Integrating Lua into an iPhone App

Friday, November 13, 2009 at 11:54AM
Robert in Cocos2d, Lua, iPhone

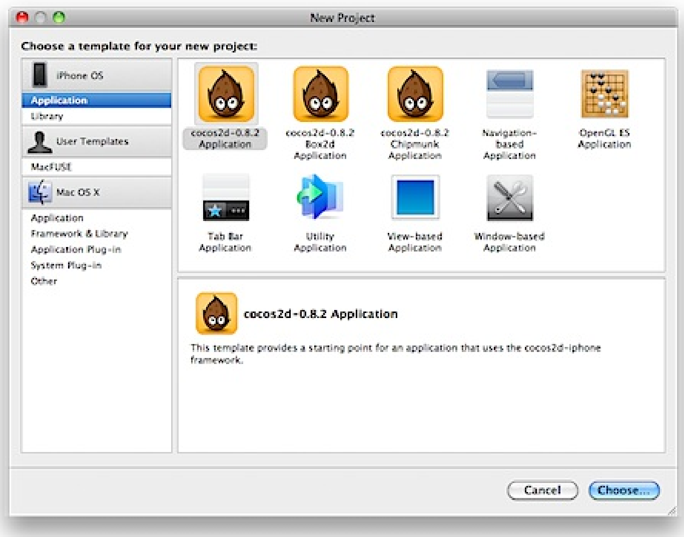
I spent a long evening getting Lua to compile and link in an iPhone project. It turned out to be a very simple task once I figured out some basic things about adding targets in Xcode and remembered one small quirk about Lua that I learned a few years ago and forgot along the way. I'm going to outline the process below using a Cocos2d project although a UIKit project would be the same. Please note that I'm still new to Xcode so this may not be the most optimal approach but it works well. I welcome your comments and corrections.

In order to keep this post as short as possible, I will only describe the process of getting everything to build and function. I'll try to post more later about the how's and why's of Lua.

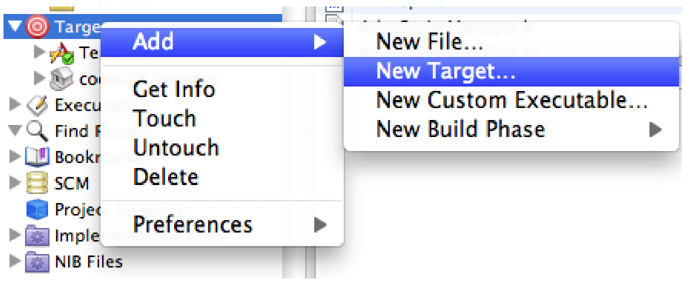
You can find a zipped version of the project here if you don't want to follow the process below.

Before starting anything in Xcode, there is the question of how to structure the project: as a single project and target, as a single project with an additional Lua target, or as two different projects. Xcode makes it easy to use any of those approaches but I chose to use a single project with multiple targets. One of the additional targets will be used to build a static Lua library. This gives the benefit of keeping Lua separate from your project code but it keeps all of the configuration and SDK information unified. Incidentally, this is the same approach taken by the Cocos2d library when you create a new project using the included template.

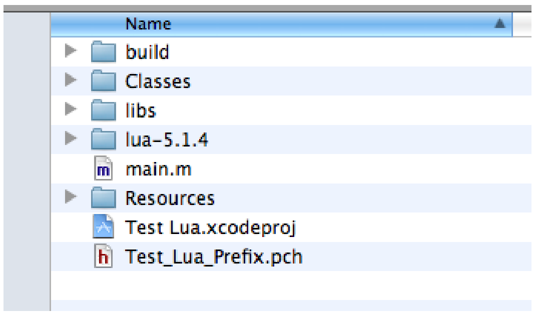
The first task is to create a new project. In Xcode , click **File->New Project**. Select a **Cocos2d Application** and name it "**Test Lua**".


New Project
Choose a template for your new project:
iPhone OS
Application
Library
User Templates
MacFUSE
Mac OS X
Application
Framework & Library
Application Plug-in
System Plug-in
Other
cocos2d-0.8.2 Application
cocos2d-0.8.2 Box2d Application
cocos2d-0.8.2 Chipmunk Application
Navigation-based Application
OpenGL ES Application
Tab Bar Application
Utility Application
View-based Application
Window-based Application
cocos2d-0.8.2 Application
This template provides a starting point for an application that uses the cocos2d-iphone framework.
Cancel
Choose...

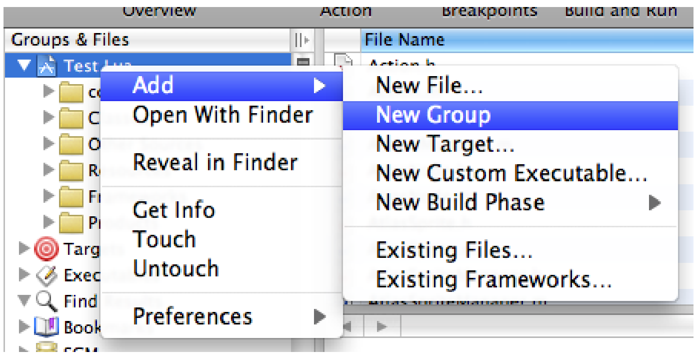
Next, right-click the "**Targets**" item on the left pane. Select "**Add->New Target…**" Select a "**Static Library**" and name it "**Lua**"



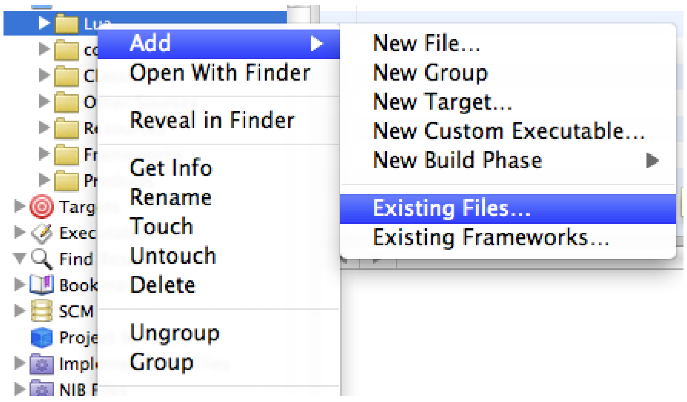
Download the Lua source from http://www.lua.org and unzip it to your project directory. The current Lua version is 5.1.4 so my project directory now looks like:



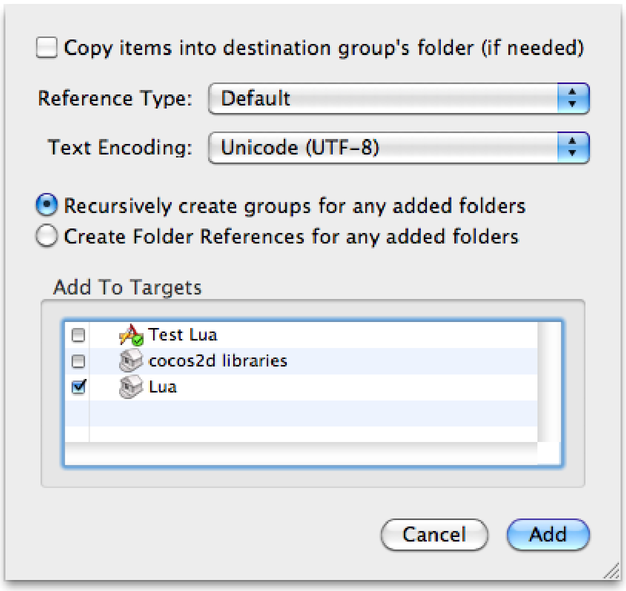
Right click on the "**Test Lua**" header in the left pane of the Xcode window. Select "**Add->New Group** " and name it "**Lua**"



Right click on the new "**Lua**" group and select "**Add->Existing Files…**" Navigate to your **Test Lua/Lua5.1.4/src** directory and select all of the .h and .c files except lua.c, luac.c, and print.c . These are used to build standalone Lua interpreters and compilers and are not used in library builds.



When presented with the options dialog below, be sure to uncheck "**Copy items into destination groups folder**" and select only "**Lua**" under the "**Add to Targets**" section.


☐ Copy items into destination group's folder (if needed)

Reference Type: Default

Text Encoding: Unicode (UTF-8)

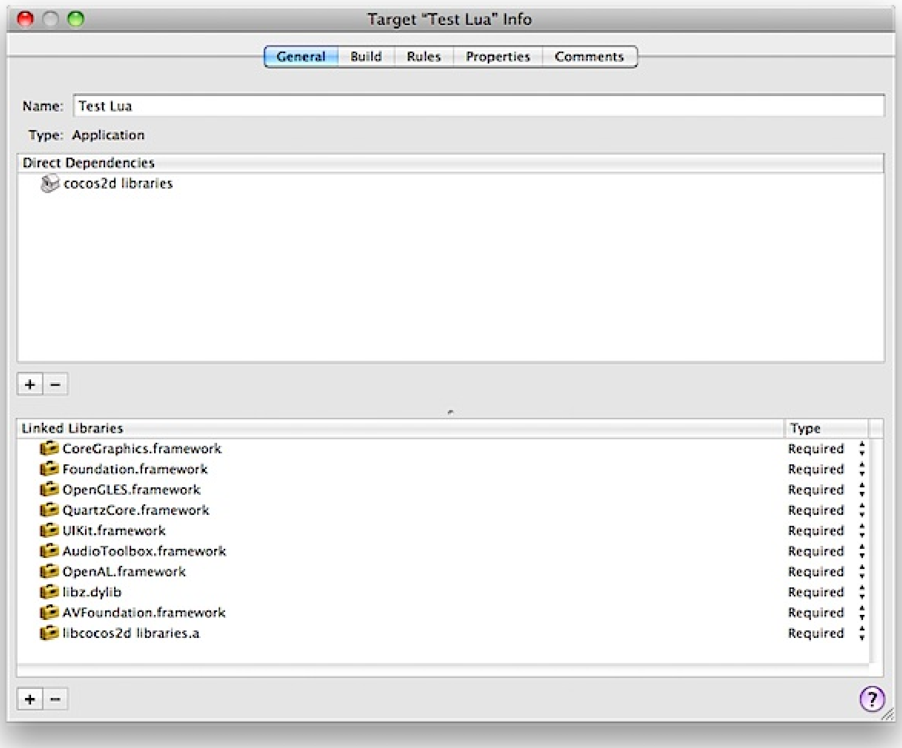
● Recursively create groups for any added folders
○ Create Folder References for any added folders

Add To Targets

☐ Test Lua
☐ cocos2d libraries
☑ Lua

You should be able to right-click the Lua target and build it without errors at this point.

Under "**Targets**," right-click the "**Test Lua**" item and select "**Get info**." Select the "**General**" tab at the top.



Name: Test Lua

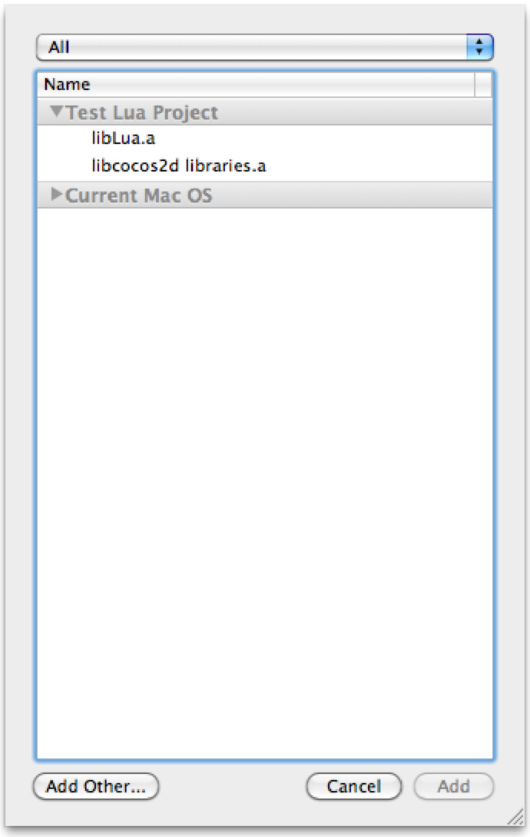
Type: Application

Direct Dependencies
- cocos2d libraries

Linked Libraries	Type
CoreGraphics.framework	Required
Foundation.framework	Required
OpenGLES.framework	Required
QuartzCore.framework	Required
UIKit.framework	Required
AudioToolbox.framework	Required
OpenAL.framework	Required
libz.dylib	Required
AVFoundation.framework	Required
libcocos2d libraries.a	Required

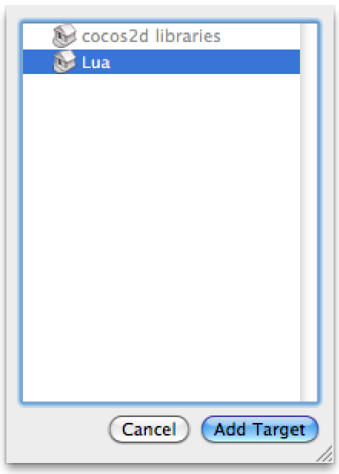
Click the "+" button beneath "**Linked Libraries**"

Select "**libLua.a**" at the top and click the "**Add**" button.



Click the "+" button beneath "**Direct Dependencies**"

Select "**Lua**" and click the "**Add Target**" button.



Close the “**Target Test Lua Info**” window.

You can now clean and build your project and you will see the three build steps happen: your project code, Cocos2d, and Lua. It should build without warnings or errors. If you get linker errors, be sure that you did not include lua.c, luac.c, or print.c from the Lua5.1.4 directory. Since they define a “main” function any builds including them will fail.

Now we can add some code to initialize Lua and run a little code. Open the "**HelloWorldScene.m**" file under "**Classes**" and add the following code after the #import "HelloWorldScene.h" line:

```
extern "C"{
#include "lua.h"
#include "lualib.h"
#include "lauxlib.h"
};



int run_lua(void)
{
lua_State *l;
l = lua_open();
luaopen_base(l);

printf("\nAbout to run Lua code\n");
luaL_loadstring(l, "print(\"Running Lua Code...\")");
lua_pcall(l, 0, LUA_MULTRET, 0);
printf("Lua code done.\n\n");

lua_close(l);

return 0;
}
```

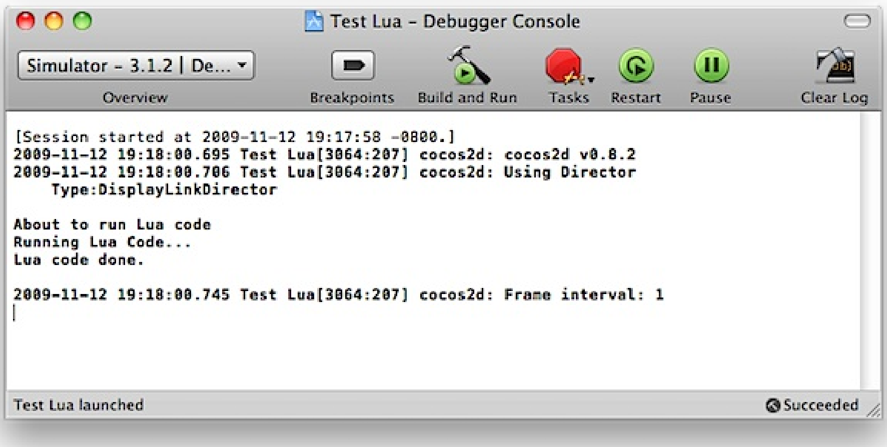
Now add the following line at the end of the HelloWorldScene "**init**" function:

```
run_lua();
```

Finally, rename the "**HelloWorldScene.m**" file to "**HelloWorldScene.mm**". This tells the compiler to treat the file as Objective-C++ so the Lua headers will be imported properly.

Build and run the code in a simulator and open the GDB window by clicking the button while the simulator is running.

You should see the window below with all three lines included, two from C and one from Lua.



```
[Session started at 2009-11-12 19:17:58 -0800.]
2009-11-12 19:18:00.695 Test Lua[3064:207] cocos2d: cocos2d v0.8.2
2009-11-12 19:18:00.706 Test Lua[3064:207] cocos2d: Using Director
    Type:DisplayLinkDirector

About to run Lua code
Running Lua Code...
Lua code done.

2009-11-12 19:18:00.745 Test Lua[3064:207] cocos2d: Frame interval: 1
```

Test Lua launched

Succeeded

That pretty much completes the point of this post- to get a working project with Lua compiling, linking, and executing. Obviously this is not very useful yet since Lua cannot interact with your application yet. Once I test my code more I'll post a small couple of classes to manage several Lua scripts and help integrate them into your game loop.

One final note, if you are doing a more standard app with UIKit then you might find iPhone Wax to be a better solution. In includes both Lua and a library to let Lua and Objective-C coexist as peers without registering functions in Lua before using them. For games you'd probably prefer the lightweight approach above.
