This document needs to be checked for relevance, and possibly be broken out into separate documents.

Xcode4 Scroll setup, Console-less apps and Resources

Orx project uses premake. If you cannot find project files for your favorite IDE you may be able to generate these files with premake. You can find premake in /extern/premake subdirectory of orx project tree.

Scroll Build Configuration in XCode 4

If all ORX projects are at the same level in the directory tree

```
./GameName/ contains game project
./orx/ contains orx clone
./scroll/ contains Scroll clone
```

Note: The model above is great for keeping up to date with external projects, but it does not work well if games have to depend on different versions of orx.

then the following search paths have to be set:

1. Header Search Paths (HEADER_SEARCH_PATHS):

```
../scroll/include/Scroll
../orx/code/include
```

2. Library search paths (LIBRARY_SEARCH_PATHS):

```
$(inherited)
"$(SRCROOT)/../orx/code/lib/dynamic"
```

Console-less Application on OS X

OS X application bundle must be built to run in console-less mode. Application bundle is a directory with .app extension. Here is a directory structure that worked for me:

```
Sample.app (directory content)
Contents\
Contents\Info.plist
```

```
Contents\MacOS\Sample  (executable file)
Contents\MacOS\Sample.ini (orx INI file)
Contents\MacOS\liborxd.dylib (orx dll)
Contents\MacOS\resources\  (app specific resources directory, specified in
INI)
```

You don't have to start from scratch. Simply create a Cocoa application from Xcode. Take its Info.plist file content and modify the content to match your needs.

Some of the parameters like Principal Class don't apply. I removed “Principal Class” and application still worked.

Related: Bundle Documentation

Converting Console Application to Console-less in XCode

Applies to XCode 4.6.2.

Console-less and Cocoa application terms are used interchangeably.

It is probably not the best idea to convert existing target from Console to Cocoa application. The better way is to simply create new Cocoa target and remove Objective-C code references.

The following is a log of steps I had to take to convert the existing target.

The project type has to be changed, so XCode can offer Cocoa specific configuration settings. Expand content of the project package file. Open project.pbxproj in the text editor. Search for productType. Console application's product type is set to `productType = “com.apple.product-type.tool”;`. Cocoa application is set to `productType = “com.apple.product-type.application”;`. Change product type to com.apple.product-type.application. Reopen XCode project. It now should have Summary and Info tabs in addition to Build Settings, Build Phases and Build Rules tabs.

Create Info.plist file. For example, file name can be based on target name such as TargetName-Info.plist.

Select Build Target → Summary tab. Click on Choose Info.plist file and select created Info.plist file.

Select build target → Build Settings

In Packaging Section fill preferably at target level

- Info.plist file name
- Product name
- Wrapper extension is set to app

XCode will now correctly generate the application bundle. However, any additional resources would have to be copied with Build Phases configuration.

Configuring Orx Resources

Orx expects INI file with name matching executable file name to be located in the same directory as the executable itself.

To add INI file:

```
1. Click on Project -> Target -> Build Phases -> Add Build Phase Button ->
select "Copy Files" phase.
2. Set Destination to Executables
3. Leave Subpath empty
4. Uncheck Copy only when installing.
5. Add your INI file. Make sure its name matches the executable.
```