# 使用lua需要做的工作

1. 在项目中引入liblua库，并添加必要的编译和链接需要的包含文件夹及lib文件
2. 在cmakelist.txt以及相应的.mk文件中添加liblua相关的编译文件

# debug与log

## debug

* debug可以使用Decoda软件，只需要配置project->setting即可，注意我们编辑的lua文件是在Resources中，不是在debug.win32/Resources，所以working参数应该选择前者，这样每次编辑完以后就能起效。可以在project->add exsiting file添加我们要debug的lua文件，不过不知道怎么能实现自动添加。
* 软件支持常见的堆栈，断点，watch等操作。

![C:\Users\xwtan\AppData\Roaming\Tencent\Users\420244052\QQ\WinTemp\RichOle\XX2E~DJ7QG5R]S8V%SGC](D.jpg](data:image/jpeg;base64,)

## Log

实现log分级的思路：

1. 将c++中的宏与lua中的全局变量对应，如果c++中定义了DEBUG宏，则在lua中定义DEBUG全局变量。
2. 在lua中有全局函数DLOG，ELOG等，如果定义了DEBUG全局变量，则调用相应的c++函数输出，否则什么都不做

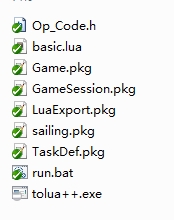
# 向Lua注册c++自定义类

## 组织结构

各个项目有LuaExport文件夹，此文件里面的文件是为了生成{projName}\_LuaCocos2d.cpp，这个文件的作用是向lua注册c++自定义类。

## 生成方法

sailing项目下LuaExport文件夹的文件如下图：



生成方法如下：

1. 将相关类的头文件复制到LuaExport文件夹，后缀名改为.pkg，并根据ReadMe.txt(这个文件在cocos2d\_ui项目的LuaExport中)里面的规则编辑里面的内容。
2. 在{projName}.pkg（这里就是sailing.pkg）里面添加上相应的pkg文件（注意按照类的继承关系确定添加顺序，参见ReadMe.txt）。
3. 在basic.lua里面修改相关的#include””部分。
4. 修改run.bat里面的PROJ\_NAME和CPP\_DIR两个参数的值，运行run.bat，这个脚本会使用tolua++.exe根据{projName}.pkg生成相应的{projName}\_LuaCocos2d.cpp，并根据basic.lua进行某些替换，最后移动到CPP\_DIR文件夹下（需要先手动创建CPP\_DIR文件夹）。
5. 再添加相应的{projName}\_LuaCocos2d.h文件（可参考sailing里面的类似文件，仅需要修改tolua\_sailing\_open函数名称，一次添加即可）。
6. 在项目的cmake.txt的include\_directories变量及android.mk的COCOS2DX\_INCLUDES变量中添加相应的lua文件夹，参考cocos2d\_ui项目相应文件。
7. 在主项目的AppDelegate.cpp中包含相应的{projName}\_LuaCocos2d.h，并调用相应的tolua\_{projName}\_open函数。
8. 只要pkg文件没有问题，就应该可以编译运行并在lua中调用相应的自定义类了。

## 需注意的问题

1. 一定要根据ReadMe.txt里面的规则正确编辑pkg文件。
2. 自定义类的构造函数注册以后，在lua里面是通过new\_local或者\_\_call调用，析构函数不需要显示调用。
3. 自定义类如果处理触摸，则要实现以下几个方法：

virtual void registerScriptTapHandler(int nHandler);

virtual void unregisterScriptTapHandler(void);

int getScriptTapHandler() { return m\_nScriptTapHandler; };

1. 由于生成的{projName}\_LuaCocos2d.cpp一般比较大，我们希望只在用到lua框架的时候编译它，实现条件编译用以下方法。

* 在vs中，以cocos2d\_ui项目为例

1. 在src下新建lua文件夹，将cocos2d\_ui\_LuaCocos2d.cpp及相应的头文件复制到此文件夹下。
2. 在CMakeLists.txt文件中添加

if(LUA\_COCOS2D)

aux\_source\_directory(./src/lua SRC\_LIST)

endif(LUA\_COCOS2D)  
还需要修改HEADER\_LIST变量。

1. configure-client.cmd调用cmake时需加上参数-DLUA\_COCOS2D=ON，注意-G "Visual Studio 10"命令最后一定要加上 ..，注意前面有空格

* 在android下

1. 在android.mk中添加

ifdef LUA\_COCOS2D

COCOS2D\_UI\_SRC\_FILES += $(wildcard $(COCOS2D\_UI\_PATH)/../src/lua/\*.cpp)

endif

1. 在主项目的android.mk最前面定义LUA\_COCOS2D变量并赋值（没有赋值认为没有定义）

# C++自定义类在lua中的内存管理

* 对于autorelease实现的类，就像cocos2d的控件一样，用create创建，不需要额外管理。
* 对于使用构造函数定义对象的类，在lua中使用new\_local或者\_\_call来调用，被创建的这个对象不再被引用时，将由lua回收。
* 如果某个类的析构函数中有工作，那么析构函数需要写在pkg文件中。

# C++调lua

使用库lua\_tinker，现在放在sailing中，以后考虑放在其他底层项目中。lua\_tinker本身并不支持android，做了一些小修改是的能在android下工作。

# 配置文件进行布局

## 配置文件

* 编码

local MapLayerLayout = {

{

ccType = "menu",

childrenLua = "lua/layout/map/MapLayerMenuItem.lua",

returnName = "menu",

},

{

ccType = "sprite",

imgPath = g\_imgTestMenu1,

position = function(pamsTable)

if pamsTable and pamsTable.test then

return ccp(10 , WIN\_HEIGHT / 2)

else

return ccp(710 , WIN\_HEIGHT / 2)

end

end

,

},

}}

return MapLayerLayout

* 注意事项
  1. childrenLua表示这个元素的子元素所在的lua文件，必须是完整路径。
  2. ccType表示元素的类型，在ParseLayout.lua最前面说明了所有的类型，必须匹配，

各个类型具有的属性也有说明。

* 1. 解析完成以后会返回一个table，如果某个元素有returnName属性，那么table中以returnName的值作为key就能找到这个元素。
  2. 任意一个属性对应的值都可以是一个函数，函数的参数是pamsTable，由parseLayout传入，函数可以根据这个参数返回不同的值。

## 配置文件解析

解析代码都在ParseLayout.lua中，调用parseLayout(parent, layoutLua, parmsTable)解析即可。如果需要对新的控件解析只需要在其中添加一个新的解析函数即可。

# Lua中各个module的生命周期管理

## Layout的module

在parseLayout函数中控制生命周期，解析完了就会置为nil。

## UI的module

### 方案1

* 编码
  1. Luainit函数中设置：

setmetatable(package.loaded, {\_\_mode = "v"})

* 1. 在module编码如下：

local modName = "BuildingLayer"

module(modName, package.seeall)

…

function sceneEventHandler(eventType)

if eventType == "enter" then

--dosomething

elseif eventType == "exit" then

deleteModule(modName)

end

end

…

return \_G[modName]

* 1. 引用module时：

function menuCallbackOpenPopup()

local Building= require "lua/ui/building/BuildingLayer"

Building:showDialog()

End

* 注意事项

1. 在离开页面是要deleteModule。
2. 最后要返回\_G[modName]。
3. 调用的时候要使用local变量保存require结果，通过这个调用（直接通过全局变量BuildingLayer也可以调，但是第二次打开页面会有问题）。
4. 由于对userdata的收集是在垃圾收集的最后一个阶段，如果某个module里面有userdata，那么当对table回收时必须经过两轮回收，第一轮先把userdata回收，第二轮才能把table回收。

### 方案2

* 编码

在方案1的基础上的改动：

1. module名字必须是文件的路径名，也就是跟require参数一样:

local modName = " lua/ui/building/BuildingLayer "

1. 结尾不需要返回。

* 注意事项

由于module名字必须是文件的路径名，维护起来很麻烦，这个方案可以放弃。

### 方案3

这个方案不使用弱引用。

* 编码

1. 在module编码如下：

local modName = " lua/ui/building/BuildingLayer "

local M = {}

package.loaded[modName] = M

setmetatable(M, {\_\_index = \_G})

setfenv(1, M)

…

function sceneEventHandler(eventType)

if eventType == "enter" then

--dosomething

elseif eventType == "exit" then

deleteModule(modName)

end

end

…

1. 引用module时同方案1。

* 注意事项

1）由于module名字必须是文件的路径名，维护起来还是很麻烦。

2）module内部的开头那部分代码是必须的。

## GameModel的module

在代码里面主动控制，某个GameModel不需要时，调GameModel的destory函数把自己删除掉。

# 数据UI交互，opcode处理



实现细节：

## GameModelManagerBase

opcodeHandleTableModel = {}

opcodeHandleTableInUI = {}

Table的格式：

{opcode={

{“obj”=obj, “handle”=handle}

,…}

,…}

## GameModelManager

--处理opcode，从c++层调, packet是网络包

function handleOpcodeInModels(packet)

--处理opcode，从c++层或者GameModel调, packet是解包以后的数据结构

function handleOpcodeInUI(packet, opcode)

--绑定opcode处理函数

local function bindOpcodeHandle(opcodeHandleTable, opcode, obj, handle)

--取消绑定

local function cancelBindOpcodeHandle(opcodeHandleTable, obj, opcode)

## GameModelBase

用table模拟类，这个相当于类的定义，每个具体的GameModel都是类的一个对象，并对这里面的方法重写。

GameModelBase = {}

--依靠这个函数模拟类

function GameModelBase:new(obj)

--在每个GameModel中重写这些函数

function GameModelBase:bindHandles()

assert(false, "not implementation!")

end

function GameModelBase:cancelBindHandles()

assert(false, "not implementation!")

end

function GameModelBase:getObjName()

return "GameModelBase"

end

return GameModelBase

# 遗留问题

## Module生命周期管理

7中已经讨论过

## 代码安全性问题

可以用luajit编译成字节码，但是这部分还没有进行调研，编译出来的文件命名、存放路径、运行都需要研究。

## 自定义类注册到lua中

基本功能用tolua++就能实现，以下几点需要额外的工作：

1. touch处理
2. 事件绑定
3. Delegate

这些内容在cocos2d的控件中都有类似的实现，需要对原来的类添加新的内容支持lua，有些实现起来比较麻烦（比如delegate）。