luaSTG+ lua层API手册

文档约定

符号

- **[新]**： 表示该方法相对于基准代码增加了新的功能并保有兼容性
- **[新增]**： 表示该方法为新增的方法，不存在于基准代码中
- **[否决]**： 表示该方法可能在后续版本中移除，不再被维护
- **[移除]**： 表示该方法被移除，与基准代码不兼容
- **[改]**： 表示该方法相对于基准代码有所修改但保有兼容性
- **[不兼容]**： 表示该方法相对基准代码有所修改且不兼容
- **[补]**： 表示方法为基准代码中缺失的，不保证完全的兼容性

框架执行流程

- luaSTGPlus初始化
- 初始化lua引擎
- 装载`launch`文件执行初始化配置、装载资源包等
- 初始化fancy2d引擎框架
- 装载文件`core.lua`
- 执行lua全局函数`GameInit`
- 启动并执行游戏循环
- 销毁框架并退出

编码

- 程序将使用**UTF-8**作为lua代码的编码，如果lua端使用非UTF-8编码将在运行时导致乱码 **[不兼容]**
- 程序将使用**UTF-8**作为资源包的编码，这意味着如果资源包中出现非UTF-8编码的字符将导致无法定位文件 **[不兼容]**

内建变量

- args:table **[不兼容]**

  保存命令行参数

  细节

  `luastg`中该变量名称为`arg`，与老式可变参列表名称`arg`冲突。由于`luajit`不再支持使用`arg`访问可变参列表，为了减少编码错误，故将`arg`更名为`lstg.args`。这可能会影响到`launch`中的代码。

内建类

lstgColor

lstgColor用于表示一个基于a,r,g,b四分量的32位颜色

方法

- ARGB(lstgColor):number, number, number, number

  返回颜色的a,r,g,b分量

元方法

- __eq(lstgColor, lstgColor):boolean

判断两个颜色值是否相等

- __add(lstgColor, lstgColor):lstgColor

  将两个颜色值相加，超出部分置为255

- __sub(lstgColor, lstgColor):lstgColor **[新增]**

  将两个颜色值相减，下溢部分置为0

- __mul(lstgColor|number, lstgColor|number):lstgColor **[新]**

  将两个颜色值的各个分量相乘或与一个数字相乘，超出部分置为255

- __tostring(lstgColor):string

  打印类名，该值为`lstg.Color(a,r,g,b)`其中argb为颜色分量值

lstgRand

基于WELL512算法的随机数发生器，初始化时将使用系统tick计数作为种子 **[改]**

```
细节
    luaSTG实现中采用线性同余(?)方法产生随机数，luaSTG+采用WELL512算法产生随机数（该算法摘自《游戏编程精粹 7》）

潜在兼容性问题
    luaSTG实现中默认使用种子(0)初始化生成器
```

方法

- Seed(lstgRand, number)

  设置随机数种子（注意：number值不应该超过32bit无符号整数范围）

- GetSeed(lstgRand) **[新增]**

  返回随机数种子

- Int(lstgRand, min:number, max:number):number

  产生一个在[min,max]区间内的整型随机数，请确保min<=max

- Float(lstgRand, min:number, max:number):number

  产生一个在[min,max]区间内的浮点型随机数，请确保min<=max

- Sign(lstgRand):number

  随机返回±1

元方法

- __tostring(lstgRand):string

  打印类名，该值始终为`lstg.Rand object`

lstgBentLaserData

该类用于控制内建的曲线激光。

```
细节
    luastg+中至多允许出现1024个激光，每个激光至多捕获512个节点。
    曲线激光目前在实现上仍有问题，待日后优化。
```

方法

- Update(lstgBentLaserData, base_obj, length, width)

  传入基础对象，获取基础对象的坐标信息构造下一个激光位置。length和width决定激光的节点数和节点宽度。节点宽度同时会影响碰撞。

- Release(lstgBentLaserData)

  回收对象。（当前实现中无作用，对象交由GC回收）

- Render(lstgBentLaserData, texture:string, blend:string, color:lstgColor, tex_left:number, tex_top:number, tex_weight:number, tex_height:number, [scale:number=1]) **[新]**

  渲染曲线激光。参数为纹理、混合模式、颜色以及激光图像在纹理中的位置。scale用于控制激光纵向的缩放。

  注意，始终以纹理宽度方向作为激光的伸缩方向。

  该函数受全局缩放系数影响。

- CollisionCheck(lstgBentLaserData, x:number, y:number, [rot:number=0, a:number=0, b:number=0, rect:boolean=false]):boolean **[新]**

  检查当前对象和位于(x,y)的对象是否发生碰撞。

- BoundCheck(lstgBentLaserData):boolean

  检查当前对象是否在边界内。若越界则返回false。

内建方法

所有内建方法归类于lstg全局表中。

框架控制方法

- SetWindowed(boolean) **[不兼容]**

  设置窗口化(true)/非窗口化(false)。默认为true。

  仅限初始化中使用，不允许在运行时切换窗口模式。

- SetFPS(number) **[不兼容]**

  设置FPS锁定值。默认为60FPS。

  仅限初始化中使用，不允许在运行时动态设置FPS。

- GetFPS():number

  获得当前的实时FPS。

- SetVsync(boolean) **[不兼容]**

  设置是否垂直同步。默认为true。

  仅限初始化中使用，不允许在运行时动态设置垂直同步。

- SetResolution(width:number, height:number)

  设置分辨率。默认为640x480。

  仅限初始化中使用，不允许在运行时动态设置分辨率。

- ChangeVideoMode(width:number, height:number, windowed:boolean, vsync:boolean):boolean **[新增]**

  改变视频选项。若成功返回true，否则返回false并恢复到上次的视频模式。

  仅限运行时使用

- SetSplash(boolean)

设置是否显示光标。默认为false。

- SetTitle(string)

  设置窗口标题。默认为"LuaSTGPlus"。

- SystemLog(string)

  写出日志。

- Print(...)

  将若干值写到日志。

- LoadPack(path:string, [password:string]) **[新]**

  加载指定位置的ZIP资源包，可选填密码。

  失败将导致错误。

  > 细节
  > 后加载的资源包有较高的查找优先级。这意味着可以通过该机制加载资源包来覆盖基础资源包中的文件。
  > 一旦zip文件被打开，将不能被访问。
  > 加载文件时将按照优先级依次搜索资源包，若资源包中不含文件则从当前目录加载。

- UnloadPack(path:string)

  卸载指定位置的资源包，要求路径名必须一致。

  若资源包不存在不发生错误。

- ExtractRes(path:string, target:string) **[补]**

  将资源包中的数据解压到本地。

  若失败将抛出错误。

- DoFile(path:string)

  执行指定路径的脚本。已执行过的脚本会再次执行。

  若文件不存在、编译失败、执行失败则抛出错误。

- ShowSplashWindow([path:string]) **[新增]**

  装载载入窗口。参数为图片路径。

  若图片加载失败或为空则使用内置的图片打开窗口。

对象池管理方法

- GetnObj():number

  获取对象池中对象个数。

- UpdateObjList() **[否决]**

  更新对象池。此时将所有对象排序并归类。

  排序规则：uid越小越靠前

  > 细节
  > luaSTG+中该函数不再起任何作用，对象表总是保持有序的。

- ObjFrame()

更新对象列表中所有对象，并更新属性。

禁止在协程上调用该方法。

```
细节
    按照下列顺序更新这些属性：
        vx += ax
        vy += ay
        x += vx
        y += vy
        rot += omiga
        更新绑定的粒子系统（若有）
```

- ObjRender()

  渲染所有对象。此时将所有对象排序。

  禁止在协程上调用该方法。

  排序规则：layer小的先渲染，若layer相同则按照uid

  ```
  细节
      luaSTG+中渲染列表总是保持有序的，将不会每次排序。
  ```

- SetBound(left:number, right:number, bottom:number, top:number)

  设置舞台边界。

- BoundCheck()

  执行边界检查。注意BoundCheck只保证对象中心还在范围内，不进行碰撞盒检查。

  禁止在协程上调用该方法。

- CollisionCheck(A:groupid, B:groupid)

  对组A和B进行碰撞检测。如果组A中对象与组B中对象发生碰撞，将执行A中对象的碰撞回调函数。

  禁止在协程上调用该方法。

- UpdateXY()

  刷新对象的dx,dy,lastx,lasty,rot（若navi=true）值。

  禁止在协程上调用该方法。

- AfterFrame()

  刷新对象的timer和ani_timer，若对象被标记为del或kill将删除对象并回收资源。

  禁止在协程上调用该方法。

  ```
  细节
      对象只有在AfterFrame调用后才会被清理，在此之前可以通过设置对象的status字段取消删除标记。
  ```

- New(class)

  创建新对象。将累加uid值。

  ```
  细节
      该方法使用class创建一个对象，并在构造对象后调用class的构造方法构造对象。
      被创建的对象具有如下属性：
          x, y            坐标
          dx, dy          (只读)距离上一次更新的坐标增量
          rot             角度
  ```

```
    omiga           角度增量
    timer           计数器
    vx, vy          速度
    ax, ay          加速度
    layer           渲染层级
    group           碰撞组
    hide            是否隐藏
    bound           是否越界销毁
    navi            是否自动更新朝向
    colli           是否允许碰撞
    status          对象状态，返回del kill normal
    hscale, vscale  横向、纵向的缩放程度
    class           对象的父类
    a, b            碰撞盒大小
    rect            是否为矩形碰撞盒
    img
    ani             (只读)动画计数器
被创建对象的索引1和2被用于存放类和id【请勿修改】

其中父类class需满足如下形式：
    is_class = true
    [1] = 初始化函数 (object, ...)
    [2] = 删除函数(DEL) (object, ...) [新]
    [3] = 帧函数 (object)
    [4] = 渲染函数 (object)
    [5] = 碰撞函数 (object, object)
    [6] = 消亡函数(KILL) (object, ...) [新]
上述回调函数将在对象触发相应事件时被调用

luastg+提供了至多32768个空间共object使用。超过这个大小后将报错。
```

- Del(object, [...]) **[新]**

  通知删除一个对象。将设置标志并调用回调函数。

  若在object后传递多个参数，将被传递给回调函数。

- Kill(object, [...]) **[新]**

  通知杀死一个对象。将设置标志并调用回调函数。

  若在object后传递多个参数，将被传递给回调函数。

- IsValid(object)

  检查对象是否有效。

- GetV(object):number, number **[新增]**

  获取对象的速度，依次返回速度大小和速度方向。

- SetV(object, v:number, a:number, track:boolean)

  以<速度大小,角度>设置对象的速度，若track为true将同时设置r。

- SetImgState(object, blend:string, a:number, r:number, g:number, b:number)

  设置资源状态。blend指示混合模式（含义见后文）a,r,g,b指定颜色。

  该函数将会设置和对象绑定的精灵、动画资源的混合模式，该设置对所有同名资源都有效果。

- Angle(a:object | x1:number, b:object | y1:number, [x2:number, y2:number]):number

  若a,b为对象，则求向量(对象b.中心 - 对象a.中心)相对x轴正方向的夹角。否则计算tan2(y2-y1, x2-x1)。

- Dist(a:object|number, b:object|number, [c:number, d:number]):number

  求距离。若a与b为对象则计算a与b之间的距离。否则计算向量(c,d)与(a,b)之间的距离。

- BoxCheck(object, left:number, right:number, top:number, bottom:number):boolean

  检查对象中心是否在所给范围内。

- ResetPool()

  清空并回收所有对象。

- DefaultRenderFunc(object)

  在对象上调用默认渲染方法。

- NextObject(groupid:number, id:number):number, object **[不兼容]**

  获取组中的下一个元素。若groupid为无效的碰撞组则返回所有对象。

  返回的第一个参数为id（luastg中为idx），第二个参数为对象

  ```
  细节
      luastg中NextObject接受的第二个参数为组中的元素索引而非id。
      出于效率考虑，luastg+中接受id查询下一个元素并返回下一个元素的id。
  ```

- ObjList(groupid:number):NextObject, number, number **[不兼容]**

  产生组遍历迭代器。

  ```
  细节
      由于NextObject行为发生变更，ObjList只在for循环中使用时可以获得兼容性。
  ```

- ParticleFire(object)

  启动绑定在对象上的粒子发射器。

- ParticleStop(object)

  停止绑定在对象上的粒子发射器。

- ParticleGetn(object)

  返回绑定在对象上的粒子发射器的存活粒子数。

- ParticleGetEmission(object)

  获取绑定在对象上粒子发射器的发射密度。（个/秒）

  ```
  细节
      luastg/luastg+更新粒子发射器的时钟始终为1/60s。
  ```

- ParticleSetEmission(object, count)

  设置绑定在对象上粒子发射器的发射密度。（个/秒）

资源管理系统

luastg/luastg+提供了两个资源池：全局资源池、关卡资源池，用于存放不同用途的资源。

资源池使用字符串哈希表进行管理，一个池中的同种资源其名称不能重复。

所有的加载函数会根据当前的资源池类别加载到对应的池中。寻找资源时优先到关卡资源池中寻找，若没有再到全局资源池中寻找。

```
资源类型表
    1 纹理
    2 图像
    3 动画
```

```
    4 音乐
    5 音效
    6 粒子
    7 纹理字体
    8 ttf字体
    9 shader [新增]

Shader补充说明
    Shader使用D3D9的fx格式，在fx中通过在Annotation中添加名为"binding"的注释来和脚本系统互联。
    允许在lua端进行赋值的类型有
        string：被解释并定位到纹理资源
        number：被解释成float
        lstgColor：被解释成float4
    当前，Shader仅被用于PostEffect。
```

- RemoveResource(pool:string, [type:integer, name:string]) **[新]**

  若只有一个参数，则删除一个池中的所有资源。否则删除对应池中的某个资源。参数可选global或stage。

  若资源仍在使用之中，将继续保持装载直到相关的对象被释放。

- CheckRes(type:integer, name:string):string|nil

  获得一个资源的类别，通常用于检测资源是否存在。

  ```
  细节
      方法会根据名称先在全局资源池中寻找，若有则返回global。
      若全局资源表中没有找到资源，则在关卡资源池中找，若有则返回stage。
      若不存在资源，则返回nil。
  ```

- EnumRes(type:integer):table, table

  枚举资源池中某种类型的资源，依次返回全局资源池、关卡资源池中该类型的所有资源的名称。

- GetTextureSize(name:string):number, number

  获取纹理的宽度和高度。

- LoadTexture(name:string, path:string, [mipmap:boolean=false])

  装载纹理，支持多种格式但是首推png。其中mipmap为纹理链。

- LoadImage(name:string, tex_name:string, x:number, y:number, w:number, h:number, [a:number, [b:number, [rect:boolean]]])

  在纹理中创建图像。x、y指定图像在纹理上左上角的坐标（纹理左上角为（0,0），向下向右为正方向），w、h指定图像的大小，a、b、rect指定横向、纵向碰撞判定和判定形状。

  ```
  细节
      当把一个图像赋予对象的img字段时，它的a、b、rect属性会自动被赋值到对象上。
  ```

- SetImageState(name:string, blend_mode:string, [vertex_color1:lstgColor, vertex_color2:lstgColor, vertex_color3:lstgColor, vertex_color4:lstgColor])

  设置图像状态，可选一个颜色参数用于设置所有顶点或者给出4个颜色设置所有顶点。

  ```
  混合选项可选
      ""          默认值，=mul+alpha
      "mul+add"   顶点颜色使用乘法，目标混合使用加法
      "mul+alpha" (默认)顶点颜色使用乘法，目标混合使用alpha混合
      "mul+sub"   顶点颜色使用乘法，结果=图像上的颜色-屏幕上的颜色 [新增]
      "mul+rev"   顶点颜色使用乘法，结果=屏幕上的颜色-图像上的颜色 [新增]
      "add+add"   顶点颜色使用加法，目标混合使用加法
      "add+alpha" 顶点颜色使用加法，目标混合使用alpha混合
      "add+sub"   顶点颜色使用加法，结果=图像上的颜色-屏幕上的颜色 [新增]
      "add+rev"   顶点颜色使用加法，结果=屏幕上的颜色-图像上的颜色 [新增]
  ```

- SetImageCenter(name:string, x:number, y:number)

  设置图像中心。x和y相对图像左上角。

- LoadAnimation(name:string, tex_name:string, x:number, y:number, w:number, h:number, n:integer, m:integer, intv:integer, [a:number, [b:number, [rect:boolean]]])

  装载动画。a、b、rect含义同Image。其中x、y指定第一帧的左上角位置，w、h指定一帧的大小。n和m指定纵向横向的分割数，以列优先顺序排列。intv指定帧间隔。

  动画总是循环播放的。

- SetAnimationState(name:string, blend_mode:string, [vertex_color1:lstgColor, vertex_color2:lstgColor, vertex_color3:lstgColor, vertex_color4:lstgColor])

  含义类似于SetImageState。

- SetAnimationCenter(name:string, x:number, y:number)

  含义类似于SetImageCenter。

- LoadPS(name:string, def_file:string, img_name:string, [a:number, [b:number, [rect:boolean]]])

  装载粒子系统。def_file为定义文件，img_name为粒子图片。a、b、rect含义同上。

  使用HGE所用的粒子文件结构。

- LoadFont(name:string, def_file:string, [bind_tex:string, mipmap:boolean=true])

  装载纹理字体。name指示名称，def_file指示定义文件，mipmap指示是否创建纹理链，默认创建。bind_tex参数为f2d纹理字体所用，指示绑定的纹理的完整路径。

  > 细节
  > luastg+支持HGE的纹理字体和fancy2d的纹理字体（xml格式）。
  > 对于hge字体，将根据定义文件在字体同级目录下寻找纹理文件，对于f2d字体，将使用bind_tex参数寻找纹理。

- SetFontState(name:string, blend_mode:string, [color:lstgColor])

  设置字体的颜色、混合模式。具体混合选项见上文。

- SetFontState2(...) **[移除]**

  该方法用于设置HGE的纹理字体，luastg+不支持此方法。

  > 细节
  > 大部分现有代码中没有使用该方法，可以无视。

- LoadTTF(name:string, path:string, width:number, height:number) **[不兼容]**

  该方法用于加载TTF字体。name指定资源名称，path指定加载路径，width和height指定字形大小，建议设为相同值。

  > 细节
  > LoadTTF方法相比luastg有巨大不同。由于使用内置的字体渲染引擎，当前实现下不需要将字体解压并导入系统。
  > 但是相对的、无法使用诸如加粗、倾斜等效果。同时参数被缩减到4个。
  > 此外，若无法在path所在位置加载字体文件，

- RegTTF(...) **[否决]**

  该函数不再起效。将于日后版本移除。

- LoadSound(name:string, path:string)

  装载音效。仅支持wav或ogg，推荐使用wav格式。

  > 细节

音效将被装载进入内存。请勿使用较长的音频文件做音效。
对于wav格式，由于受限于目前的实现，故不支持非标准的、带压缩的格式。（如AU导出时若带有metadata将导致无法解析）

- LoadMusic(name:string, path:string, end:number, loop:number)

  装载音乐。name指定名称，path指定路径，end和loop指定循环节终止和持续时间（秒），即循环节范围为end-loop ~ end。

  仅支持wav或ogg，推荐使用ogg格式。

  细节
  音乐将以流的形式装载进入内存，不会一次性完整解码放入内存。故不推荐使用wav格式，请使用ogg作为音乐格式。
  通过描述循环节可以设置音乐的循环片段。当音乐位置播放到end时会衔接到start。这一步在解码器中进行，以保证完美衔接。

- LoadFX(name:string, path:string) **[新增]**

  装载Shader特效。

渲染方法

luastg/luastg+使用笛卡尔坐标系（右正上正）作为窗口坐标系，且以屏幕左下角作为原点，Viewport、鼠标消息将以此作为基准。

luastg/luastg+中存在一个全局图像缩放系数，用于在不同模式下进行渲染，该系数将会影响对象的渲染大小、与图像绑定的碰撞大小和部分渲染函数。

luastg/luastg+不开启Z-Buffer进行深度剔除，通过排序手动完成这一工作。

另外，从luastg+开始，渲染和更新将被独立在两个函数中进行。所有的渲染操作必须在RenderFunc中执行。

- BeginScene()

  通知渲染开始。该方法必须在RenderFunc中调用。所有渲染动作必须在BeginScene/EndScene中进行。

  不兼容性
  从luastg+开始，渲染操作将被移动到RenderFunc中进行。

- EndScene()

  通知渲染结束。该方法必须在RenderFunc中调用。

- RenderClear(lstgColor)

  使用指定颜色清空屏幕。在清除颜色的同时会清除深度缓冲区。

- SetViewport(left:number, right:number, bottom:number, top:number)

  设置视口，将影响裁剪和渲染。

- SetOrtho(left:number, right:number, bottom:number, top:number)

  设置正投影矩阵。left表示x轴最小值，right表示x轴最大值，bottom表示y轴最小值，top表示y轴最大值。

  细节
  创建的正投影矩阵将把z轴限制在[0,1]区间内。

- SetPerspective(eyeX:number, eyeY:number, eyeZ:number, atX:number, atY:number, atZ:number, upX:number, upY:number, upZ:number, fovy:number, aspect:number, zn:number, zf:number)

  设置透视投影矩阵和观察矩阵。(eyeX,eyeY,eyeZ)表示观察者位置，(atX,atY,atZ)表示观察目标，(upX,upY,upZ)用于表示观察者向上的正方向。fovy描述视角范围（弧度制），aspect描述宽高比，zn和zf描述z轴裁剪距离。

- Render(image_name:string, x:number, y:number, [rot:number=0, [hscale:number=1, [vscale:number=1, [z:number=0.5]]]])

  渲染图像。(x,y)指定中心点，rot指定旋转（弧度制），(hscale,vscale)XY轴缩放，z指定Z坐标。

  若指定了hscale而没有指定vscale则vscale=hscale。

该函数受全局图像缩放系数影响。

- RenderRect(image_name:string, left:number, right:number, bottom:number, top:number)

  在一个矩阵范围渲染图像。此时z=0.5。

- Render4V(image_name:string, x1:number, y1:number, z1:number, x2:number, y2:number, z2:number, x3:number, y3:number, z3:number, x4:number, y4:number, z4:number)

  给出四个顶点渲染图像。此时z=0.5。

- SetFog([near:number, far:number, [color:lstgColor = 0x00FFFFFF]])

  若参数为空，将关闭雾效果。否则设置一个从near到far的雾。

- RenderText(name:string, text:string, x:number, y:number, [scale:number=1, align:integer=5])

  使用纹理字体渲染一段文字。参数name指定纹理名称，text指定字符串，x、y指定坐标，align指定对齐模式。

  该函数受全局图像缩放系数影响。

  ```
  细节
      对齐模式指定渲染中心，对齐模式可取值：
          左上  0 + 0  0
          左中  0 + 4  4
          左下  0 + 8  8
          中上  1 + 0  1
          中中  1 + 4  5
          中下  1 + 8  9
          右上  2 + 0  2
          右中  2 + 4  6
          右下  2 + 8  10
      由于使用了新的布局机制，在渲染HGE字体时在横向上会有少许误差，请手动调整。
  ```

- RenderTexture(tex_name:string, blend:string, vertex1:table, vertex2:table, vertex3:table, vertex4:table)

  直接渲染纹理。

  ```
  细节
      vertex1~4指定各个顶点坐标，其中必须包含以下项：
          [1] = X坐标
          [2] = Y坐标
          [3] = Z坐标
          [4] = U坐标（以纹理大小为区间）
          [5] = V坐标（以纹理大小为区间）
          [6] = 顶点颜色
      注意该函数效率较低，若要使用请考虑缓存顶点所用table。
  ```

- RenderTTF(name:string, text:string, left:number, right:number, bottom:number, top:number, fmt:integer, blend:lstgColor) **[不兼容]**

  渲染TTF字体。

  该函数受全局图像缩放系数影响。

  ```
  细节
      暂时不支持渲染格式设置。
      接口已统一到屏幕坐标系，不需要在代码中进行转换。
  ```

- PostEffectCapture() **[新增]**

  开始捕获绘制数据。

  从这一步开始，所有后续渲染操作都将在PostEffect缓冲区中进行。

  必须使用PostEffectApply执行PostEffect方可退出。

高级方法。

- PostEffectApply(name:string, blend:string, [args:table]) **[新增]**

  应用PostEffect。参数指定传递给FX的参数表，将会影响后续对该FX的使用。

  其中blend指定posteffect要以什么样的形式绘制到屏幕上，此时blend的第一分量无效。

  高级方法。

  ```
  细节
      对于PostEffect只会渲染第一个technique中的所有pass。
      可以在PostEffect中使用下列语义注释(不区分大小写)捕获对象：
          POSTEFFECTTEXTURE获取posteffect的捕获纹理(texture2d类型)。
          VIEWPORT获取视口大小(vector类型)。
          SCREENSIZE获取屏幕大小(vector类型)。
  ```

声音播放

- PlaySound(name:string, vol:number, [pan:number=0.0])

  播放一个音效。vol为音量，取值范围[0~1]，pan为平衡，取值[-1~1]。

  ```
  细节
      luastg+每次只播放一个音效，如果一个音效已在播放中则会打断这个播放从头开始。
  ```

- StopSound(name:string) **[新增]**

  停止播放音效。name为资源名称。

- PauseSound(name:string) **[新增]**

  暂停播放音效。name为资源名称。

- ResumeSound(name:string) **[新增]**

  继续播放音效。name为资源名称。

- GetSoundState(name:string):string **[新增]**

  获取音效播放状态，将返回paused、playing、stopped。

- PlayMusic(name:string, [vol:number=1.0, position:number=0])

  播放音乐。name为资源名称，vol为音量，position为起始播放位置（秒）。

- StopMusic(name:string)

  停止播放音乐。该操作会使音乐播放位置回到开头。name为资源名称。

- PauseMusic(name:string)

  暂停播放音乐。name为资源名称。

- ResumeMusic(name:string)

  继续播放音乐。name为资源名称。

- GetMusicState(name:string):string

  获取音乐播放状态，将返回paused、playing、stopped。

- UpdateSound() **[否决]**

  该方法已不起任何作用，将于后续版本移除。

- SetSEVolume(vol:number)

  设置全局音效音量，将影响后续播放音效的音量。

- SetBGMVolume([vol:number] | [name:string, vol:number]) **[新]**

  若参数个数为1，则设置全局音乐音量。该操作将影响后续播放音乐的音量。

  若参数个数为2，则设置指定音乐的播放音量。

输入

当前，手柄输入被映射到0x92~0xB1和0xDF~0xFE（共2个手柄、32个按键）的位置上。

其中，X轴Y轴的位置被映射到前4个按键上，对应上下左右。

- GetKeyState(vk_code:integer):boolean

  给出虚拟键代码检测是否按下。

  > 细节
  > VK_CODE的具体含义请查阅MSDN。

- GetLastKey():integer

  返回最后一次输入的按键的虚拟键代码。

- GetLastChar():string

  返回上一次输入的字符。

- GetMousePosition():number,number **[新增]**

  获取鼠标的位置，以窗口左下角为原点。

- GetMouseState(button:integer):boolean **[新增]**

  检查鼠标按键是否按下。button可取0、1、2，分别对应鼠标左键、中键、右键。

杂项

- Snapshot(file_path:string)

  截屏并保存到file_path。格式为PNG。

内置数学方法

下述数学函数均以角度制为基准，含义同C语言库函数。

- sin(ang)
- cos(ang)
- asin(v)
- acos(v)
- tan(ang)
- atan(v)
- atan2(y,x)

内置对象构造方法

- Rand()

构造一个随机数生成器。以当前系统tick数做Seed。

- Color([argb:integer] | [a:integer, r:integer, g:integer, b:integer])

  构造一个颜色。

- BentLaserData()

  构造一个曲线激光控制器。

调试方法

- ObjTable():table

  该方法可以获得对象池所在的table。慎用。

- Registry():table **[移除]**

  返回注册表。

  > 细节
  > 由于该方法过于不安全，已被移除。

全局回调函数

下述回调函数必须定义在脚本的全局范围中。

- GameInit()

  在游戏框架初始化完毕后，游戏循环启动前调用。

- FocusLoseFunc()

  在渲染窗口失去焦点时调用。

- FocusGainFunc()

  在渲染窗口获得焦点时调用。

- FrameFunc():boolean

  帧处理函数，每帧被调用来处理逻辑。

  若返回true将终止游戏循环，退出游戏。

- RenderFunc() **[新增]**

  渲染处理函数，每帧被调用时用来渲染场景。

第三方库

cjson [新增]

方法

更多帮助请参考cjson主页

- encode(table):string

  编码一个table为字符串。

- decode(string):table

解码一个字符串为table。