16
"w"	0x11	17
"e"	0x12	18
"r"	0x13	19
"t"	0x14	20
"y"	0x15	21

"u"	0x16	22
"l"	0x17	23
"o"	0x18	24
"p"	0x19	25
"lbracket"	0x1a	26
"rbracket"	0x1b	27
"backslash"	0x2b	43
"capslock"	0x3a	58
"a"	0x1e	30
"s"	0x1f	31
"d"	0x20	32
"f"	0x21	33
"g"	0x22	34
"h"	0x23	35
"j"	0x24	36
"k"	0x25	37
"l"	0x26	38
"semicolon"	0x27	39
"quote"	0x28	40
"enter"	0x1c	28
"lshift"	0x2a	42
"non_us_slash"	0x56	86
"z"	0x2c	44
"x"	0x2d	45
"c"	0x2e	46
"v"	0x2f	47
"b"	0x30	48
"n"	0x31	49
"m"	0x32	50
"comma"	0x33	51
"period"	0x34	52
"slash"	0x35	53
"rshift"	0x36	54
"lctrl"	0x1d	29
"lgui"	0x15b	347
"lalt"	0x38	56
"spacebar"	0x39	57
"ralt"	0x138	312
"rgui"	0x15c	348
"appkey"	0x15d	349
"rctrl"	0x11d	285
"insert"	0x152	338
"home"	0x147	327
"pageup"	0x149	329
"delete"	0x153	339
"end"	0x14f	335
"pagedown"	0x151	337
"up"	0x148	328
"left"	0x14b	331
"down"	0x150	336
"right"	0x14d	333
"numlock"	0x45	69
"numslash"	0x135	309

"numminus"	0x4a	74
"num7"	0x47	71
"num8"	0x48	72
"num9"	0x49	73
"numplus"	0x4e	78
"num4"	0x4b	75
"num5"	0x4c	76
"num6"	0x4d	77
"num1"	0x4f	79
"num2"	0x50	80
"num3"	0x51	81
"numenter"	0x11c	284
"num0"	0x52	82
"numperiod"	0x53	83
“calculator”	0x121	289
“web”	0x132	306
“media_prev_track”	0x110	272
“media_next_track”	0x119	281
“media_play_pause”	0x122	290
“media_stop”	0x124	292
“volume_up”	0x130	304
“volume_down”	0x12e	302
“volume_mute”	0x120	288