Setting up a game project on the Mac

This is the third article in a series for Orx on Mac OS X.

The first article helped the user set up their Mac for development by downloading the *Xcode Commandline Developer Tools* which gives the user the GCC compiler, the `git` command, and a MacOSX SDK.

The second article guided the user how to compile the Orx library.

This article will teach how to create your own game project using the Orx tools and Orx library.

Creating a new game project with "init"

In the root of the orx project folder, you will see a script called `init.sh`.

This tool let's you create a project for the Mac in all available IDEs and compilers.

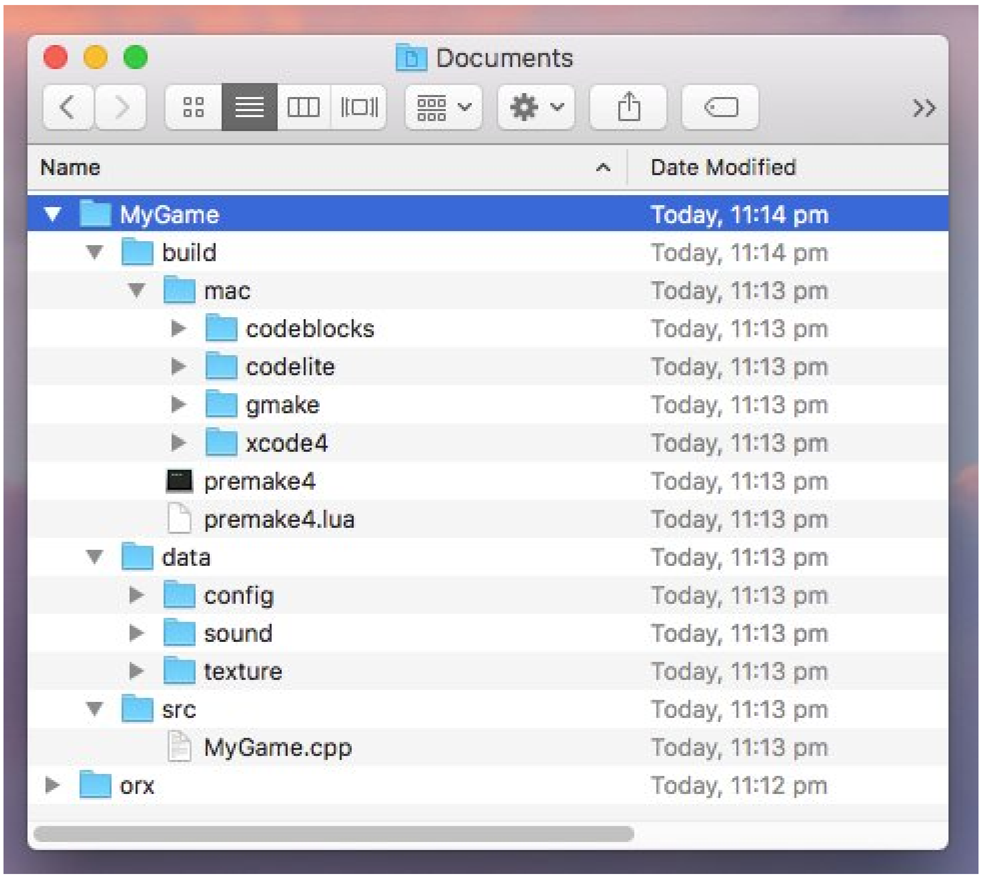
1. Open a Terminal
2. cd into the `~/Documents/orx/` folder
3. Type: `./init.sh ~/Documents/MyGame`

You will see output like:

```
[ 12:09:40 ] Initializing [ MyGame ] in [ /Documents/ ]
[ 12:09:40 ] == Creating files:
  + MyGame/.editorconfig
  + MyGame/build/premake4.lua
  + MyGame/data/config/MyGame.ini
  + MyGame/data/config/MyGamed.ini
  + MyGame/data/config/MyGamep.ini
  + MyGame/data/config/CreationTemplate.ini
  + MyGame/data/config/SettingsTemplate.ini
  + MyGame/data/sound/appear.ogg
  + MyGame/data/texture/logo.png
  + MyGame/src/MyGame.cpp
[ 12:09:40 ] Generating build files for [ mac ]:
  * gmake
Building configurations...
Running action 'gmake'...
Generating mac/gmake/Makefile...
Generating mac/gmake/MyGame.make...
Done.
  * codelite
Building configurations...
Running action 'codelite'...
Generating mac/codelite/MyGame.workspace...
Generating mac/codelite/MyGame.project...
Done.
  * codeblocks
Building configurations...
Running action 'codeblocks'...
Generating mac/codeblocks/MyGame.workspace...
Generating mac/codeblocks/MyGame.cbp...
Done.
  * xcode4
Building configurations...
Running action 'xcode4'...
Generating mac/xcode4/MyGame.xcworkspace/contents.xcworkspacedata...
Generating mac/xcode4/MyGame.xcodeproj/project.pbxproj...
Done.
[ 12:09:40 ] Init successful!
```

You can see that Solution Projects have been created for gmake, Codelite, Code::Blocks and Xcode.

Use Finder to take a look at the structure of your game folders:



```
MyGame            Today, 11:14 pm
  build           Today, 11:14 pm
    mac           Today, 11:13 pm
      codeblocks  Today, 11:13 pm
      codelite    Today, 11:13 pm
      gmake       Today, 11:13 pm
      xcode4      Today, 11:13 pm
    premake4      Today, 11:13 pm
    premake4.lua  Today, 11:13 pm
  data            Today, 11:13 pm
    config        Today, 11:13 pm
    sound         Today, 11:13 pm
    texture       Today, 11:13 pm
  src             Today, 11:13 pm
    MyGame.cpp    Today, 11:13 pm
orx               Today, 11:12 pm
```

Your project can be compiled and developed in any of these IDE's (or a text editor in the case of gmake). We'll use gmake because if you have been following the series with a clean Mac, gcc/make is what we have.

Compiling our game project with gmake

1. cd into `~/Documents/MyGame/build/mac/gmake`
2. type: `make`

(the above is the same as typing the default of: `make config=debug64`)

You'll see your game compile with:

```
==== Building orxLIB (debug64) ====
Creating obj/x64/Debug/orxLIB
orxPlugin_EmbeddedList.cpp
orxAnim.c
orxAnimPointer.c
orxAnimSet.c
----- >8 snip ------
orxLinkList.c
orxString.c
orxTree.c
Linking orxLIB
ld: warning: directory not found for option '-L/usr/lib64'
Running post-build commands
cp -f ../../../lib/dynamic/liborx*.dylib ../../../bin
==== Building orx (debug64) ====
Creating obj/x64/Debug/orx
orxMain.c
Linking orx
ld: warning: directory not found for option '-L/usr/lib64'
==== Building Bounce (debug64) ====
Creating obj/x64/Debug/Bounce
orxBounce.c
Linking Bounce
ld: warning: directory not found for option '-L/usr/lib64'
```

You'll also notice the post-build step of:

```
Running post-build commands
cp -f ../../../lib/dynamic/liborx*.dylib ../../../bin
```

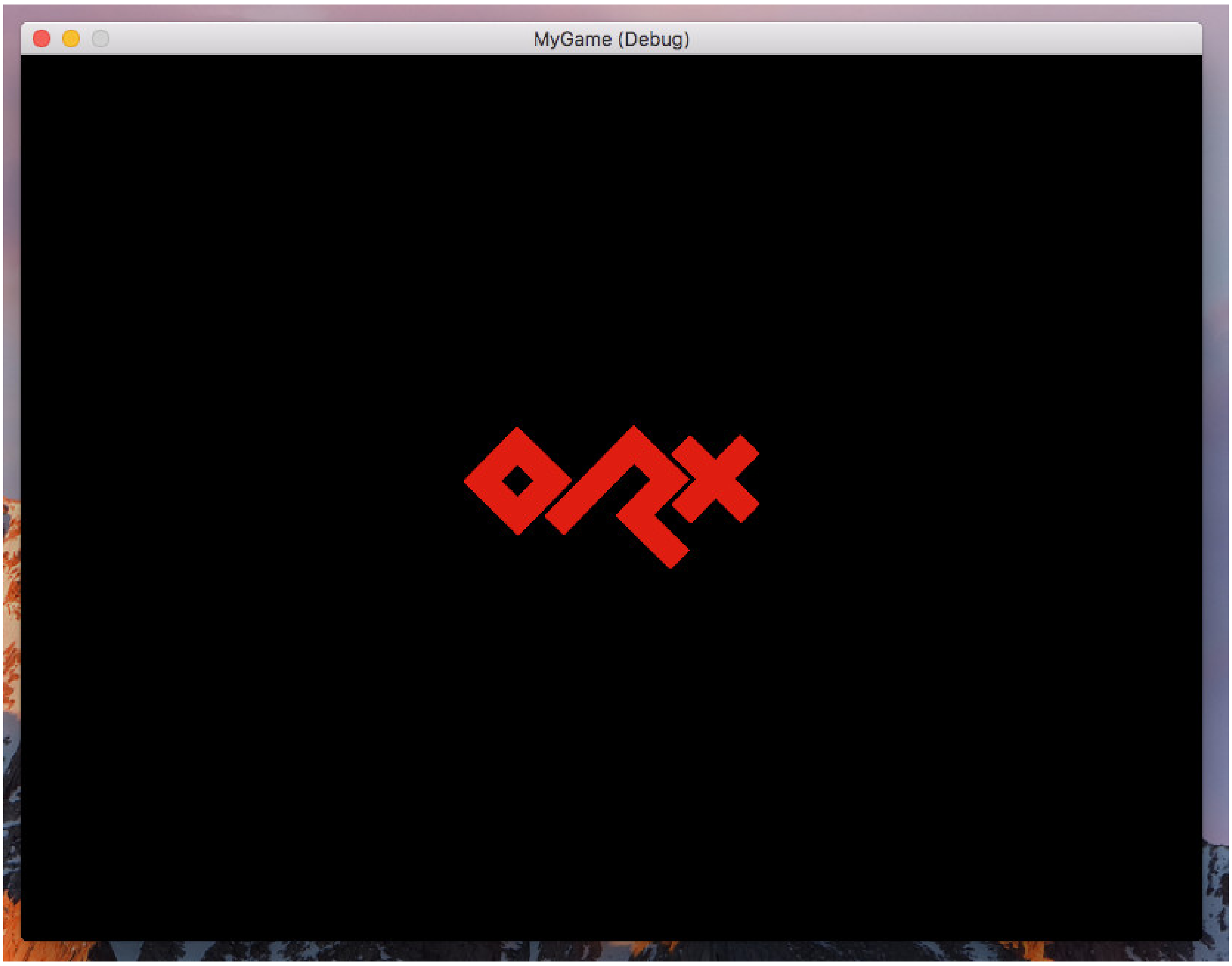
This step is used to copy the dylib(s) into the game's bin folder. How is the compiler able to do this if our game project is in a completely different place to the Orx library project? The answer is: the location stored in the $ORX environment variable.

The variable is taken from your `.bashrc` and `.profile` files. The variable holds the location of the Orx Includes and Orx libraries (dylib) to link to.

Running our game project

Now to run our compiled program.

- cd `~/Documents/MyGame/bin`
- Type: `./MyGamed`



You're all done! Congratulations on compiling the Orx library, and your very first Orx game project.

You might be interested to move onto the Beginner's Tutorial and create a platform game.

Troubleshooting

MacOS varies in it's treatment of environment variables, So depending on which MacOS version or Codelite version you have, you may need to perform some extra steps.

Environment Variable Problems

If when trying to compile you may get an error similar to the following:

```
cp: /Users/xxxx/Documents/orx/code/no/lib/dynamic/liborx*.dylib: No such
file or directory
make: *** [PostBuild] Error 1
```

or

```
/Users/xxxx/Documents/simplescroll/src/simplescroll.cpp:6:10: fatal error:
'orx.h' file not found
#include "orx.h"
         ^~~~~~~
```

This one is generally caused by the $ORX environment being set, however, the MacOS version or Codelite version not honouring the variable. In this case, you need to set the variable within Codelite.

To do this, go to: Settings / Environment Variables

Add the following line:

```
ORX=/Users/canberra-air/Documents/orx/code/
```

Debugging Problems

When trying to run the debugger you may get a dialog box saying:

```
Failed to locate gdb at ‘gdb’
```

Right click on your project. Then go to `General` / `Debugger` property (not the Debugger section), Switch from `GCC Debugger` to `LLDB Debugger`.