Build Xcode Workspace and Project files for IlmBase and OpenEXR on MacOSX using cmake

Make sure you have at least Cmake 2.8.9.

Locate the Cmake application in your Applications folder and run it.

Note that if you have run Cmake here previously, the cmake cache may have values in it that will confuse it. Under Cmake's file menu, there is an option to delete the cache. After each Configure and Generate step, visually inspect the "Where is the source code" line, and each of the build variables. If the cache is interfering with the build process, these values will show wrong results. If this occurs, delete the cache, and back up a few steps until everything works consistently.

Also note that despite there being a cache, Cmake will forget the settings, so do not forget to re-add the ILMBASE_PACKAGE_PREFIX manually every time you run.

Let's assume the workspace and xcodeproj files will go in /Users/me/openexr/build, and that the installation will go into /Users/me/openexr/build/distro.

If we set it up that way, we can clean everything out and start over easily by simply deleting the build folder.

Point "Where is the source code" to the openexr/IlmBase folder.
Point "Where to build the binaries" to /Users/me/openexr/build/build/IlmBase
Point CMAKE_INSTALL_PREFIX to /Users/me/openexr/build/distro
Select Configure
Select the output to be Xcode, and use the default native compilers.
Select Generate

Now, Point "Where is the source code" to the openexr/OpenEXR folder.
Point "Where to build the binaries" to /Users/me/openexr/build/build/OpenEXR
Note that the binaries directory for IlmBase and OpenEXR have to be different, or Cmake will clobber the other project's set up.
Point the install to something like openexr/build/distro
Point CMAKE_INSTALL_PREFIX to /Users/me/openexr/build/distro
Use the Add Entry button to create a FILEPATH variable called ILMBASE_PACKAGE_PREFIX and set the value to /Users/me/openexr/build/distro
Select Configure
Select the output to be Xcode, and use the default native compilers.
Select Generate

A) Open Xcode. Open the IlmBase.xcodeproj. Switch the target to BUILD ALL. Build. Switch the target to Install, and build again.

B) Open the OpenEXR.xcodeproj. Build All. Switch the target to install and build again.

Now, if you open a terminal, and go to /Users/me/openexr/build/distro/bin, you will find all the OpenEXR tools there, and you will be able to run them from the command line.

There might be exr tools on your path elsewhere, so be sure to use ./ to make sure you get the one in your bin. ie, to run exrenvmap, use ./exrenvmap.

Every time you modify something you'll need to go back to A) or B) depending on where you made the modification. If you modify one of the exr programs, you'll need to run the install step separately before you can test.

Known issues:

The Cmakefiles are a work in progress.

- Each of the exr tools references each of the other tools include files via unwanted -I references.
- The install step puts the debug and release into distro, but does not differentiate them by name. The xcodeprojects reference the distro/lib folder, and so the OpenEXR tools build, but the xcodeprojects also reference the distro/lib/Debug folder, which does not exist. This means that currently it is necessary to build IlmBase in debug mode, build OpenEXR in debug mode, then do something like rename the distro folder to debugDistro; then switch IlmBase to release, build and install, switch OpenEXR to release build and install.
- It would be nice if the IlmBase projects were simply subprojects of OpenEXR, but it is not clear that Cmake can set things up this way.