**Lua编码规范**



**Version Control**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | Change Description |
| 1.1 | 05/20/2016 | 花成 | 初稿 |



[**Version Control** 1](#_Toc451525485)

[1 命名部分 1](#_Toc451525486)

[1.1 代码命名规范 1](#_Toc451525487)

[1.2 文件、文件夹命名规范 1](#_Toc451525488)

[2 文件组织部分 2](#_Toc451525489)

[3 编码风格部分 2](#_Toc451525490)



# 命名部分

## 代码命名规范

* 全局变量所有单词所有字母均大写，字母间加下划线。如UI\_CONST。
* 函数名及局部变量名首单词首字母小写，其后单词首字母大写，非首字母 均小写，字母间无下划线。如importModules和tempPosition。
* 类名所有单词首字母大写，非首字母均小写，字母间无下划线。如ObjBoss。
* 类中成员变量的命名在局部变量命名规范基础上额外要求加单下划线如\_uniqueObj和\_cardId，对于那些希望仅内部访问的变量（私有变量）则加双下划线如\_\_index。
* 代码缩进（每行起始缩进）统一使用一个tab，一个tab的长度设置为4个空格单位。

## 文件、文件夹命名规范

* 文件命名为首单词首字母小写，其后单词首字母大写，非首字母均小写，字母间无下划线。如cdEffect.png和guanYu.png
* 文件夹命名为所有字母小写，字母间加下划线（这样可与文件命名区分开）。如res，sprite,src,naming\_standard

编码风格部分

# 文件组织部分

* 文件开头加上此文件的功能、 职责的简要描述； 每个文件都加 module 限定词； 导

入的模块都加 local 限定词.

* 重要函数都加如下格式的注释。

例如：

--此函数检测是否可以从 A(oldx, oldy)点走到 B 点（newx, newy）

--@param oldx 当前所在点 x

--@param oldy 当前所在点 y

--@param newx 目标点 x

--@param newy 目标点 y

--@return 若可以到达，返回 true；否则返回 false

function obj:checkbar(oldx, oldy, newx, newy)

…

End

* 函数与函数间、以及一些定义之间加上空行。
* 文件内不允许出现全局变量，\_G.instance 例外
* 函数内的临时变量、文件内的局部函数都加上 local 限定词
* 常量、消息号、枚举值行末都加上分号。
* 函数的行数过长（大于 100 行）时，尽量拆分为多个子函数；函数中一些晦涩的部

分，一定要加上注释。

* 短小的注释使用 --； 较长的注释使用 --[[ ]]
* assert 函数开销不小，请慎用。
* Lua 类设计时，用元表来实现 oop。

不要直接增加函数成员，因为直接增加函数成员会导致内存增加并且在 jit 下执行

效率和用元表方式无差异。

# 编码风格部分

lua作为一门语法宽松的动态语言，允许你在任何时候任何地点给table或者userdata添加变量。但这样作会将导致对象管理上的极大困难（尤其在初始化和清空的时刻），而且查找bug时也会非常麻烦。因此对此部分做如下要求：

类的任何非静态成员属性都应该在ctor()函数中被恰当的申明，不允许在ctor()外给类或者对象添加变量。

# 代码建议

1． 代码中使用的一些函数尽可能在文件开头或者当前局部环境中加local前缀重新定

义下。

例如：

local assert = assert

2． 不要使用元表来实现继承

3． 高级特性尽可能不用

4． 写代码时尽可能写的简单，考虑性能时先做好推断，看看能提升多少，增加的复杂

度以及造成的代码晦涩有多严重，然后再决定如何做

5． 加载的 xml 数据表，尽可能的做好数据校验，若校验失败，要出发断言，使服务器

无法启动；不要等出错时，回过头来检查是数据表问题还是逻辑问题。

6． 出错时，记录好错误日志。

有的函数开销比较大，而调用的频率很低，那么可以不对他做优化；

反之，有的函数开销较小，但是调用的频率很高，从如何降低调用频率以及减少函

数开销两个角度去思考，然后定下优化方案

7． 提交代码之前，去掉或者注释掉无关的代码； 测试下保证服务器可以正确启动。

8． 尽量减少表中的成员是另一个表的引用。 考虑 lua 的垃圾收集机制、 内存泄露等。

